TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

Master's Thesis

oPage
Framework for Web based Content Management Systems

carried out at the

Information Systems Institute
Distributed Systems Group
Technical University of Vienna

under the guidance of

Priv.-Doz. Dipl.-Ing. Dr.techn. Engin Kirda

by

Johannes Dorn
Weintraubengasse 7/4, A-1020 Vienna
Matr.Nr. 9326012

Vienna, 28th of February, 2008

Acknowledgements

I like to thank my advisor Priv.-Doz. Dipl.-Ing. Dr.techn. Engin Kirda, who has been a great help for me to improve my thesis, and was very generous every time I missed my deadline.

I owe a great deal to Gernot Ihrybauer for his creative input in the design process of oPage and for the lasting friendship. I still remember our tempered discussions at the Extone.

I have to thank my customers and the users of the oPage framework for their feedback and support.

Special thanks also go to Dr. Eva Ptak for her wise and humorous advocacies.

If you have troubles writing, visit the writers' studio of Mag. Judith Wolfsberger or one of her courses. Her advices made writing my thesis less complicated.

I am deeply grateful to my beloved girlfriend Christine, who encouraged and pushed me to complete this master thesis and finish my studies. I am sure, I would not have accomplished it without her sympathy and her support. And I would have a lot more misspellings in my thesis.

A big "hello, I did it" goes out to my parents and all my former school mates and university colleagues. Yes, I did it. Yes! Yes! Yes!

Hannes Dorn
Vienna, Austria, February 2008

Abstract

oPage is a framework for developing Web based content management systems. It is more specific than a general purpose framework like ASP.NET, but more flexible than content management systems like TYPO3 or Joomla!.

Web sites using oPage can run on every platform, where both PHP and MySQL are available, and are divided into the core system, the backend and the frontend. The core system contains the framework base classes and can be shared among different projects. The backend provides the user interface for administering the Web site and managing the content. The frontend is individually created for each Web site and can be customized and extended as needed.

oPage uses a database access layer and modules to store the content. For each type of structured content, a module class is used. Modules can be combined to represent the database structure and provide operations to create, retrieve, update and delete records and also operate as a report generator. The user interface can be extended by navigation, pager and form controls. In the controller script, the module and control objects are tied together. The content is queried with the module and merged with the template engine, which renders the output using page and content layout templates.

The robust infrastructure, the versatile template engine, the structured backend and the provided modules and controls make oPage a good foundation for Web sites.

BibTeX

```
@mastersthesis
{
    opage,
    author = "Hannes Dorn",
    title = "oPage - Framework for Web based Content Management Systems",
    school = "Technical University of Vienna",
    year = "2008",
    month = "February",
    address = "Vienna, Austria"
}
```

Zusammenfassung

oPage ist ein Basissystem (Framework) für die Entwicklung von Web-basierten Content Management Systemen. Es ist spezifischer als ein allgemeines Framework wie ASP.NET, aber flexibler als Content Management Systeme wie TYPO3 oder Joomla!.

Websites, die oPage verwenden, können auf allen Betriebssystemen laufen, für die PHP und MySQL verfügbar ist, und sind unterteilt in einen System-Kern, das Backend und das Frontend. Der System-Kern enthält die Basisystemklassen und kann in beliebigen Projekten verwendet werden. Das Backend stellt die Benutzeroberfläche für die Verwaltung der Website und die Bearbeitung des Inhalts zur Verfügung. Das Frontend wird für jede Website individuell erstellt und kann entsprechend angepaßt und erweitert werden.

oPage verwendet eine Datenbank-Abstraktions-Schicht und Module zum Speichern des Inhalts. Für jeden Inhaltstyp wird eine eigene Modulklasse verwendet. Um die Datenbankstruktur abzubilden, können Module auch verschachtelt werden. Die Modul-Basisklasse stellt Methoden zum Erzeugen, Lesen, Aktualisieren und Löschen von Datensätzen bereit und arbeitet auch als Report-Generator. Die Benutzeroberfläche kann mit Navigations–, Pager- und Formular-Elementen (Controls) erweitert werden. Im Ausgabeprogramm (Controller) werden die Modul- und Control-Objekte zusammengefügt. Der Datenbankinhalt wird mit den Modulen gelesen und an die Vorlagen-Verarbeitung (Template Engine) übergeben, die eine Webseite basierend auf Seiten- und Inhaltsvorlagen erstellt.

Die robuste Infrastruktur, die flexible Template Engine, das strukturierte Backend und die zur Verfügung gestellten Module und Controls machen oPage zu einer guten Basis für Websites.

BibTeX

```
@mastersthesis
{
    opage,
    author = "Hannes Dorn",
    title = "oPage - Framework for Web based Content Management Systems",
    school = "Technische Universität Wien",
    year = "2008",
    month = "Februar",
    address = "Wien, Österreich"
}
```

Contents

1 Introduction

Since the first Web site has been written by Tim Berners-Lee[1] on November 13th in 1989 at the European Organisation for Nuclear Research (CERN) and the development of the browser "Mosaic for X" by Marc Andreessen and Eric Bina[2] in 1993 the popularity of the Web rised tremendously. At the time of writing, Netcraft[3] reports 155,230,051 active Web sites.

The first generation of Web sites just consisted of a few static HTML pages. A Web master created those pages by hand or by using an HTML editor program. Over time more and more pages were added and management of the Web site became complex. Design changes had to be applied on every single page and editing could only be done by the Web master, who became more and more a bottleneck, so Web site owners started demanding to update the Web site by themselves.

Nowadays on many Web sites server side programs are used to create HTML pages as users request them. The content, which is stored in a database, is merged with a layout template into a HTML page, which finally is sent to the user. The Web master creates the layout templates, a programmer adds functions like handling user feedback and the Web site owner can add and change content in a secure manner.

Such programs are called Web content management systems. According to [Wika], "a content management system (CMS) is a system used to organise and facilitate collaborative creation of documents and other content". Depending on the purpose of the CMS, specializied types of CMS are used for Web (WCMS), enterprises (ECMS) or learning environments (LCMS) and other areas.

A Web content management system (WCMS) aids Web masters, programmers and authors in the process of creating, publishing and updating a Web site. Through separation of logic, content and layout it helps keeping a consistent look and feel and raises the overall quality of the user experience. WCMS can be installed as client software on the users desktop system or as Web application on a server using the Web browser as front end.

This thesis is about oPage, which I am developing. It is a framework for browser based content management systems and written in PHP (section A.1 on page 127), it is platform independent and uses a relational database to store the content. oPage has been used in a lot of projects [oPa, AKM, Dem] and is constantly enhanced.

This paper is intended for Web designers, Web masters, software engineers and interested Web users. It is useful if the reader has an idea of programming, Web technologies and object oriented concepts. This work is not about integrating a WCMS into an organisation, recommended readings: [Nak01, AES02, Hac02, ZTZ02]

1.1 Internet Services

The Internet as we know it today began in 1969 with the implementation of ARPANET by academic researchers under the sponsorship of the United States Department of Defense Advanced Research Projects Agency (ARPA). Their goal was to build a fault-tolerant decentralized communication infrastructure.

Simply speaking the Internet consists of computers which are connected by wires. By using standardized protocols[4] this computers can exchange data. Specialized protocols are used at every network layer, e.g. for transmitting raw signals over a cable (at the physical layer) or sending an e-mail (at the application layer). Hard- and software is used to connect computers to networks and these with other networks, e.g. a router can connect a local area network (LAN), as you may have it at home or at the office, with a wide area network (WAN) like the one your Internet service provider (ISP) has, which again is connected with other networks.

[1] Tim Berners-Lee invented the World Wide Web. Read the full story at `http://en.wikipedia.org/wiki/World_Wide_Web`.

[2] Marc Andreessen and Eric Bina developed the first version of the Mosaic browser. `http://en.wikipedia.org/wiki/Mosaic_(web_browser)`.

[3] Netcraft is an internet service company tracking internet technology. Numbers are taken from Netcraft December 2007 Web Server Survey `http://news.netcraft.com/archives/web_server_survey.html`.

[4] Read more about internet protocols at `http://en.wikipedia.org/wiki/TCP/IP`.

Internet services are network aware programs which offer some kind of function to other programs by using the Internet as their underlying infrastructure [MW94, KaC01]. Popular services are e-mail, file transfer, newsgroups and the World Wide Web. Newer topics are instant messaging, file sharing, internet radio and telephony. Also worth mentioning are the Domain Name System, Telnet, Secure Shell, Archie, Gopher and WAIS, but they are less known by the typical Internet User.

The most important Internet services (by today) are:

Electronic mail (email)

It is a way of sending and receiving messages electronically.

File Transfer (FTP)

Used to transfer files from one computer to another.

Newsgroups (News)

It is an electronic discussion system, where users can read posted messages and add new entries.

World Wide Web (WWW or just Web)

It is an information space which integrates text, images, animations, audio and video and supports cross-references. Users often mistakenly use it as a synonym for the Internet, although the Web is a service that operates on the Internet.

Domain Name System (DNS)

The Domain Name System is like a phone book for the Internet. A DNS server translates a Web address into an IP-address.

Telnet

With a Telnet client one can connect to a computer running a Telnet server to interactivly execute commands.

Secure Shell (SSH)

SSH provides file transfer and telnet service through secure encrypted communication.

1.2 Web Terminology

The Web is an information space which integrates text, images, animations, audio and video and supports cross-references. This mixture of different types of media is called hypermedia. Its predecessor is hypertext, which has been developed in the 1960s, and integrated only text and cross-references (called hyperlinks). Today both terms are used interchangeably. The history goes back until 1965 when Ted Nelson coined the word “hypertext”. In the 1980s some programs have been developed, including HyperCard by Apple Computer, but not until the broad availability of the Internet and the invention of the Web in 1990s the concept achieved a widespread success both in academic society and commercial business.

Web sites

The Web consists of countless Web sites providing information and offering services. A Web site is a collection of Web pages, which are provided by a Web server and displayed by a Web browser. Examples are Web sites with personal information, information about companies, products and services or the Web sites of newspapers, radio or TV stations. In colloquial German, *Homepage* is used instead of Web site[5]. Usually the home page is the main page of a Web site.

[5] Everywhere else, a mobile phone is called mobile phone, but we call it *Handy*

Web page

A Web page is a chunk of data that can contain text, markup information and references to resources like other pages, images or files. Each resource is addressed by an unique URL[6] which is simply speaking a combination of a hostname and a path. For example if you open the URL `http://www.opage.at/index.php`, your Web browser requests the Web page generated by the script "index.php" from the server "www" of the domain "opage.at" using the protocol "http". This page contains HTML"Hypertext markup language" code generated on the server upon request by a PHP[7] script.

Hypertext Markup Language (HTML)

HTML (and its successor XHTML) are markup languages to structure and format text. Cascading style sheets (CSS) provide a more flexible way of creating Web pages by writing the content in HTML and separated formatting instructions in CSS. Different designs can be created using the same HTML file just by applying different CSS instructions[8]. Additionally content can be stored on the Web server in XML[9] files and combined with XSL[10] or XSLT[11] instructions to prepare pages for different environments like personal computers and mobile devices.

Web application

"A Web application is a software system that is based on specifications of the World Wide Web consortium (W3C) and provides contents and services, which are used over an user interface, the Web browser" [KPRR04, p. 2]. It is "similar to a Web site in that it also presents related information in a uniform graphical layout. The focus of Web applications, however, lies in the application logic (functionality) offered via the Web" [Ker03, p. 10]. Examples are Web mail systems, online auction platforms, discussion forums or content management systems. A standard Web browser acts as front-end and communicates with a centrally installed service on a server by using the standard protocol HTTP. There is no need to distribute and install special client software. This reduces both development and maintainance costs, makes the application platform and operating system independent and allows decentralized data processing. Some limitations exist and not every piece of software should be developed as Web application. There are inconsistent implementations of HTML, CSS, JavaScript and other specifications and user interfaces developed in HTML are less powerful than regular client software: Drawing on the screen, drag and drop and interprocess communication with other client applications or the operating system are virtually not possible. Although some client side user interactivity can be provided through client side scripting, usually for every user interaction sending data to the Web server and reloading the Web page is required, which leads to longer response times compared to regular client software. This can be reduced by using Ajax, a technology using a combination of asynchronous JavaScript, CSS, DOM and XMLHttpRequest[12]. Frameworks like Prototype[13] and Scriptaculous[14] support Web developers in creating dynamic Web applications. But with all the limitations, Web applications are widely used [GM01].

Web application framework

Web application frameworks provide a modular infrastructure and are a reusable set of libraries, which are frequently used when writing Web applications. For design and implementation of applications methodologies and frameworks exist. By using a framework, productivity of the developers and the quality of software produced can be raised [Wöh04]. Commercial and free available Web frameworks for various programming languages and of different size and complexity are available, implementing miscellaneous approaches of application design. Examples are Struts, Velocity, Cocoon and Java Server Faces (all based on Java), ASP.NET (for C#, VB.NET and other .NET languages), Zope (using Python), Ruby on Rails (Ruby) and binarycloud, Hord or PEAR (for PHP), just to name a few.

---

6 URL: Uniform Resource Locator

7 PHP is a programming language used on the Web. Others are Perl, Phyton, ASP or .NET languages like C# or VB.NET.

8 See `http://www.csszengarden.com` for a very nice example of how to restyle a Web page using CSS.

9 XML: eXtensible Markup Language

10 XSL: Extensible Stylesheet Language

11 XSLT: XSL Transformation

12 See `http://www.adaptivepath.com/publications/essays/archives/000385.php` for an overview.

13 `http://prototype.conio.net/`

14 `http://script.aculo.us`

Modern Web applications are complex software systems. Although practices of traditional software engineering can be applied, the characteristics of Web applications require additional methods of software development [DH01]. According to [Low99] Web engineering is "the employment of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of Web sites and Web applications".

A Web content management system (WCMS) is an application for managing text, graphics, links and other information of a Web site. Its primary focus lies on Web publishing and its main principle is the separation of content, logic and layout [Wöh04].

1.3 Motivation

These days it is mandatory for organisations to be present on the Web. Thereby the fact is often underestimated that content and structure have to be maintained [JM02].

I want to outline the way of developing oPage by describing my personal involvement with the Web. This started with some "hacked together" pages and leaded me to a CMS framework.

Since my first contact with the Web in 1994, the way I am creating Web sites changed a lot. I prepared the first Web pages using a text editor. It worked, but was not very convenient, and adding another page involved changing the navigational links in every other pages by hand. The next time I used an offline WYSIWYG word processor which also managed the navigational links. The initial Web site was easy done, but after the site started growing, managing became time consuming since nearly each change required to recreate and upload all pages of the Web site. Also it was hard to keep a consistent page layout because this program offered to much design freedom in the content areas.

I like to automate things and I did not want to get to deep into HTML coding. Write it once and run it forever. I created a database, some forms to enter data and a program written in BASIC to read the data and create static HTML pages, which could be uploaded to the server. This was my first content management system (I had no idea that it was a CMS, but I knew that this was something to hang on with). I extended this program to create a Web site, which is still in use today. This was the first step in content-layout separation. Although it was possible to change fonts, colors and some icons through the user interface, HTML code was deeply embedded into the program and the pages were still created offline and uploaded to the server.

With the introduction of Microsoft Active Server Pages (ASP), the language Basic entered the Web. I started writing classes which could be useful for creating a Web site. These classes were used to hide the HTML code from the programmer. A base page class created a frame with navigational links and a derived class[15] created the individual page content using module classes, which read the data through a database class.

In the next project at another company I developed an onlineshop. Once again a database and some forms were used to enter data. But the Web pages were created dynamically upon request. The next step was to add forms to the Web site, where authors could submit and change content.

Object orientation is limitated in ASP, so I searched for something else and became interested in PHP. For a project I rewrote my ASP classes in PHP, but still with the idea to hide the HTML code behind some source code.

One more project had to be done to show, that more flexibility in layout and design was needed. Another problem was, that a programmer was needed to write the code for the Web pages, because it was to complicated for a Web designer, who usually do not have programming skills.

I needed something with the following characteristics:

- used with standard Web browser
- structured in reusable modular parts
- easy to use for content managers
- easy to understand by Web designers
- easy to extend by programmers

[15] There is no inheritance in ASP, it was done by nesting the instance of the parent class into the derived class.

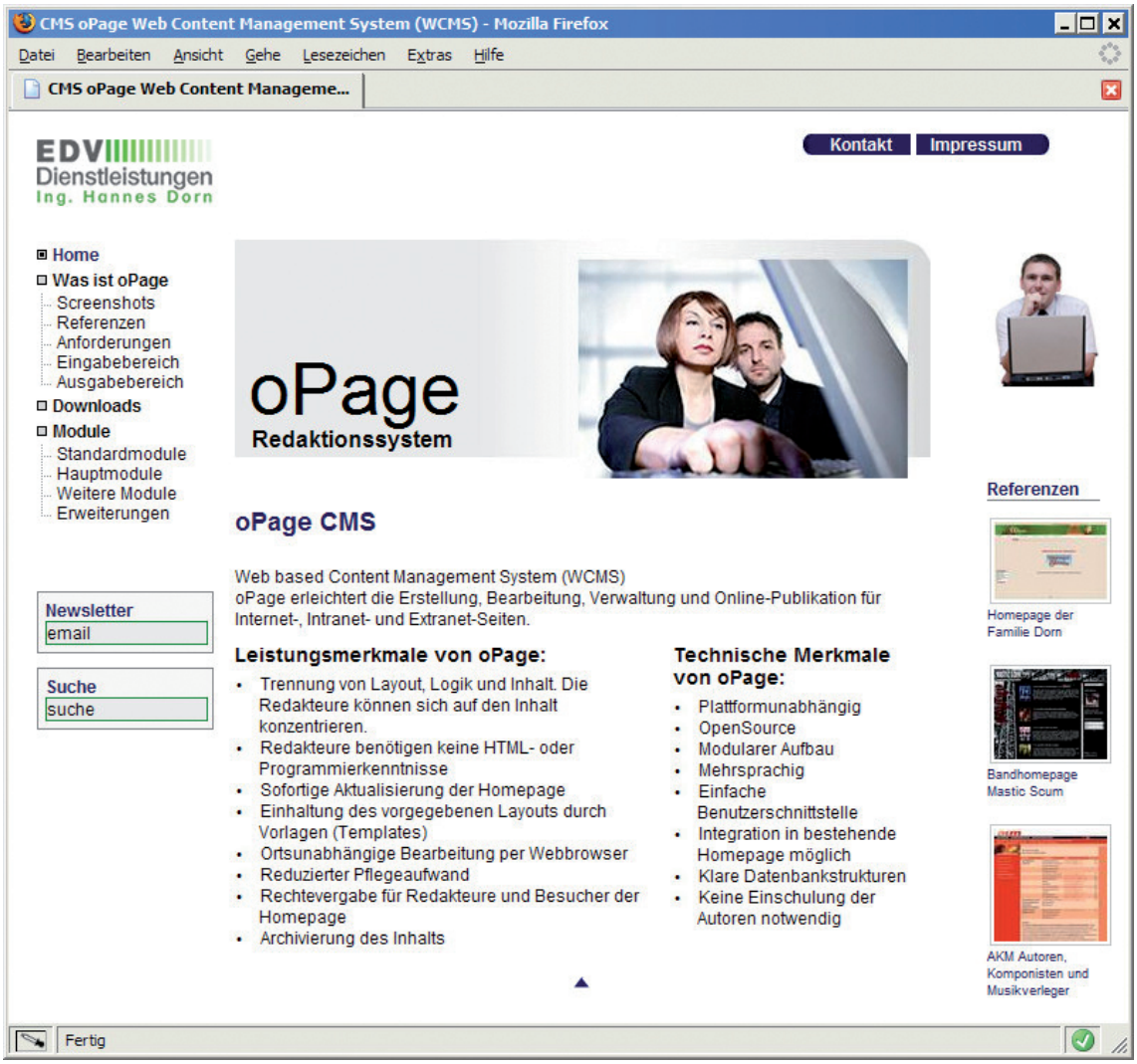


Figure 1: *Web site www.opage.at*

- easy to install by system administrators
- not limiting design ideas

In this paper I am introducing the oPage framework for Web based content management systems. Software developers can use oPage in their projects to create customer specific Web sites and content management systems. Web designers can take advantage of preprogrammed modules and apply their own design by editing just a few templates and CSS files without the need of programming experience.

oPage provides a useful infrastructure for developing Web applications:

- Model-View-Controller (MVC) object oriented software architecture
- fast template engine
- base class for data retrieval and output
- formular engine for single and multiline forms
- serverside image manipulation
- import and export interface
- database abstraction layer
- support for multiple languages
- extendible environment for content editing

Users of oPage experience a steep learning-curve and get instant results. It is easy to migrate an existing

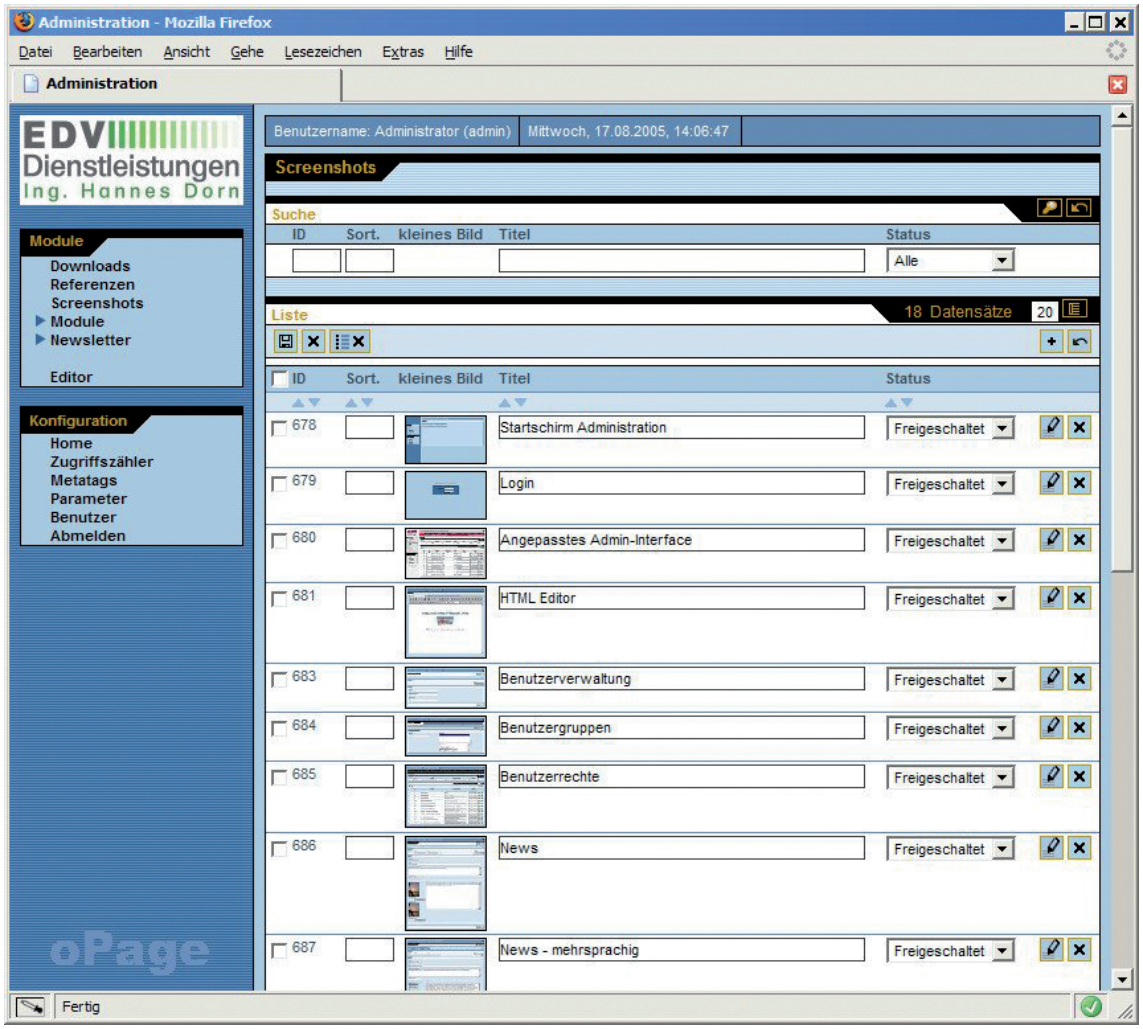


Figure 2: *Administration interface of www.opage.at*

Web site to oPage or integrate oPage just at a few places of a given Web site.

Since the first draft of oPage, it has been extended alot to fulfill the various requirements of Web sites created with it. This covers simple guestbooks, private homepages, company Web sites, online stores of different sizes, discussion forums and newsletter systems. References can be found at [oPa].

1.4 Problem Definition

The main subject was to create a reusable framework for Web based applications, which can be chosen to develop platform independent content management systems for various purposes. It has to work with minimal system prerequirements, and a typical Web space of an average internet service provider has to be sufficient for hosting Web sites using oPage. Typical used programs used on Web servers are Linux, Apache, MySQL and PHP[16], but it should run as well with Windows, IIS, PHP and various database management system.

Getting started with oPage should be very easy and providing instant results to users. Web sites created with oPage have to follow Web standards and have to be compatible with all major Web browsers.

The administration user interface (the backend) has to offer a WYSIWYG word processor and has to be robust against faulty input. Authors should not need special training.

[16] This combination is often referred as LAMP

The development of the framework must follow a clear path, so that other developers can extend the system in the fields of ecommerce, sending newsletters, discussion forums or picture galleries.

1.5 Organisation of this thesis

Chapter 1 gives background information on Internet, Hypermedia and WWW. Terminology is explained as well as the purpose of this work.

Chapter 2 discusses the state of the art of WCMS and what kind of functions are provided. System classification criteria are defined and related approaches are shown.

Chapter 3 covers oPage. By means of a small and easy understandable sample Web page, it is described how oPage is designed. A diagram and a class tree gives an overview of all areas of oPage. Main classes are outlined and samples are enclosed to demonstrate how to customize and extend the framework.

Chapter 4 evaluates what has been accomplished, and tells the lessons we learned.

Chapter 5 gives an outlook on potential future extensions.

Chapter 6 concludes the thesis and summarizes the major contributions of this work.

2 Web Content Management Systems

This chapter provides an overview about what Web Content Management Systems are. Beginning with a general definition of content management, the principles of Web content management are explained. The important topics like authoring, storing, workflow and publishing are described as well as the different user roles involved in a Web project. Four different Open Source Content Management Systems (TYPO3, Drupal, Joomla! and eZ publish) are described and my opinion on their strengths and weaknesses are listed.

"A Web Content Management System is a type of Content management system software used for managing Web sites." [Wikb]

Web	**Content**	**Management**
Internet Intranet Extranet	Text Images Sounds Videos ...	Creating Processing Managing Publishing Archiving
System		

Figure 3: *Web Content Management System [ZTZ02, p. 70]*

The CMS stores content (text, graphics, links, etc.) for distribution on a Web server and provides tools where users can create and manage content with little or no knowledge of HTML. "Most systems use a database to hold content, and a presentation layer displays the content to regular Web site visitors based on a set of templates" [Wikb]. The systems usually provide a user interface which can be accessed using a Web browser.

2.1 Content Management

"Content management is the systematic and structured way of procuring, creating, treating, managing, presenting, processing, publicating and reusing content" [RR01]. This definition is independent of the use of electronic devices [JM02].

In "traditional publishing" [RR01, p. 5-37], independent industries are involved in creating and processing different types of media like texts, sounds, images or movies. In this traditional publishing process the author creates the texts, the graphic artist prepares the images and the layouter combines everything into a page of a magazine or book or brochure. In each step, separate programs are used by experienced professionals: The author uses a word processor, the graphic artist an image editing program and the layouter a precise layouting program. Nowadays corporations and private persons have joined publishing companies and advertising agencies in distributing information. This drives the need for special tools which satisfy the expectations of professionals as well as the requirement of occasionally (or inexperienced) users.

Managing content to meet the diverse demands of a large user community requires four primary components [Hac02, p. 52], [BL01]:

- an environment for creating and acquiring content (**authoring**)
- a repository for storing and retrieving content (**storing**)
- a method for assembling and linking content (**workflow**)
- a delivery mechanism for delivering content to your customers (**publishing**)

Core features of a content management system (CMS of first order [RR01, p. 64]) are

- structured store for typed content and meta information
- access control
- protocol functions
- check-in and check-out mechanism

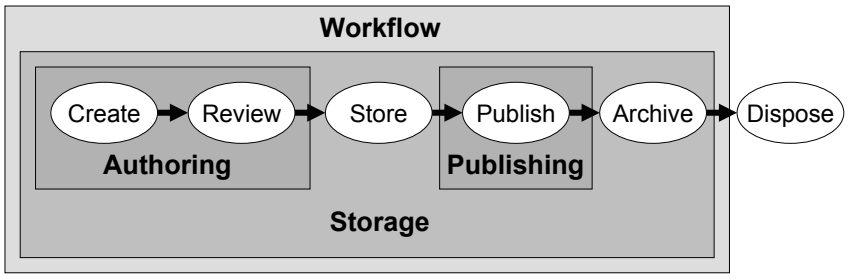


Figure 4: *Content Life Cycle [BL01]*

- query operations
- multiuser system
- mass operations
- management instruments

Additional operations (CMS of second order [RR01, p. 65]) include workflow, data aggregation and relationship management, a design tool, version control, a protocol log for atomic changes, backup and rollback methods and the support of foreign formats and multiple languages.

By using a CMS, content is [Hac02, p. 77]:

- more easily accessible to everyone within the organisation
- available at one virtual location without having users search on multiple servers or individual hard drives
- a single point of source, so that users can be confident, that they have found the latest version of the information
- labeled according to when and by whom it was written
- tracked through a workflow process that records when and by whom it was modified
- managed under a security process, which ensures, that it can be checked out and modified by certain authors, read by selected users, and not accessed at all by other individuals in the organisation

"Fundamentally, a CMS devolves control over content to the owners of that content (rather than the technician), and then scales without increasing management overheads" [BL01].

There are several types of content management systems:

- Document management
- Knowledge management
- Media asset management
- (Intranet) groupware
- Enterprise content management
- Web Content Management Systems
- Specialized WCMS for portals, Wikis[17], Blogs[18], calendars, forums, galleries, e–Learning or online shops.

In this work only Web Content Management Systems are covered.

2.2 What is a WCMS

A Web content management system is an application for managing text, graphics, links and other information of a Web site. Its primary focus lies on Web publishing and its main principle is the separation of logic, content and layout [Wöh04]. Also enclosed are functions for Web site management [WS00, p. 1334] like file management or link checking (see figure 5 on the next page).

[17] WikiWikiWeb. A form of hypertext document where every user can be an author. [Lei01]
[18] Blog is an artificial word combined from Web and log.

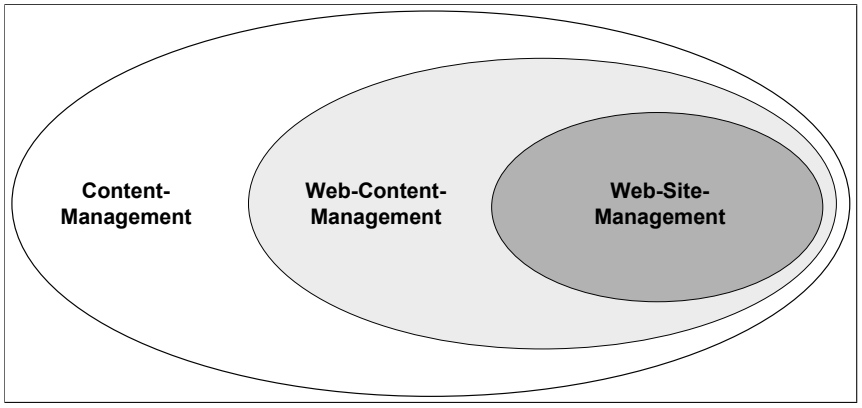


Figure 5: *Relationsship between Web site management and content management [WS00, p. 1334]*

2.2.1 *Web* in WCMS

The term "Web" both stands for Web site and Web application: Web site because we talk about content management of Web sites, Web application because we talk about Web based content management systems.

The main focus of a Web site is to provide information, while a Web application offers some kind of service [Ker03, p. 1]. Is an online shop a Web site or a Web application? The part, providing information about the articles is a Web site while the part with shopping cart, customer registration and order form is a Web application. Since a WCMS is always a Web application, a Web site built with a CMS will also be a Web application. In this work, the term Web site is used synonymously with Web application.

Another usage of this term is in Web standards and technology, like they are used on Internet, Intranet and Extranet[19], which are disdinquished by their user groups. Internet Web sites can be accessed by everybody, Intranet Web sites just by employees via the company network, the Extranet, a special part of the internal network, can be accessed by external users with username and password.

2.2.2 *Content* in WCMS

Information handled in the authoring environment might include [Hac02, p. 60]:

- information authored by members of an organisation
- information brought in from outside an organisation (third-party)
- information licensed by an organisation from external information providers (syndication)

I extend this list with

- information coming from internal applications
- information acquired from Web applications

Data of internal applications could be shop-items which are exported from a stock keeping application to the database of the company's online store as well as reports from the accountings database. Data of Web applications are orders, newsletter subscriptions or entries in the guestbook, online forum or FAQ database.

Important content elements are [RR01]

- Unstructured text
- Structured documents
- Formated documents
- Information in databases
- Binary objects or files: 2D and 3D images, audio clips, videos, flash movies or document files like PDF, word processor, spread sheets and others.

[19] Internet, Intranet and Extranet are sometimes referred as I*net

But what about elements for user interaction, navigational items like buttons and links, or forms to subscribe a newsletter or the shopping cart of an online store? These parts of a Web site are not typically seen as content, but need also to be managed.

2.2.3 *Management* in WCMS

"Management"[20] in WCMS covers "things we have to take care of". This includes of course processes dealing with content, but also the administrative tasks of the WCMS itself.

Authors create, process and publish content using the content management system. They start their Web browser and access the administration part of their Web site. There they select the area, e.g. the news section, where they want to add an entry or change one. Then they type their news into a form and submit it to the server. The CMS validates the entered data and stores the content in its database. Depending on the system authors can apply formatting like bold or underline. Also options for structuring the text into headlines and paragraphs may be available. Multimedia objects can be added either by uploading files from the local computer to the server or by selecting items from the media library within the CMS. This images, sounds and videos can be processed in various manners, e.g. an image has to have a given size to fit into a predefined frame or a version with reduced quality (and filesize) has to be prepared for users with slow internet connections. Exporting existing content for further use can be done in different formats like CSV, XML, PDF or RTF files. Even text could be converted by a speech engine into a sound file.

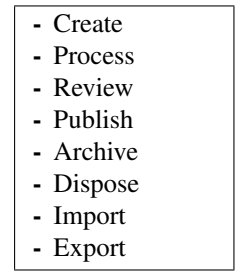


Figure 6: *Content Management*

If everything is ready to be published, the author forwards the entry to the editor for review. This is either done automatically by a workflow system or can be done manually by changing the associated user. The editor reviews the news article and aprooves it or sends it back to the author for a rewrite.

In most systems, entries can be equipped with an expiration date, after which they are not be displayed any longer. Also a starting date may be set. Instead of taking the entries from the Web site, the state can be set to archive, where the content may be not displayed on the main page any longer, but is still available through fulltext search.

2.2.4 *System* in WCMS

A broad range of systems is needed to run a WCMS. There are hard- and software systems. Hardware can be as little as a shared Web space at an Internet Service Provider or as big as a cluster of world wide distributed computer systems.

Software like operating system, Web server, mail server and database management system have to be taken into consideration. Depending on the size of your organisation and the specification of your Web site, different kinds of configurations can be selected.

For smaller Web sites, a shared Web space at an Internet service provider may be sufficient. This Web space usually includes an Internet domain address, email services, space to store the files of your Web site and services for running server side programs written in Perl, PHP or .NET.

Ambitious Web sites may require dedicated servers. This are computers where only your application is running, in contrast to the shared Web space, where many Web sites share the same resources.

The server can be placed in a data center at an Internet Service Provider (ISP), it can be rented from and/or managed by the ISP. If you have a fast and reliable Internet connection, you can place your server at your own place. This opens up more possibilities but requires additional infrastructure (backup Internet connection, uninteruptable power supply, standby servers) for running a 24/7 system (which Web sites usually are).

[20] from Old French ménagement "the art of conducting, directing", `http://en.wikipedia.org/wiki/Management`

2.3 WCMS roles

In this section, roles are described, which has to be filled in a typical Web site project. Depending on the size of the project, members can play multiple roles.

System administrator

A system administrator is responsible for the operating system, Web server, database management software and all other system programs.

CMS administrator

The CMS administrator has the highest privilege level of all users and handles tasks like creating user accounts, setting access privileges and other management tasks.

Designer

This person is responsible for the look and feel of the Web site and creates graphical previews of the Web site, which are used as page templates for the CMS.

Webmaster

A Webmaster integrates the templates of the designer into the CMS. Depending on the CMS, the Webmaster may also be able to create and configure the Web site and customize the CMS.

Developer

The developer knows the API of the CMS and is able to extend the CMS to meet customer requirements. Examples are customer specific extensions or the integration of external data sources.

Editor

The editor is responsible for the content displayed on the Web site. He may be responsible for the whole Web site or just for a part of it. As part of the workflow he reads articles written by authors. If an article fulfills all requirements, he changes the state of the article that it will be displayed on the Web site. If an article needs to be changed, he sends it back to the authors.

Author

The author submits articles into the CMS. This can be writing text, and also preparing pictures. It may be formatting and linking the content with other articles. Usually, authors are not allowed to publish their articles without authorization of an editor.

User

In this paper, a user is a person, who visits the Web site. In contrast to the other roles, he or she has no access to the CMS. Still, users may add content to the Web site through message boards, forums or Wikis.

2.4 WCMS functions

In this section, the main functions and features of modern WCMS are described. figure 7 on the following page shows the CMS feature onion. Although this figure is from a paper dated 2001 (which is nearly medieval in Internet times), in my opinion it is still relevant.

2.4.1 Versioning

In a multi user environment versioning provides a save way to edit content without overwriting each others changes. Also older versions can be recalled. Two kinds of versioning approaches exists.

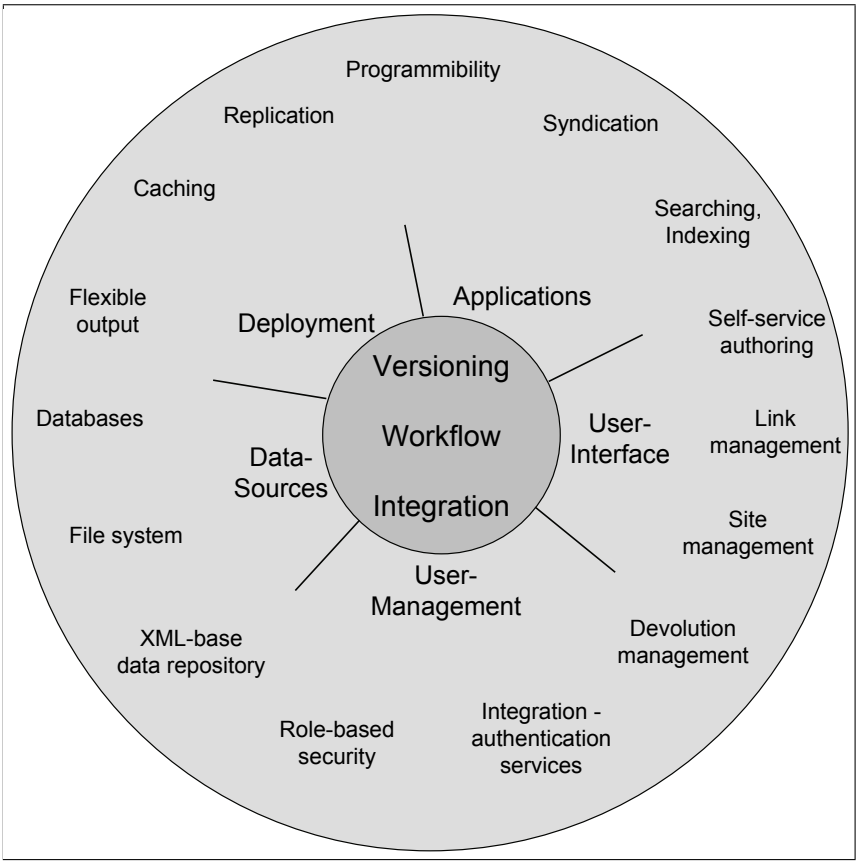


Figure 7: *The CMS feature onion [BL01, p. 6]*

The first is called **checkin/checkout**. Users either have explicitly checkout a record before they can modify it or this is implicitly done by the system. Only one user can have checked out an individual record. After the user has finished working, the record has to be checked in again. If the data has been modified, a new version is created.

In the second approach, a record gets not locked, instead the system saves the state of the record at the start of the modification and compares this state with the state at the time of updating. If it has changed, a new version is created and the new data is stored. If another user applied updates in the meantime, the first user is notified and the new data is rejected.

The advantage of the checkin/checkout approach is, that it is rather easy to work with, but hinders parallel work. The second aproach is more complex and version conflicts can occur. In CMS implicit checkin/checkout is usually used.

In my personal opinion, record locking is optional in smaller project, but is crucial when a larger group of content managers is working with the system. Although I never had a request for versioning in the Web projects I was involved, versioning is still a nice feature to track changes or restore a previous state of a record if something happend.

2.4.2 Workflow

A workflow is a process, where a document is tracked “through a number of individuals who are required to take some action such as edit, review or approval” [Hac02, p. 61]. This can be as simple as each record having a field for the current owner (e.g. an editor or an author). The other end is a flexible, event driven workflow system, with which custom routes for different types of content can be built as needed. People involved in the workflow are notified about changes either by receiving an email or by having new entries in their workbox.

A workflow system is a technical implementation of a formal organisational process. In my experience, it often hinders more then it helps. I prefer the simple version, where a user is attached to each record as owner and a mechanism to display only records for a selected user.

2.4.3 Integration

Templating

Templates separate the layout from the content and the application logic, and are merged with the content to produce the final output. Design templates give the Web site a uniform look and are applied on the whole site, while content templates are related to specific types of content. Different types are used for shop items and news entries, while both the shop as well as the news section make use of the same design templates. In general to change the design of the Web site, only the design templates need to be modified.

Asset management

As described in section 2.2.2 on page 11 different types of content has to be managed.

Textual content can be typed directly into the content management system using the user interface provided. External files can be uploaded from the authors desktop computer to the Web server using internet protocols like HTTP form upload, FTP or WebDAV and are stored in the filesystem (see also section 2.4.4 on the following page). In a multiuser environment record locking provides a safe way to make changes without getting in the way of someone else (see section 2.4.1 on page 13). It should be easy to create extensions for managing custom structured content.

How can be unstructured content like plain text, word processor or PDF files be integrated into the content repository? Can they be automatically converted to HTML, are they included in the fulltext search? Which output formats are available? Beside of HTML, content could be needed in formats like plain text, XML, rich text or spread sheet (see section 2.4.9 on page 19).

Some assets may have to prepared before putting them on the Web site. For large images, a thumbnail needs to be created, the size of the image may be reduced, or the image may be cropped, rotated or converted to gray scale. This is referred as server side image manipulation. Some systems even offer ways to create graphical headlines, banners, menu items and buttons dynamically from text.

Metadata

Metadata are data about data, each Web page should contain metadata like keywords, an abstract or a description as well as information for search engine spiders, if the page should be included[21] into the search index. Content fragments can be equipped with metadata for personalisation. Images should be stored along with size, resolution, place of origin and name of the photographer or copyright holder. The "Semantic Web" [BLHL01] was created to make content *understandable* by computers. An example is the XML based Resource Description Framework (RDF) format [RDF], which can be used for content syndication.

Staging

Changes of a Web site should not be done live. Broken links to pages could occur or applications developed may contain bugs. Usually only two stages are used, staging and production. The staging site is only available for a restricted user group, while the production site can be accessed by the regular (Internet) users. Depending on the CMS, content management is either done at the staging site or at the production site.

Personal note: I extend this list with two more stages. The development stage is at the very beginning, where software developers make modifications on the Web site. A programmer has a copy of the Web site on his computer and submits changes to a repository of a version management software. At the next stage, which I call testing, unit- and integration-tests are performed, before the site is copied to the staging environment. In smaller projects, only development and production stages exist. If multiple developers are

[21] excluded from search engine results should be pages only containing navigational links or the print version of a page

working on a site, a testing environment is vital. In my projects content management is only done at the production site. Staging of the Web site is done in case of major changes, to have an environment, where the customer can review the site.

2.4.4 Data repository

Data repositories are databases, XML-based storages or files. Most of the CMS require a RDBMS, for simpler systems the filesystem is sufficient. XML formatted content can be stored in special XML databases, or in relational databases as well as in files. For performance reasons, binary objects like images or pdf files are usually stored in the filesystem, while their filenames reside in database fields.

An interesting part is the integration of external data sources. For example a warehouse management software keeps track on article information like article code, title and the quantity in stock. This information should be reused for the online shop, but it needs to be extended by article images, description, size or color. Since the original tables can not be changed, some extra tables have to be created and a user interface has to be provided. The online shop may be on a different server at a different location and a direct connection may not be up for 24/7. A replication software can be used to synchronise the article entries of the online shop with the warehouse system. In the other way round, online shop orders have to be imported as purchase orders into warehouse software. Order state changes and shipping tracking numbers may also be presented on the Web site.

2.4.5 User management

This topic covers author management (as defined in section 2.3 on page 13). Accounts and groups can be managed, accounts and rights can be assigned to groups, to prevent authors from accessing areas, where they have not the permisson for changes. An existing user database may be used for authentication. Authors can be limited to update the Web site content only, can be restricted to certain areas of the Web site or limited to only perform certain operations.

Personal opinion: In my projects only a small group of people works with the CMS and updates the Web site. I never had the need to integrate an existing user base with LDAP or Active Directory, since this services are usually not accessible from the outside, but the Web site is usually hosted outside. This may be different for organisations like universities or big enterprises.

2.4.6 User interface

Editing interface

Browser-based client and administration tools allow authors to work from any location and minimise deployment and support costs. Non-technical authors should be able to submit content directly either to a staging environment or the live Web site, people with average knowledge of word processing applications can create content easily, without HTML or programming skills. But the user interface must as well fulfill the needs of experienced designers and developers. Both desktop Web creation software[22] should be integrated as well as development tools (IDE[23]) or text authors[24]. Authors should be able to preview their changes before they submit them to the repository.

Managing the site structure

The Webmaster, site designers or authors with the appropriate permissions can define the site structure, lay out section hierarchies and create necessary page entries. Based on this data the navigation controls are automatically filled and a corresponding site map is created.

[22] e.g. Microsoft Frontpage, Adobe GoLive!, Makromedia Dreamweaver
[23] integrated development environment
[24] my favorite is UltraEdit

Session analysis and reporting

Most interesting are logfile analyses for access statistics, as well as error reporting. But the nature of the CMS works against it, because in many systems, each page is created by calling the same script with a non-selfexplaining parameter. Usually included in the CMS are access counters for pages and content items. Extensive systems contain tools to measure page sizes and the performance of the Web site, and report possible slow pages and broken links. High end systems use tracking cookies or counting pixels to identify user visits and entry- and exit-pages. Counters can be activated for each fragment and are increased every time, the fragment is displayed. Questions like how often was an external link clicked, or how often was a file downloaded in the last two weeks can be answered.

Session analysis tracks how users use the Web site, which areas are popular, what are the entry- and exit-pages, which way took the user through the Web site and how long was he on the Web site. A nice feature is to display the current active user number on the Web site (if it is not too embarrassing because of low numbers). To take part in the official Austrian Web site statistics[25] tracking- and statistic-software has to be integrated. A special code has to be placed on each Web page to track user activity as well as to calculate overall access statistics.

2.4.7 Applications

Software architecture

Software engineering has developed a lot of different ways to structure complex programs. Well known examples are the monolithic system, client-server, the three- or multi-tier model, software componentry, peer-to-peer or the service-oriented architecture.

Above this structures, design patterns have been described to provide reusable solutions for common problems. Architectural patterns are Model-View-Controller (MVC), Presentation-Abstraction-Control (PAC) or the microkernel [Avg05].

"A robust software architecture is said to be one that exhibits an optimal degree of fault-tolerance, backward compatibility, forward compatibility, extensibility, reliability, maintainability, availability, serviceability, usability, and such other ilities as necessary and/or desirable."[26] That is what we want.

Security

Can software be secure? Unfortunately, software systems leave plenty of room to undermine security since programs can be modified. Microsoft has been (and still is) confronted with a high number of security issues in its products. Since more people and businesses rely on computing every day, Microsoft formed the "Trustworthy Computing" in January 2002, to focus on solid engineering and best practices to ensure the delivered product or service is more reliable and secure[27].

Secure by Design[28] means that design of the software is made to be secure, that it can not be abused, and to minimise the impact if someone breaks in. E.g. a Web server should run with the least amount of privileges possible.

Secure by Default[29] means that security is prefered over user friendliness. E.g. a fresh installed mail server is locked down and can not be accessed by other computers at all.

Common API

For customization and extending the CMS provides an application programming interface (API), which enables software developers and third party vendors to build a tailor made solution. For open source products, various repositories[30] exist. Extensions (some times called modules) like News, Gallery, Shop,

[25] http://www.oewa.at/
[26] http://en.wikipedia.org/wiki/Software_architecture
[27] http://www.microsoft.com/security
[28] http://en.wikipedia.org/wiki/Secure_by_design
[29] http://en.wikipedia.org/wiki/Secure_by_default
[30] e.g. http://typo3.org/extensions/

Blog or Wiki can be downloaded for free from there. They can be installed by the Webmaster with just a few clicks and are integrated seamlessly into the CMS.

Integration of external tools

Although management of advertisement can be done with the CMS itself, it is more common to use the service of an advertisement agency or to take part in the promotion programs of Amazon, Ebay or Google. The CMS must be able to embed this content comming from external servers into the Web pages.

Weather information or stock quotes can be grabbed from other Web sites. The CMS provides a way to configure the access to the Web service and to retrieve the information.

Localisation

If needed content should be provided in various languages. Texts, templates and images may be language dependent. At any time additional languages can be added.

Programmability

Is it possible to automate tasks by creating scripts within the CMS. Systems written in a serverside scripting language (like PHP or ASP) can easily be scripted by extending the source code. But systems which are on the server as compiled binaries (e.g. .NET) have to provide a special interface. Depending on the complexity of the API, an author or the Webmaster can prepare scripts otherwise an experienced programmer is required.

Scheduling

For each fragment, a starting and ending timestamp can be set. The item will be displayed on the Web site only between the defined time frame.

Searching and indexing

Search- and index-tools are often part of the CMS. Users can select if they want to scan the whole site or just a part of it, e.g. only the news section. If multiple sections are searched, the resulting items can be group by their associated section. Each module has to offer a interface which is called by the search engine of the CMS. A fulltext index of database entries is provided by the DBMS. For content in flat files, a fulltext index has to be built to avoid going through all files on each request.

Syndication

If you have valuable content, which is interesting to other Web sites, you can increase the popularity of your own site by making it available to other Web masters. They can integrate the data you provide and enrich their own site. They usually get only abstracts a link back to your Web site for the full-length article. Although content could be made available in HTML or text, it is usually done in XML. Popular XML syndication formats are Really Simple Syndication (RSS)[31] and Atom[32].

Personalisation

User accounts are stored in a membership database. Either users can register them self or are registered upon request. Users can be assigned to groups, for which content is made available through the CMS. The same membership database can be used for sending newsletters, the online shop or a discussion forum.

Examples for personalisation: On the main entry page, the user can be welcomed with a personal address. For a family gallery, pictures can be categoriesed into private, relatives, friends and public. Public pictures are accessible by everyone, all other pictures require a previous login of the user. Images are display according to the status level of the user. Depending on the items a visitor has in his shopping cart, the online shop can suggest other items, which the user may have interest in. E.g. if the user buys tea, he may need sugar or a tea strainer as well.

[31] http://en.wikipedia.org/wiki/RSS_(file_format)
[32] http://en.wikipedia.org/wiki/Atom_(standard)

2.4.8 Import/Export

All or just a group of items can be exported. Import has to support complete replacement or the synchronisation of the stored data with the new data. Content can be processed as CSV, XML or an application specific format.

For example: On a Web site CSV formated shop items are imported. These records are created on the warehouse server and uploaded automatically with a script on a regular base. To reduce the size of the list, only changed items are processed. Later a partner uses the export function to get the articles into his online-shop.

2.4.9 Deployment

Web sites are accessed using different devices with different capabilities. Screens of desktop computers have a higher resolution than those of pocket computers or mobile phones. HTML is not always the format requested. Content can be produced in formats like PDF, RTF, XML or MP3.

Web compatibility

According to the Web Accessibility Initiative (WAI[33]) Web sites have to be made accessible for people with disabilities. This requires e.g. filling out the alternative text attribute of images and making use auf cascading stylesheets (CSS) and XHTML instead of old HTML and tables. Programs[34] exist to check Web sites for compatibility with Web Content Accessibility Guidelines[35].

Caching

To reduce the workload on the server caused by dynamic Web sites, complete pages or fragments are cached[CDIW05]. The tricky part is to determine, which parts are outdated and need to be recreated. High volume Web sites use a load balancer and multiple production servers. These servers can be located at the same place or distributed worldwide. The content is replicated to each production server from the staging server. While this is easy for read-only content, updates made by users have to be planned carefully to avoid synchronization conflicts.

Search engine optimization

Search engines treat dynamic Web sites different then static pages. In the worst case, the index gets caught in an endless loop. To avoid that, some spiders crawl pages without any parameters.

The CMS should provide self speaking URLs and use methods to hide parameters to get a better position in the search result. Instead of “index.php?id=125” the URL may look like “index.php/id=125/Some_new_-stuff_on_my_website”. Some systems even can hide themselves completely, although this requires usually specific settings on the Web server. Another trick is to present a search engine optimized version of the Web site if the client is a crawler.

ISP mass hosting support

Parts of the content management system are needed only once on the server and are common for all Web sites. Providing Web space with an included CMS can increase revenue for ISPs, especially when they offer additional services like customizing and support.

[33] http://www.w3.org/WAI/
[34] http://wave.webaim.org/index.jsp
[35] http://www.w3.org/TR/WCAG20/

2.5 Web Content management systems

In this chapter I take a look at various systems to get an idea of how it looks in the real world of web content management. Various Web sites[36] [37] [38] [39] list more then 300 different systems, so the choice was not an easy task, which systems to examine here.

On behalf of the Austrian ministry for education, 285 Content Management Systems where evaluated [BHMH03, p. 17] for their possible use in schools and universities. The authors made a recommendation for 3 open source products: PHP-Nuke (and derivatives), TYPO3 and Eduplone.
Others ([Ped04, p. 8], [Mic04, p. 29]) applied different criterias and took the following systems for a closer look: Drupal, eZ Publish, Mambo, PHP-Nuke (and derivatives), Xaraya and Xoops.

At the CMS Matrix Web site[40], a place dedicated to provide information on both commercial as well as open source content management systems, users can rate products in various categories like system requirements, ease of use, flexibility or performance. The well known names usually show up in the rankings at the top places.

My personal experience comes from Small and Medium Enterprises (SME)[41]. In section 2.6 on page 32 the criteria for my own system are described.

I applied these criteria and the information collected from the above mentioned sources as a guideline to select the CMS for this paper. I have chosen the following systems:

- TYPO3
- Drupal
- Joomla!
- eZ publish

These system have been selected, because they are widely used and well documented, and because they have a strong developer community. These systems are open source, which makes it easier to try them and examine, how they work.

If you are interested in proprietary products, a brief description of three major content management systems can be found in [Wer05, p. 9]. This article covers ECM Suite RedDot XCMS 6.5[42], Vignette V7[43] and Interwovens TeamSite 6 Content Management Server[44].

All my tests where done on a PC running the German version of Microsoft Windows XP SP2 with all available updates installed. I used Apache 2.2.2 as Web server with mod_rewrite and mod_ssl, .htaccess support enabled and PHP loaded with php4apache2 server API module. Typo3, Drupal and Joomla where tested with PHP version 4.3.7, for eZ publish I had to switch to version 4.4.1. The following PHP extensions were active: php_gd2, php_xslt and php_mbstring. As DBMS I used MySQL version 4.1.14, as management frontend phpMyAdmin 2.7.0-pl2 has been installed. Both the Microsoft Internet Explorer 6 and Mozilla Firefox 1.5.0.4 have been utilised. To check the HTML source for compatibility, the Web developer plugin[45] for Mozilla Firefox has been used.

2.5.1 Feature comparison

Table 1 on the next page has been created using results of my own experiments as well as data of CMS Matrix[46]. An extensive feature comparison of the systems mentioned here and some other interesting

[36] http://www.cmswatch.com/Reports/Vendors/
[37] http://www.cmsmatrix.org/matrix/cms-matrix
[38] http://www.contentmanager.de/itguide/marktuebersicht.html
[39] http://www.opensourcecms.com/
[40] http://www.cmsmatrix.org/matrix/cms-matrix?func=viewRatingDetails
[41] The European Union defines enterprises with less then 50 employees as small and companies up to 250 employees as medium sized. http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/index_en.htm
[42] http://www.reddotsolutions.com
[43] http://www.vignette.com
[44] http://www.interwoven.com
[45] http://chrispederick.com/work/webdeveloper
[46] http://www.cmsmatrix.org

products can be found in the appendix (section A.5 on page 137).

Table 1: *CMS comparison*

	TYPO3!	**Drupal**	**Joomla!**	**eZ publish**
Version	4.0	4.7.2	1.0.10	3.8.1
Operating system	Any	Any	Any	Any
Data repository	MySQL, PostgreSQL, Oracle, MSSQL	MySQL, PostgreSQL	MySQL	MySQL, PostgreSQL, Oracle, MSSQL
Installation	simple	medium	simple	simple
Versioning	Yes	Limited	Yes	Yes
Workflow	Limited	Limited	Yes	No
Integration				
Templating	yes	yes	yes	yes
Asset management	yes	yes	yes	yes
Metadata	yes	yes	yes	yes
Staging	free add-on	no	no	yes
User management	advanced	medium	limited	medium
User interface				
Editing interface	advanced	medium	medium	advanced
Managing the site structure	yes	yes	yes	yes
Session analysis and reporting	free add-on	yes	yes	free add-on
Applications				
Software architecture	mainly object oriented, but no obvious architectural pattern	procedural, hook system	object oriented and procedural, controller / view	object oriented and procedural
Common API	yes	yes	yes	yes
Integration of external tools	free add-on	yes	free add-on	yes
Localisation	yes	yes	free add-on	yes
Programmability	easy	medium	easy	easy
Scheduling	yes	free add-on	yes	yes
Searching and indexing	free add-on	yes	yes	yes
Syndication	yes	yes	yes	yes
Personalisation	free add-on	free add-on	yes	yes
Import/Export)	xml, xml, csv, swx	csv, xml	csv, xml	csv, xml
Deployment	free add-on for pdf, rtf, csv, xml	free add-on for csv, xml	pdf, free add-on for csv, xml	pdf, csv, xml, open document
Web / WAI compatibility	yes / free add-on	yes / limited	no / no	yes / yes
Caching	yes	yes	yes	yes
Search engine optimization	yes	yes	yes	yes
ISP mass hosting support	yes	yes	free add-on	yes

2.5.2 TYPO3

"TYPO3 is an enterprise-level open source content management system released under the GPL. It runs on more than 122,000 servers worldwide. The application has been translated into 43 languages and is actively being developed in a community of over 27,000 users in 60 countries. Some of its users include BASF, DaimlerChrysler, EDS, Konika-Minolta, Volkswagen, UNESCO, as well as numerous universities, government agencies and non-profit organisations"[47]. At the TYPO3 Web site more of references are

[47] `http://www.typo3.com/TYPO3_Version_4_0.1852.0.html`

listed[48].

The concept behind of TYPO3 requires a considerably amount of time to become familiar with it and has a flat learning curve. Also a big chunk of knowhow about Internet technology is needed to build a Web site with TYPO3.

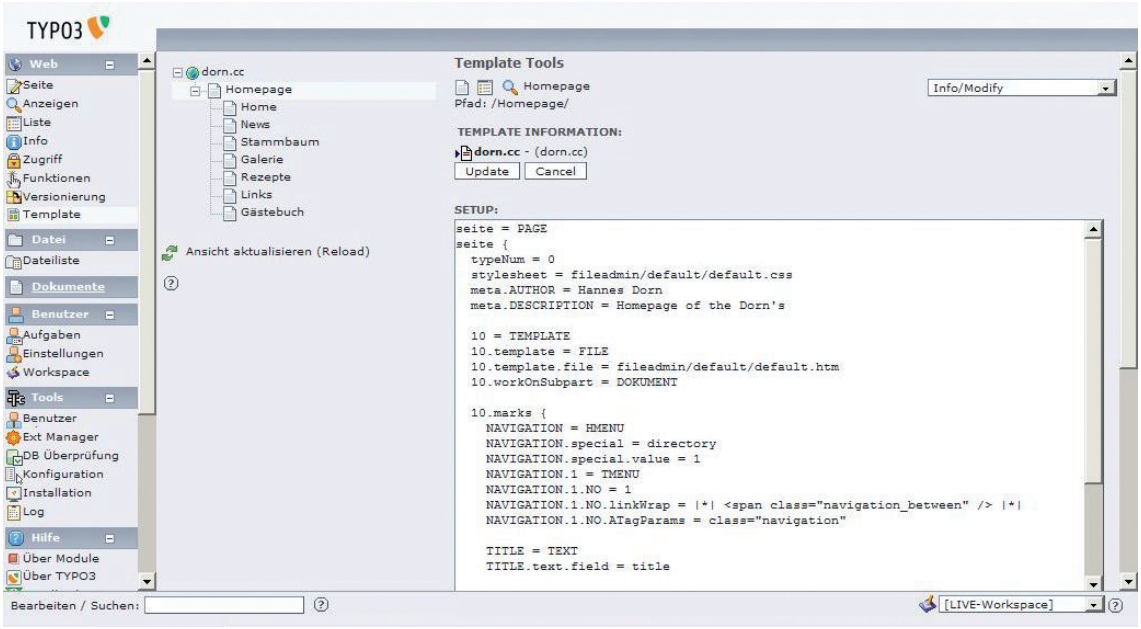


Figure 8: *TYPO3 backend*

Web site: `http://www.typo3.com/`
Version evaluated: 4.0
License: GPL

System requirements

TYPO3 can be installed at nearly every Internet service provider and requires a Web server like Apache 1.3.x, PHP 4.1.2 or above, MySQL 4.x and ImageMagick. Shell access to the Web server is not required. A PHP accelerator like eAccelerator or Zend engine is highly recommended.

Strengths and weaknesses

This is not a complete analysis. I just mention topics which I found interesting when exploring TYPO3. It may seam, that I am bringing more weaknesses and less strengths. To be clear, I think that TYPO3 is a great system and therefore used by software developers for good reasons.

Installation

+ Although TYPO3 is a rather complex program, installation is quite easy with the 1-2-3 setup program. Configuration settings for the database access are asked and a database can be selected or created. All tables are created by the setup script. In the standard setup paramters for sending mail or image manipulation can be set. It checks PHP parameters, the Web server environment and the filesystem, that all necessary directories exist and if they are writeable. Tests for image processing exists as well as for the database tables.

+ The database schema can be created, checked and updated within the installation program. There is no need to execute SQL statements with a database management program like PhpMyAdmin.

- As noted above, installation is easy, but in the standard setup a lot of parameters can be changed. If a directory does not exist, no suggestions are made, what needs to be done. Do I have to create the directories

[48] `http://typo3.org/about/sites-made-with-typo3`

by myself or are they created automatically, when they are needed later? A lot of information is presented, but most of it can confuse unexperienced users.

Documentation

+ Video tutorials are available by Video2Brain[49].

- The documentation "TYPO3 Inline User Manual", which is installed with TYPO3 tells what can be done, but not *how* it can be done. Tutorials about how to create a Web site exist[50] and should be integrated.

Integration

- Typo3 can be hardly integrated with an existing Web site.

User Interface

+ Beside of editing the content in the backend, it is as well possible, to switch the frontside into an editing mode. Near the content elements, a link is displayed, which opens a window, where the specific content can be changed without the need to go to the administration interface.

+ Different languages for the user interface can be installed easily with the extension manager.

+ The TemplaVoila extension provides a very innovative approach to combine TYPO3 with pre-created HTML layouts. The design is displayed in a WYSIWYG environment and the Webmaster can assign the position of various page elements like navigation controls, header images, banners or the content areas with a few mouse clicks.

- The user interface and the complexity of TYPO3 requires extensive training for both the Webmaster and the authors. TYPO3 is by far not selfexplaining.

- The backend of TYPO3 looks a bit old-fashioned. Other systems like Joomla! have a more modern design.

Templates

+ One of TYPO3's major strength is its template language. TYPO3 distinguishes between design templates (which are more or less HTML) and templates using TypoScript. TypoScript is a declarative programming language (and could be lumped together with Prolog, XSLT or SQL) and is used to declare how TYPO3 has to create the output.

+ Images for navigation or banners can be created by TypoScript. This is very useful and can save a lot of work.

- TYPO3's strength is at the same time one of the biggest weaknesses. TypoScript is though not hard to understand, but needs to be learned to make use of the CMS.

- TypoScript can grow fast into a long list of statements, which create the HTML code. I think, this makes it unnecessary difficult to create templates. oPage tries an approach, where users with HTML knowhow can create templates without the need of learning another language.

- The mixture of TypoScript and HTML code in the templates makes it sometimes hard to correct bugs.

Web compatibility

+ The HTML code of the CMS is pretty good and has no errors or major warnings.

- TYPO3 can not guarantee, that a page is error free and compatible with Web standards. The quality of the resulting HTML code lays in the hand of the designer, who is creating the templates with TypoScript.

[49] http://www.video2brain.com/

[50] http://typo3.org/documentation/document-library/tutorials/

Application

+ Many add-ons for TYPO3 exist. Installation is part of the CMS and is very simple. The "Extension Repository Kickstarter" helps to create the basic framework for a new extension.

- I did not find a high end online shop made with TYPO3. The eCommerce section of TYPO3's reference list only rather simple approaches, (compared to e.g. `www.demmer.at` or {www.amazon.com).

Performance

+ TYPO3 depends heavily on caching. With caching it is very fast.

2.5.3 Drupal

Drupal was originally a tool for building community sites which makes it a good choice for blogging or news sites, though it is also capable for building standard websites or ecommerce applications[51]. Drupal has a core, which can be extended by plugins (modules) that provide additional features. For example Drupal's taxonomy extension allows any content to be classified with a flexible tagging system. Modules interact with the core system via hooks. Drupal's core provides protection against many typical security threats, like SQL injections[52]. A prominent Web site using Drupal is "Spread Firefox"[53].

Currently only estimates about the quantity of Web sites using Drupal are available. The number is at least higher then 50.000[54].

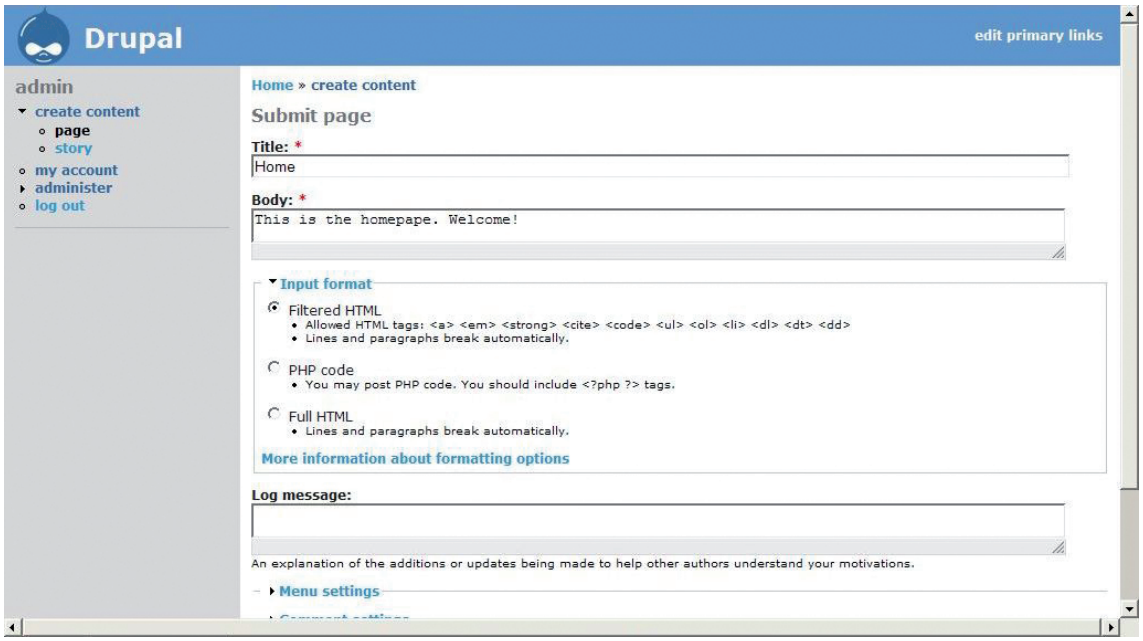


Figure 9: *Drupal backend*

Web site: `http://www.drupal.org`
Version evaluated: 4.7.2
License: GPL

System requirements

Web server with PHP4 (4.3.3 or greater) or PHP5, either MySQL or PostgreSQL. Apache web server and MySQL database are recommended by the developers. The PHP-extension XML is needed for XML-based

[51] `http://www.openadvantage.org/articles/oadocument.2005-04-20.1805785131`
[52] `http://en.wikipedia.org/wiki/Drupal`
[53] `http://www.spreadfirefox.com`
[54] `http://drupal.org/node/36748`

services like RSS. Support for clean URLs (self speaking URLs) requires the Apache module mod_rewrite and support for .htaccess.

Strengths and Weaknesses

Drupal shows its strengths on sites with user interaction. Although Drupal is very fresh and provides a clean user interface with lots of cool stuff (see below), the system itself is written with procedural PHP code. Drupal does not have a scripting language like TYPO3, but provides quicker results when starting with a site. I found it hard to rebuild my own Web site with Drupal. It is easier to broaden the features on your Web site then to adjust Drupal to your needs. If you need what Drupal offers, it does a great job.

Installation

+ The installation of the core files is rather easy: download an archive from the homepage, extract the files and upload them to the Web server.

- The Drupal archive is compressed with tar and gzip, tools which are unfamiliar to Microsoft Windows users. An extra version compressed with Zip should be made available.

- The database schema for Drupal is provided in a SQL file, which has to be executed with the database management program (or PhpMyAdmin). Other CMS create the database and all tables by themselves.

- Database connection settings and other configuration entries have to be set in a PHP file.

- On my local system, processing of .htaccess was limited in the Apache Web server configuration file. To a new user, it may not be obvious what to do when "Internal Server Error" is displayed. At least "AllowOverride FileInfo Indexes Limit Options" is needed in httpd.conf.

- Drupal requires at least PHP 4.3.3 (released 25th August 2003), some servers (e.g. Red Hat 7.3) are still running older versions of PHP (and can not be easily upgraded), which are then not compatible with Drupal.

Documentation

+ The user interface is pretty much self explaining. The Drupal Web site overs comprehensive documentation and a number of videocasts.

- Online help is available, but only partly translated into German.

User Interface

+ Drupals user interface looks very fresh and provides new useful control items: Text fields can be enlarged with the mouse, file uploads can be done without a post back of the whole form, suggestions for content node tags are made while typing them. Content management forms are divided into parts, which can be folded and expanded.

- There is no kind of separated backend for editing the Web site. The administration interface is integrated into the Web site itself. I think, this concept makes troubles when creating bigger Web sites.

Integration

- Drupal can not be integrated with an existing Web site.

Templates

+ The position of various content blocks can be defined within the CMS.

+ Complete design themes can be downloaded from Drupals Web site.

+ Alternative template engines can be installed.

- Many sites using Drupal look similar to PHPNuke sites: In the center the content is displayed and the navigation, the number of current users online and other information is placed in boxes around it.

- Creating a new theme requires knowledge of HTML, CSS, the template engine, PHP programming and the Drupal hook system.

Web compatibility

+ The code of Drupal is completely written in XHTML and has no errors or major warnings.

+ Depending on the design theme used, Drupal is compliant with Section 508 and WCAG Priority 1, 2, 3 rules[55].

Application

+ Extensions (here called modules) can be installed with the CMS by just a few clicks. No manual file or database manipulation needed.

+ Many extensions are installed with the core installation of Drupal and can be activated with just a click. From the `drupal.org` Web site, additional extension can be downloaded.

+ The "module builder" add-on creates a skeleton as basis for module development.

+ For ecommerce, a free add-on module is available providing a wide range of features like sub-products, creditcard and PayPal integration, digital distribution and fulfillment centers.

- Because of the large number of modules, the module installation list can be confusing. Modules should be grouped into categories like ecommerce or community.

- The extensions from the Drupal Web site have to be installed by hand. Other CMS provide a more convenient way to include new modules.

Performance

+ PHP offers an extension called mysqli (improved) which is a new interface to access MySQL. This data access library allows to prepare statements which improves performance, if this statements are repeatedly executed. Drupal optionally supports the use of mysqli.

+ Caching can be activated when needed. At Web sites with a high number of requests, a minimum cache time can be set.

2.5.4 Joomla!

Joomla! is a free open source content management system. It includes features such as international language support, page caching, search engine indexing, printable versions of pages in HTML and PDF and extensions for blogs, forums, polls, calendars.

Joomla! is a fork[56] of Mambo Open Source. The name is a phonetic spelling of "jumla", which is the Swaheli word for "all together". There is no big difference between the latest stable versions of Joomla! and Mambo. I analyse Joomla!, because the development team as well as a lot of the community sites left Mambo and switched to Joomla!.

Joomla! can be customized and extended with templates, components, modules and plugins. A template is a collection of files and is used for the layout of the Web site. Components are applications within Joomla!, like a reservation system. Modules are used to display certain information on the Web site, e.g. "who is online" (in some other CMS these are referred as blocks). Plugins are extensions of Joomla! mainly for the backend. For example the WYSIWYG editor TinyMCE is integrated as a plugin.

Today, no official usage numbers exists. After a post in the Joomla! forum, I got the following answer[57]:
1. until the Joomla! name was announced about 10 months ago there were zero hits on Google and today

[55] Web Content Accessibility Guidelines `http://www.w3.org/TR/WAI-WEBCONTENT/`
[56] After a disagreement in the developing of Mambo, a group of developers took the source code and started working on Joomla!
[57] `http://forum.joomla.org/index.php/topic,72983.0.html`

there are 37,900,000
2. Google reports 73,900 hits for the string “Joomla! is Free Software released under the GNU/GPL License.” which is in the footer of many sites.
3. Google reports 35,000 hits for the string “Joomla! - the dynamic portal engine and content management system” found in the metatag of many sites.
4. forge.joomla.org reports approx 1,115,000 downloads of Joomla!.

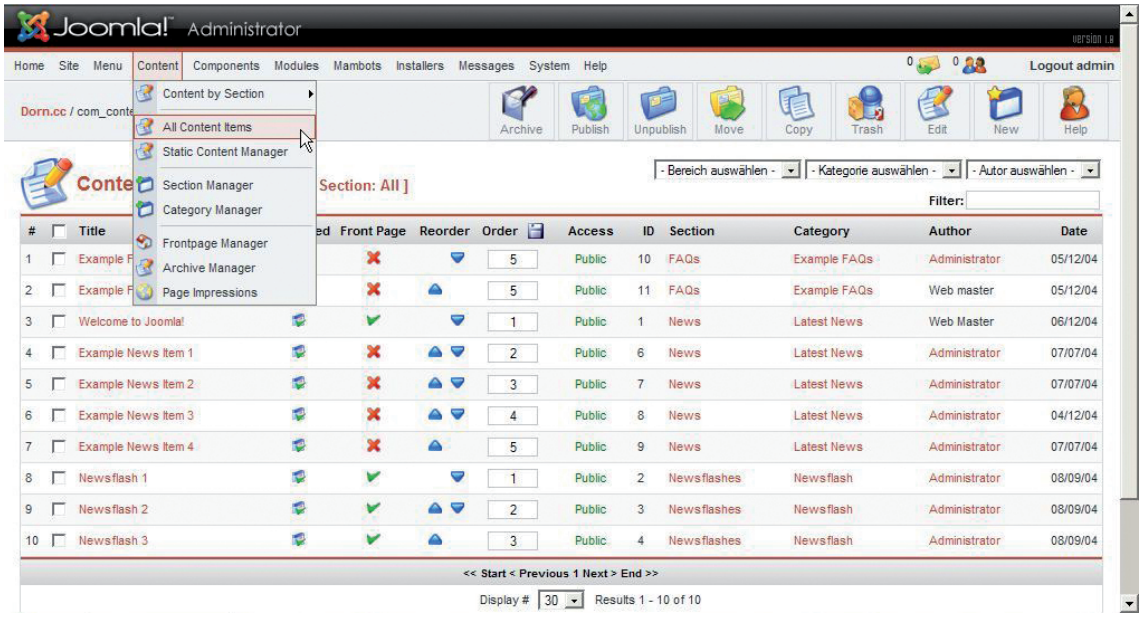


Figure 10: *Joomla! backend*

Web site: `http://www.joomla.org`
Version evaluated: 1.0.10
License: GPL

System requirements

Joomla! requires at least Apache 1.13.19, MySQL 3.23.x and PHP 4.2.x (with MySQL, XML and Zlib support). Joomla! supports all major browsers.

Strengths and weaknesses

I am evaluating Joomla! 1.0.10, which is the latest stable version available and make additional comments on the current development version (Joomla! 1.5 pre-beta). Joomla!’s strenght are the community functions (and is somehow similar to Drupal).

Installation

+ All prerequisites for running Joomla! are checked in the installer program. This includes PHP version, extensions and settings as well as checks for directories and access rights in the filesystem.

+ Sample data is included in the installer and can be imported if needed. This is useful to get an instant success.

- A database must already exist. The Joomla! installation program does not create a new database, this has to be done by hand with some other tool.

- Joomla! does not display a page, until the installation files are completely removed from the server. To avoid to remove the installation program, it should check, if the installation is already configured and request the admin password. Improvements for this issue will come with version 1.5.

Documentation

+ The Web site offers manuals about the installation of Joomla! Additional information for administrators is provided as well as a PDF file for authors.

+ Tool tips with context sensitive help are displayed near the input fields.

+ Local help files are available. If needed, the CMS can be configured to display more recent files from the Joomla! help server.

User management

- Joomla! supports only a couple of pre-defined user groups. These groups can not be changed.

User Interface

+ Joomla! looks nice and cuddly. This can increase the acceptance of Joomla! by the authors.

+ Deleted items are moved to the trash folder and can be restored if they where accidently removed.

+ The next release of Joomla! will make use of AJAX to enrich the user interface.

- The backend is only available in English. Translations of the backend are planned for Joomla! 1.5.

- The main menu is disabled while a record is edited. To switch to another section, the current record has to be saved (or canceled) first.

- Although the user interface seams to be pretty much self explaining, a tutorial how to create a website should be added to the local help files.

Integration

- Joomla! can not be integrated with an existing Web site.

Templates

+ Both the frontside as well as the backend design can be changed with template packages.

+ Professionally designed templates are available both commercially and free of charge.

+ Themes can be applied to the whole site or just to a part of it.

- The templates are a mixture of HTML and PHP. In the next major release of Joomla!, a new template engine is used, where the PHP code in the template is replaced by XHTML compatible tags.

- There is no strict separation of program and HTML code: Templates, components and modules contain HTML fragments.

- The Web page is composed of “modules” (blocks, which display some data) like menus, polls or “who is online”. The layout of the content produced by modules can only be changed in the program code.

Web compatibility

+ The HTML code generated by the default installation of Joomla! contains no errors or warnings.

- Currently Joomla! does not comply with many WCAG/508 requirements[58].

- HTML code quality depends on the template and extensions used.

Application

+ The extensions directory on the Joomla! Web site lists about 700 items which are very well organized into categories.

[58] http://help.joomla.org/content/view/805/60/

- Extensions have to be downloaded prior to installation. Other systems can download and install extensions directly.

Performance

+ If available, Joomla! can take advantage of prepared statements with the new iMySQL interface.

+ To increase the performance of the Web site, the administrator can activate content caching in Joomla!

- Caching can not be configured for selected content items, only for the whole site.

2.5.5 eZ publish

eZ publish is a framework for content management systems. Although it is open source, it is still developed, distributed, and supported by eZ systems, a company providing services around eZ publish.

Two different licensing models are offered by eZ systems: The CMS can be used under the general public license (GPL)[59] and a proprietary license (eZPUL)[60]. Both allow the licensee to modify the source code, but the GPL requires to release the changes to the public, while this is not necessary with the proprietary license. For both licences, a 12 month bug fix guarantee can be bought.

Standard editing features include version control with roll back option, a trashcan, time based publishing, translation management, preview of pages, automatic image conversion, search engine, self speaking URLs and dead link checking. The content is organized in a hierarchical tree structure and user permissions define where it is possible to edit content.

eZ publish has been downloaded more than a million times. eZ systems has more than 1500 international customers.

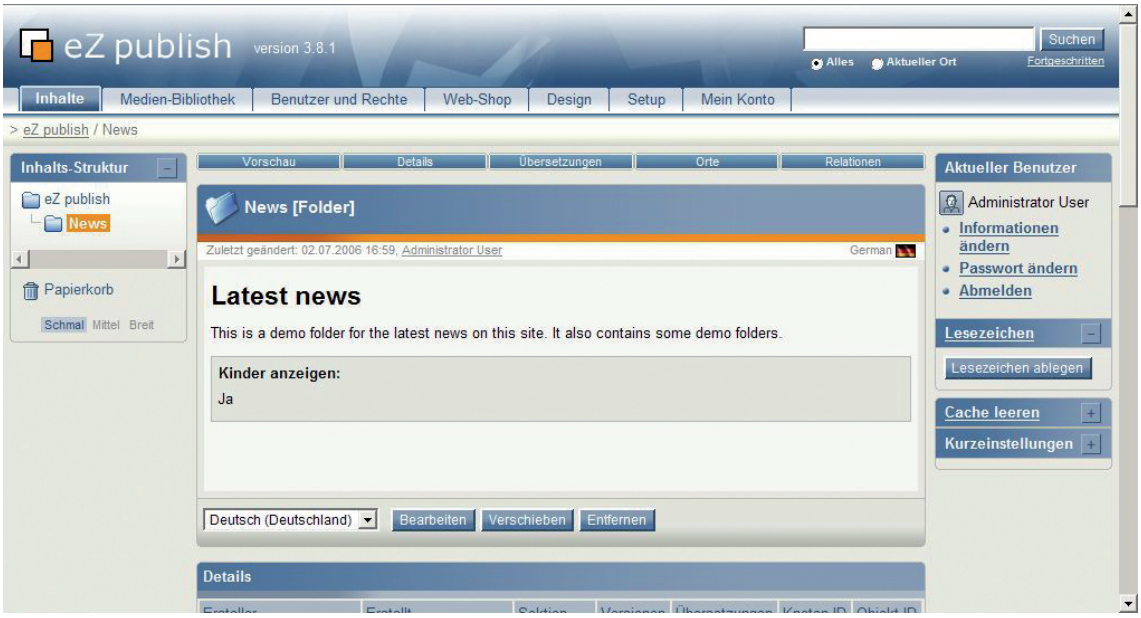


Figure 11: *eZ publish backend*

Web site: `http://ez.no`
Version evaluated: 3.8.1
License: GPL

[59] `http://www.gnu.org/copyleft/gpl.html`
[60] `http://ez.no/products/licenses/ezpul_license`

System requirements

eZ publish requires an Apache Web server version 1.3.x or 2.x, PHP version 4.4.x or above, MySQL version 3.23 or later and ImageMagick or the GD library of PHP for image conversion. Instead of Apache, any other Web server with PHP support can be used. Alternatively to MySQL the Postgre DBMS version 7.3 can be installed. For clean URLs, the Apache module mod_rewrite and .htaccess support is required.

The documentation notes, that Apache 2 has to be run in forking[61] mode, because PHP is not thread safe. This may be outdated, as the current PHP version 4.4.1 has thread safety enabled.

Against the installation instructions I did not change the PHP memory limit from 16 MB to 64 MB. All other tested systems had no problems with a limit of 16 MB and so it seams to be here.

At my first tests, I used the latest PHP version 4.4.2 on Windows, which has a serious bug and eZ publish installation crashed the Apache Web server. I had to step back to version 4.4.1.

Strengths and weaknesses

Although eZ publish is available under the GPL, a proprietary license and guarantees can be bought, which is a valuable option for professional users. The biggest drawback of eZ publish is the lack of performance. As this is comprehensible because of the XML storage, and may not be a problem if the Web site is hosted on an exclusive server, but it may be a problem when eZ publish is used on a shared Web space. The use of an PHP-accelerator is highly recommended.

Installation

+ Four different installation procedures are described in the documentation. This includes manual installation as well as a bundled installation, where eZ published gets installed along with the Web server, PHP and the DBMS.

+ Path to ImageMagick is automatically detected.

+ The installation program includes eight site packages like corporate-, forum- or shop-site.

- eZ publish requires a rather recent PHP version which may not be available on every Web server.

- As noted in the documentation, PHP 5 is not supported. I did not try to use eZ Publish with PHP 5.

- The database has to be created manually.

- Unicode support for the database has to be activated.

- Only one of the site packages can be installed. It would be nice, if one could choose to install the corporate Web site package together with the shop and the forum packages.

Documentation

+ The Web site offers comprehensive documentation covering all areas of eZ publish. Video tutorials describing how to install and use the CMS can be viewed.

- The videos on the Web site have only textual explanations, which makes it a bit boring to follow the tutorial. Audio comments, as seen at the Drupal Web site, could improve the learning effect.

- In reference to the notes I made in the section system requirements of eZ publish, I do not have the impression, that the documentation is up to date.

- While the documentation about the end user part of eZ publish is substantial, information about the core system lacks of details, especially at the system overview, the kernel and the libraries.

[61] Apache 1 worked in a similar way: for every request a new instance of the Web server is started. In Apache 2 an extension has to be installed for that.

Data repository

While other CMS store the content as plain text or HTML code, eZ publish saves everything in XML format.

+ New content classes with user defined fields can be created in the backend of the CMS.

+ An application programming interface (API) can be used to access the content repository to import or extract data.

- Working directly with the database is not possible since the content objects are distributed across multiple tables.

User Interface

+ The WYSIWYG editor produces XML code instead of HTML code.

+ The design and layout of the backend can be customized the same way as for the frontend.

- The backend user interface looks a bit technical and cold.

Integration

+ OpenOffice documents can be imported into eZ publish using WEBDAV.

+ With WEBDAV it is e.g. possible to mass-upload images and other files using drag and drop of the Windows Explorer.

+ The program, which creates the document index for the search engine can be configured to include Microsoft Word and PDF files.

- eZ publish can not be integrated with an existing Web site.

Templates

+ The CMS completely separates Logic, Content and Layout (LCL).

+ Each content class has a default template which can be overridden by a user defined template.

+ Templates for creating PDF files can be defined as easy as templates for HTML output.

Web compatibility

+ Web pages produced by eZ publish use XHMTL and CSS and do not contain any errors.

+ The look and feel of the Web site can be customized by changing the CSS files. A video tutorial demonstrates this procedure using Makromedia Dreamweaver.

- If displayed with Mozilla Firefox the pages look slightly different as when they were displayed with Microsoft Internet Explorer.

Application

+ Since content objects are stored in XML format, they can be used to produce other formats than HTML, e.g. PDF.

+ The CMS includes a Web shop, a forum and a gallery.

+ On the Web site of eZ publish a broad range of extensions are listed.

- There is no wizard which helps to create an extension.

- Extensions have to be installed manually. No user interface exists in the backend for that task.

Performance

- Compared to the other systems tested, eZ publish produces a much higher CPU load and is noticeably slower. A database trace shows, that about 100 queries are executed to display the start page, while the other systems need about 30 queries.

+ A cluster setup with several servers can be used for high traffic Web sites.

+ Depending on the Web site eZ publish can be tuned for higher performance by optimizing page layout and cache parameters.

2.6 Goals for oPage

Operating system

oPage must run with minimal system pre-requirements (hard- and software), to allow a widespread use. It has to be platform independent and must run without any modification at least on Linux and Microsoft Windows. The framework and the Web sites created with it have to be compatible with all Web servers (here implemented for Apache and Microsoft Internet Information Server) which supports PHP (running either as Web server module or as CGI[62] program). A typical Web space of an average internet service provider has to be sufficient for hosting the Web sites.

Data repository

Web sites have to be independent of the database management system (DBMS) and queries have to be abstracted with a separate layer. At least MySQL has to be supported, since this DBMS is usually offered by typical internet service providers. It must be possible to share a database with other installations of oPage as well as other programs. This is important when hosting multiple Web sites in one Web space.

Installation

Installation of the CMS should be as easy as possible. In the best way, the user uploads a bunch of files and starts a setup script, which asks a few questions, checks prerequisites, installs everything and pops up with a running Web site.

Web compatibility

HTML pages created by oPage have to follow standards like XHTML and have to be compatible with all major Web browsers like Mozilla Firefox, Microsoft Internet Explorer, Opera, Safari both on Linux, Microsoft Windows and Apple Computer operating systems. Content management functions must not rely on browser or operating system specific features. These features should be made available optionally.

Search engine optimization

Where possible, self speaking path names should be used and ways to improve search engine compatibility may be provided.

User interface

The user interface should be self explaining. Authors should not need special training and knowledge of HTML must not be assumed. For the Web site, the content and the editing interface support for multiple languages have to be provided. A way to enter hyperlinks without HTML coding has to be defined and a WYSIWYG editor should be integrated for creating prelayouted content and editing static files. The user interface has to be robust against faulty input and support content managers in keeping a consistent layout and design on the Web site.

User management

oPage has to provide a flexible method to define access rights.

[62] common gateway interface

Security

Data entered by regular users must be processed securely, especially when used in SQL queries.

Software architecture

oPage has to be developed using a modular and object oriented software design. This includes the usage of abstract base classes, the Model View Controller (MVC) concept, the factory principle and other design patterns. A main goal is Logic Content Layout (LCL) separation through the use of templates and content classes. Precautions have to be taken to support shared hosting of multiple Web sites on one Web space. All platform specifics have to be hidden in an abstraction layer. Also it has to be kept in mind, that the development platform may be different from the hosting system (development takes place on Microsoft Windows, hosting is done on a Linux system). It has to be possible to share core files among multiple projects. Customizations have to applied in separate areas to allow easy updates of future versions of oPage.

Application programming interface

oPage should be easy to use and support developers to get instant results. Web designers should be able to create Web sites and customize the CMS without the need of programming skills. Software developers should be able to extend the framework by reusing existing or creating new modules.

2.7 Summary

In this chapter web content management systems have been described. First the basic principles of content management have been shown: authoring, storing, publishing and workflow. Each term in WCMS has been explained and the different user roles involved in a Web project were outlined.

Four different open source content management systems have been described and my opinion on their strengths and weaknesses have been listed.

TYPO3's major strength is its template language. TYPO3 distinguishes between design templates (which are more or less HTML) and templates using TypoScript. TypoScript is a declarative programming language and defines how TYPO3 has to create the output. To create a Web site with TYPO3 the Webmaster first has to learn TypoScript. The backend of TYPO3 is not very user friendly and looks a bit oldfashioned and may scare authors at the first look.

Drupal's origin as a tool for building community sites makes it a good choice for a blogging or news site. Drupal's taxonomy extension allows any content to be classified with a flexible tagging system. Although Drupal is very fresh and provides a clean user interface, the system itself is written in procedural PHP code. Drupal does not have a scripting language like TYPO3, but provides quicker results when starting with a site. It is easier to broaden the features on your Web site then to adjust Drupal to your needs. If you need what Drupal offers, it does a great job.

Joomla! includes features such as international language support, page caching, search engine indexing, printable versions of pages in HTML and PDF and extensions for blogs, forums, polls, calendars. The backend of Joomla! looks very well designed and will be even nicer in the next release. Similar to Drupal, Joomla! has its strenghts in the community features.

eZ publish is an open source CMS and powered by a commercial company. In contrast to the other systems showed here, eZ Publish uses XML to store content in the data repository. Somehow, this CMS is way slower than the other systems. Still, tuning tips can be found, but I would recommend a dedicated server. All in all, eZ publish is a powerful general purpose content management system.

In the last chapter of this section, the design goals for oPage in the fields of system requirements, installation, compatibility, user interface and management, software architecture and programming interface were specified.

3 oPage Framework

This chapter provides a technical description of the oPage framework.

3.1 Structure

Each Web site consists of the oPage core system (which provides the framework), the administration part (also called the backend) and the frontend. The frontend and the backend are just two different Web sites using the same framework (see figure 12). The framework and the administration system are not supposed to be modified by the Webmaster and can thus be shared between different projects. There is no user interface provided by the framework directly. The administration Web site is pretty much standardized while the frontend Web site is highly customizable.

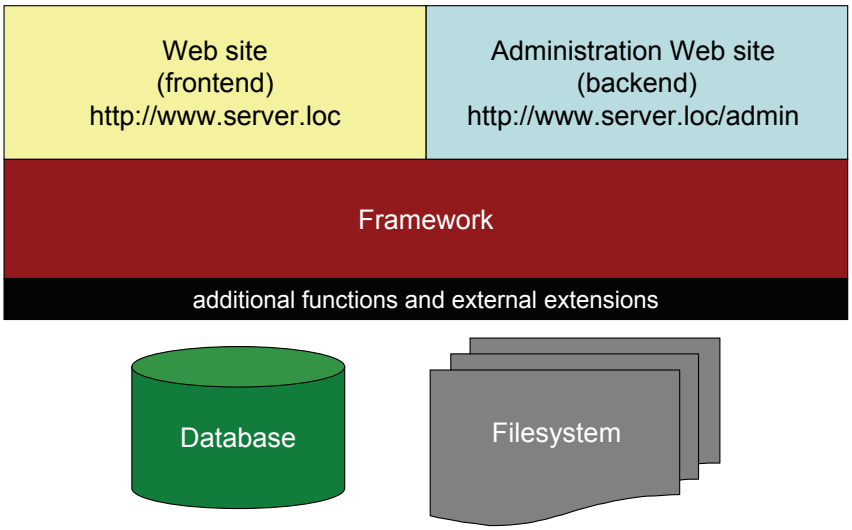


Figure 12: *oPage structure*

The framework is located in the directory `include/` (see table 6 on page 132 for details) and the administration system in `admin/` (see table 7 on page 132). The Web site files can be stored wherever the Webmaster wants to (see table 8 on page 133).

Please note, that the color schema of figure 12 is used in all following illustrations.

The content is divided into structured pieces and managed by so called modules. From the storage point of view, a module represents a database table and has properties like the name of the table and field-names and -types. Through a base class, all modules provide operations to create (insert), retrieve (query), update and delete rows (also referred as CRUD[63] operations). In object oriented speaking, a module represents a list of entries and provides operations to manage these entries.

Figure 13 on the following page shows a screenshot of `http://www.opage.at`. Everything around the content area is common to all pages and is defined in a base template.

(1) In the top right corner, links to the contact form and imprint of the Web site are displayed (menu).

(2) On the left side, the navigation control shows links to other sections of the Web site (menu).

(3) Also displayed on the left side are two forms, where uses can subscribe the newsletter or search the Web site (form).

(4) The title of the current page *Module* is displayed above the content area (page title).

[63] CRUD is described at `http://en.wikipedia.org/wiki/CRUD_(acronym)`

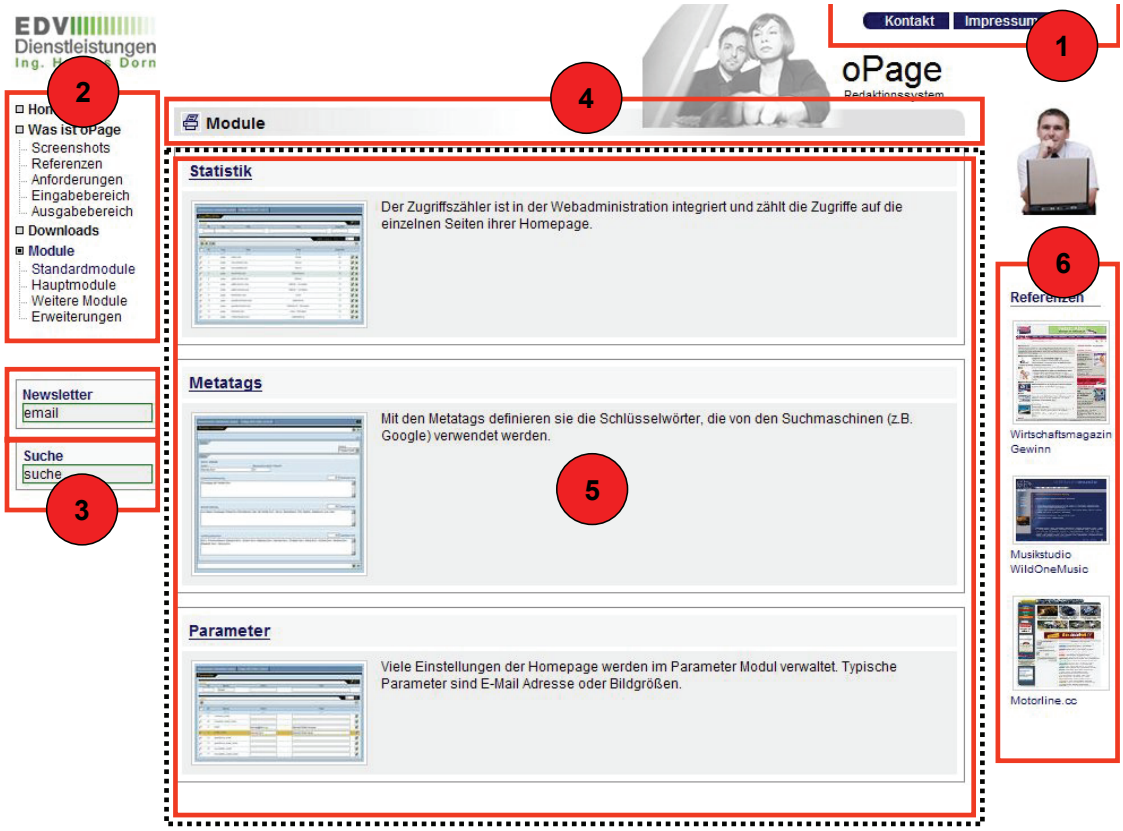


Figure 13: *sample page screenshot*

(5) This is the content area. On this page its filled with a list of standard modules of oPage (module).

(6) On the right side, three randomly picked references are displayed, which change every time the page is created (module).

Figure 14 on the next page presents a brief overview how a page is created with oPage. The script `index.php` creates an instance of a module, which uses a data access layer to read records from the associated module table. The content is merged by the template engine with a layout file to create the final output, which is sent to the browser. Structured content is always stored in a database while unstructured and binary content stays in the file system. The templates which are used to generate the Web page are also stored as files.

oPage offers logic, content and layout separation (compare figure 15 on the facing page). There are no HTML commands embedded into the program. And of course there are no PHP commands in the templates.

3.2 Website

To get an idea, how oPage is used to output a web page, the source code of a simple page is provided and discussed here.

This paragraph describes listing 1 on the next page: The code between `<?php` and `?>` is executed by the PHP interpreter. All statements outside the PHP tags would be treated as HTML code. In oPage, PHP files contain only program code, the layout information is stored in templates.

First the oPage main file `include/opage.php` is included, which initializes the framework. Then an instance of CWebpageX is created with the `Webpage()` method of the global factory. In this example, the values for the page `home` are queried from the database. A template file for the layout of Web site is loaded automatically by the page object.

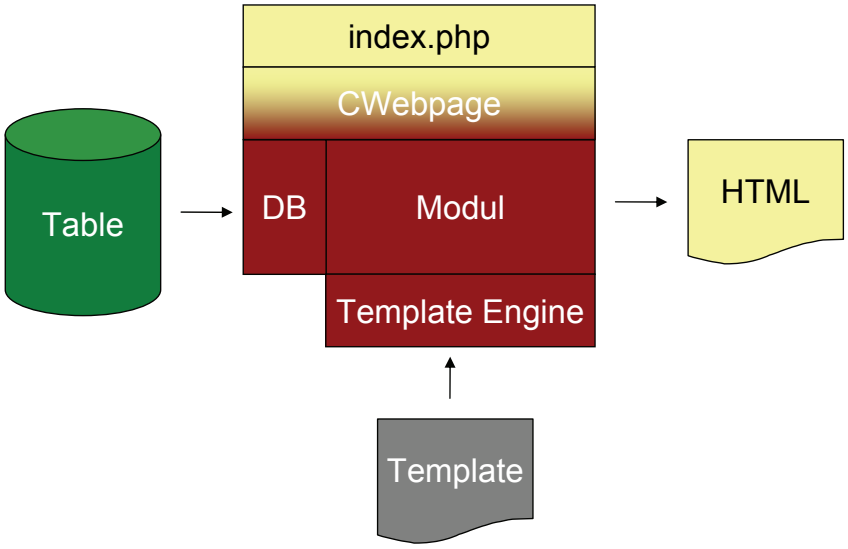


Figure 14: *oPage dataflow*

Logic (Presenter)	Content (Model)	Layout (View)
index.php	Database Filesystem	Template CSS
CWebpage CModul, CForm CAdmin	CPage CContent CLinks	CTemplate

Figure 15: *oPage architecture*

To demonstrate the use of a module, an object for metatags is created. Then a filter is set to query only records where the field `name` is equal to `default`. `QueryData()` creates a SQL statement by combining field names, table name, filter criterias and order directives, queries the database and stores the result within the object, from where it is merged into the template of the page object.

For each language used on the Web site, an extension is defined in `custom/config.php`. For the default language, the extension is an empty string, the second language has the extension `_1`. The extension for the active language can be retrieved with `GetLanguage Postfix()`.

In the current sample, the produced filename would be `index.htm` for the default language and `index_1.htm` for the second language. The content (see listing 2 on the following page) is read with `HtmlGetFile()` and appended to the variable `$sContent`.

Then a second module is created (overwriting the previously created module, which is no longer needed), all records are queried (the module internally sets filters to query only valid records) and merged with the template `index.tpl` (see listing 10 on page 65), which is located in the current directory. The return of the method `$oModul->Output( 'index.tpl' )` is appended to the content variable.

By assigning the variable `$sContent` to the placeholder `content` of the template object, every ocurrence of `content` in the template file is replaced with the content of `$sContent` (puh, a lot of content here...).

Now the page is finished and the result is sent to the Web browser by calling the `End()` method. Although in object oriented terms, `index.php` should contain a derived class of CWebpageX, procedural code is used to make it easier for beginners.

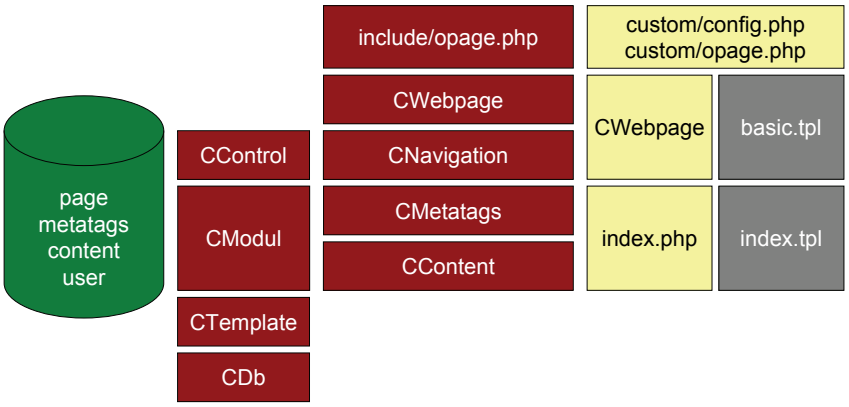


Figure 16: *oPage sample page*

Listing 1: *example index.php*

```php
<?php

include_once 'include/opage.php';

// create page
$oPage = $oFactory->Webpage( 'home', 'CWebpageX' );
$oPage->Init();
$oPage->Begin();

// create modul, query and output data
$oModul = $oFactory->Modul( 'CMetatags' );
$oModul->SetFilter( 'name', 'default' );
$oModul->QueryData();
$oModul->OutputTpl( $oPage->Template() );

$sContent = '';

// append static content from html file
$sContent .= HtmlGetFile( 'index' . $oApp->GetLanguagePostfix() . '.htm' );

// append content from a modul
$oModul = $oFactory->Modul( 'CContent' );
$oModul->QueryData();
$sContent .= $oModul->Output( 'index.tpl' );

// assign the content to the page template
$oPage->Assign( 'content', $sContent );

// finish page and send output to browser
$oPage->End();

?>
```

Listing 2: *example index.htm*

```html
<html>

<head>
    <title>This is a file with content</title>
```

```
5  </head>
6
7  <!-- remember: only the content between <body> and </body> is used -->
8  <body>
9
10 <h1>Headline</h1>
11 <p>Some static text in index.htm</p>
12
13 </body>
14
15 </html>
```

3.3 oPage Core

In this section all relevant parts of the oPage framework are described. Figure 17 provides an overview of the components of oPage

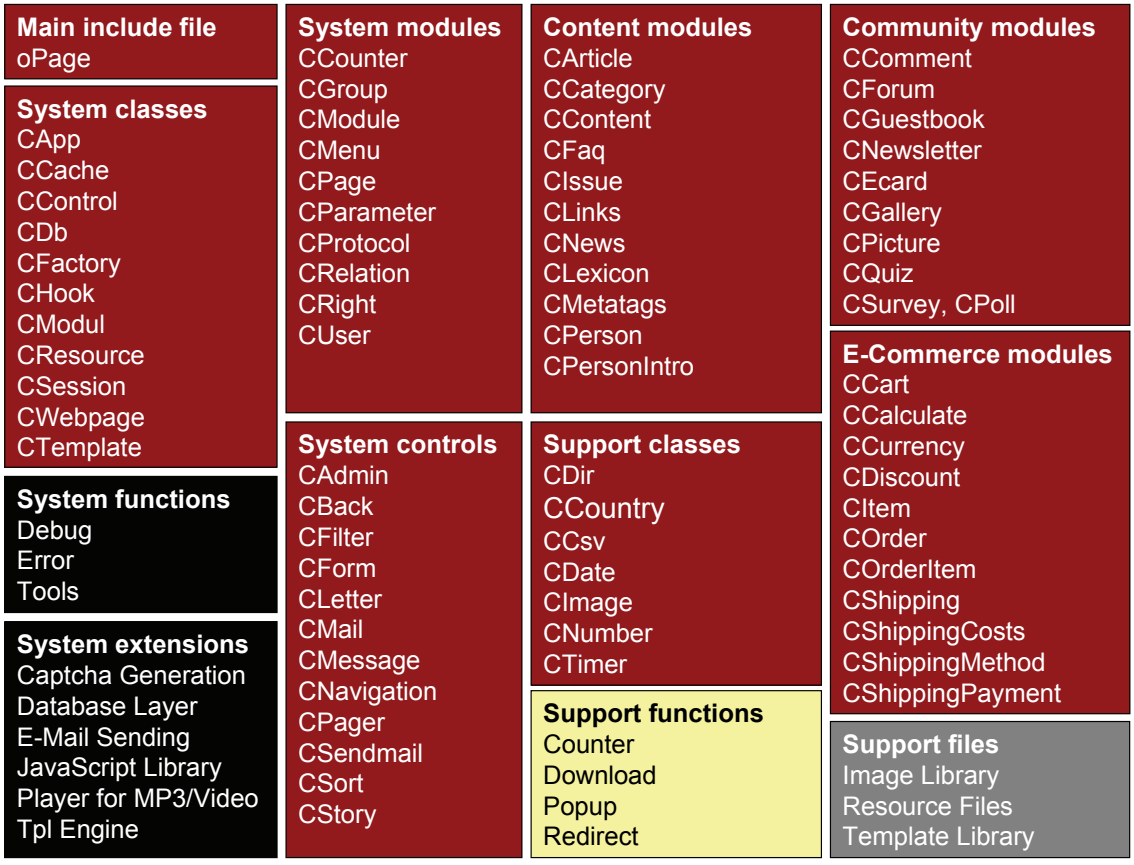


Figure 17: *oPage system*

Main include file

`include/opage.php` is the main project file, which has to be included in every script. It configures a few PHP parameters and sets the global path to the Web site's root directory. Then it instances the global factory object, which is responsible for creating all other objects. The Web site specific configuration file is included and the global application object is initialized as well as the database connection and the counter object. Also the parameters stored in the database are loaded. If the Web site specific file `custom/opage.php` exists, it will be included. In this script additional objects can be created and the oPage framework can be configured.

System classes

The classes described here represent the core of oPage.

CApp

CApp works as an abstraction layer to hide the different configurations of Web servers. An instance `$oApp` is always created by the main include file as a global object. Up to four different sets of application parameters can be applied, from which `$oApp` selects the appropriate set depending on the current hostname. Additionally CApp provides methods to get the active configuration values and retrieve and set HTTP parameters and cookie values.

CCache

This class is used to store dynamically created content like pages or page fragments in the cache. Currently the cached content is stored in the file system.

CControl

CControl is the base class for controls. In contrast to modules, controls manage data which is not stored in a database. Examples for controls are the administration interface, which itself uses controls for managing forms to enter, to filter, to sort and to page data. Menus can be managed with the navigation controls. A link to the previous page is provided by the back control. System messages can be collected with the message control while the mail control is used to send emails.

CDb

This database class provides a facade[64] to the underlying database classes. In oPage, data is stored in arrays of rows which are exchanged with the database by methods like read, insert, update and delete. Other methods are used to create and alter the database tables. To support other DBMS, a new underlaying class can be created and used instead of the MySQL class.

CFactory

The factory is used to create instances of classes. No object should be created with the `new` operator. This makes it easy to configure objects at a central place. The global factory object is set in the main include file.

CHook

With a derived class of CHook it is possible to react to events. When a newsletter recipient registers, an event is fired and the hook system is called, where e.g. the latest newsletter can be sent immediately. This allows Web site specific code to be separated from general code.

CModul

CModul is the base class for modules. A module is used to manage structured content and is usually associated with a database table. It works as an object relational mapper and provides methods to query data and to insert, update and delete records. Modules can provide information for creating the user interface. In general, the class is used in combination with the template engine as a report generator to create the content of a Web page.

CResource

Its class provides texts for system messages. They are divided into error and notification messages and are available in English and German. This class is mainly used by the admin control in combination with the message control.

[64] `http://en.wikipedia.org/wiki/Facade_pattern`

CSession

This class implements a custom session handler to store data, which should be available the next time the user requests a page. An instance of CSession is not automatically created by the main include file, but it can be created with the factory object in the Web site specific `custom/opage.php` file.

CTemplate

The main task of the template engine is to merge data with a layout file to create the final output. Templates contain placeholders which are replaced with data by the template engine.

CWebpage

This class is used to create the output for a Web page. Usually a basic template is loaded (which contains the common parts of a Web page) and some values are assigned. Then other objects assign more values. Finally the content is retrieved and sent to the browser. To increase the performance of the Web site, the created output can be cached in a file and loaded at the next call. A derived class can be used, when additional features are needed.

System functions

This section describes functions, which are not limited to oPage and can be used in other projects as well.

Debug

Offers a few functions to display debug information. If a parameter is supplied, debug tries to print the content of the variable. If debug is called without arguments, it displays the call stack.

Error

Is included by default on development servers. It changes the PHP error parameters and installs a custom error handler. If a fatal error occurs, it displays the call stack using the above debug functions. Currently, all error messages are displayed in the browser, but it is possible to use other output methods.

Tools

Is a collection of general purpose functions. There are functions to create passwords, encrypt and decrypt strings, validate email addresses and credit card numbers, tools to load and manipulate html code, support functions to handle HTTP file uploads, as well as functions to handle arrays, to read, write and copy files.

System extensions

Here are some nice tools, which extends oPage.

Captcha Generation

A Captacha[65] is a method to make (relatively) sure, that a form is filled out by a human being and not by a robot. Here a Captacha is an image with letters, which have to be written into a form field to successfully submit the form. It can be used to avoid e.g. guestbook spam.

Database Layer

The database layer class provides access to the database. Currently, only a database layer for MySQL is supported. Layers for Microsoft SQL server, Postgres SQL, Oracle, Sybase and ODBC exist, but have not been tested. The database layer is part of the PHPlib[66] and published as open source under the GPL.

[65] Captacha: Completely Automated Public Turing Test to Tell Computers and Humans Apart
[66] A collection of PHP classes. `http://phplib.sourceforge.net/`

E-Mail Sending

For sending emails, oPage uses PHPMailer[67], which is a class for sending email using either sendmail, PHP `mail()`, or SMTP. It was originaly written by Brent R. Matzelle and is published as open source under the LGPL.

JavaScript Library

Here are some files with nice JavaScript functions for client side cookie handling, date and number formating, form evaluation and window opening.

Player for MP3/Video

Three Flash programs are available to play audio and video files directly within the Web site. For audio (mp3) files, the XSPF player[68] is used, which is published as open source under the BSD license. For playing videos (flv), the FlowPlayer[69] is used, which is published as open source under the Apache license.

Tpl Engine

It was originally developed as XTPL by Barnabás Debreceni and published under the GPL. Today it is mantained by Jeremy Coates as XTemplate[70]. Because the original code was to slow, the class has been completely rewritten, but the original template syntax has been preserved.

System modules

Like other modules, system modules are used to manage data stored in a database table. In contrast to content modules, the system modules manage other data, not directly content of the Web site.

CCounter

Each counter entry can have a type, an URL, a title and the current counter number. A counter can be increased with the `Count()` method, the current value can be retrieve with the `GetCounter()` method.

CGroup

This class is used to manage user groups. Each group can have multiple users as members and can get assigned multiple rights. To store this assignments (which are m:n relations) the CRelation module is used.

CModul

This module is used by the setup program. In the backend the user can add new modules to be used in oPage.

CMenu

The CMenu module stores references to page entries. The content of the module can be retrieved and forwarded to a navigation control.

CPage

With CPage the properties of a page entry are managed. This covers an unique name, the path to the script, the name of the target frame, a global title, an internal title and special titles to be used in navigation objects.

CParameter

This module stores an unique name, a title and a value for each parameter. Usually only one global instance of the parameter class is created. In the QueryData method all parameter values are read at once. If a parameter does not exist, an empty entry is created by the `GetValue()` method.

[67] http://phpmailer.sourceforge.net/
[68] http://musicplayer.sourceforge.net/
[69] http://flowplayer.sourceforge.net/
[70] http://www.phpxtemplate.org/

CRelation

The CRelation module is used to store m:n relations of modules. For example, if an entry in the links module can have more then one category, an instance of the relation class has to be configured (in the custom factory) to store both link and category ids.

CRight

A entry in the rights tables is defined by a unique name and a title. Each right can be assigned either to a user or a group. To store this assignments (which are m:n relations) the CRelation module is used.

CProtocol

Currently only the newsletter system uses the protocol module. Sending a newsletter can result in a long running process on the server. To monitor this process, the newsletter system writes its current activities into the protocol. Each time a newsletter is sent, a new procotol entry is created and is updated after each recipient processed. If something fails or the script terminates because of a server timeout limit, the sending process can be continued using the information from the protocol entry.

CUser

Each user has a unique username, a password and a full name. If the Web site supports multiple languages, for each user the primary content and administration interface language can be set. Each user can be assigned to multiple groups and have multiple rights. To store this assignments (which are m:n relations) the CRelation module is used.

System controls

Controls provide functions as extensions to the core system and to the modules. All controls are derived from the base class CControl. In contrast to modules, controls manage data which is not stored in a database.

CAdmin

The adminstration control processes the backend forms for managing the content. Two kinds of forms can be used: The list view, where multiple records can be edited at once, and the detail view, where just one record can be modified. The admin control retrieves field names and types from the assigned module. The module is also used to read and write the data to the database table. A filter and a sort control are used to create a user interface for the *WHERE* and *ORDER BY* clauses of the SQL statement. The result set is divided into pages with the pager control, which is also used to navigate through the records. A back control provides a link from a sub page to the main section. If necessary (e.g. to implement a 1:n relation), sub admin controls can be assigned. A modified record can also be stored temporary and displayed for preview.

CBack

A backlink is a link to a previous page including all the parameters to reload the page in the same state as it was. The back control evaluates the query parameter back and creates the link.

CFilter

The filter control is similar to the form control and handles form fields for a user interface to enter *WHERE* parameters. The admin control uses the filter control to create the filter input fields in the list view. The HTML code for the form fields has to be included in the used templates (see the template fragments `admin/custom/index/filter_*.tpl` for details).

CForm

This control processes forms. A form can have a number of fields of different types like text, multiline text, yes/no, checkbox, tristate, radio, dropdown, username, password, email, date and time, captcha tests and file uploads. Depending on the field type, server side input validation can be applied. A form can also have multiple rows. Uploaded images are automatically resized to a predefined width and height. For each field, the possible minimum, the maximum and a default value can be assigned.

CLetter

The letter control creates navigational links to group data by first letters. In combination with a module, the control reads all first letters of the selected column and displays only letters as links where records are available.

CMail

This class integrates the PHPMailer class into oPage (using the facade[71] design pattern). E-Mails can be sent through a SMTP server, with the PHP `mail()` function or by calling the sendmail program. An instance of a module can be assigned. When sending the email, the output method of the module is called to merge the data of the module with the template of the email. It also provides methods to encrypt the email with a public key system like PGP or GPG.

CMessage

The message control collects system notifications and errors. The admin control gets the error message from the resource file and puts it into the message control. Later all messages stored in the control are displayed at once (using a template).

CNavigation

This control is used to create navigational links and menues to navigate through the site. It can handle a simple flat list as well as an hierarchy of links. The effective functionality of the navigation has to be provided in a template. With different templates, the same program can create a pulldown menu or a tree control. The factory is used to create instances of CNavigation and CMenu, which is a virtually derived class of CNavigation. Details are described in section 3.12 on page 75.

CPager

The pager control is used to create navigational links (a "pager"). The total number of available pages and the current page have to be set. If the total number of pages is one, the page control does not display any links. By default, nine page numbers around the current page are displayed at most. The layout of the page is provided by a template. Standard pager templates are available in the template library.

CSendmail

This class is a generic class to handle data submitted by forms. First the empty form (which is provided in a template) is displayed. When the form is submitted, various parameters (provided in hidden fields by the form template) are evaluated. The data can be stored in a CSV file and can be sent to a predefined recipient and to the email address the user provided.

CSort

The sort control is similar to the filter control and handles form fields to create a user interface for *ORDER BY* parameters. For each field submit buttons for sorting upwards and downwards are created and the current sort order is stored in a hidden field. The sort control is used by the admin control to create the sort buttons in the list view. The HTML code for the form fields has to be provided in the template used (see the template fragment `admin/custom/index/sort_field.tpl` for details).

CStory

The story control is used to split a long HTML page into smaller pages. Content can be set with the `SetHtml()` method or loaded from an HTML file. The default page break separator is `<!- break ->`. A pager control can be used to create links to navigate between the pages. If the page is created in print mode (that is when the query parameter print is set to 1), the whole content is displayed as one page.

[71] `http://en.wikipedia.org/wiki/Facade_pattern`

Support classes

These support classes are still essential for oPage, but are not related to the field of content management systems and can be used in other PHP programs as well. Some of these classes are already published as open source[72].

CDir

With CDir the content of directories can be read. Regular expressions can be used to include and exclude file and directory names from the search. Parameters define, whether only directories, only files or both directories and files should be listed. If needed, the class can recurse into subdirectories.

CCountry

In `include/countryiso.php` there are two classes. CCountryIso (which can be created by the factory by calling the Country Method) and the CStateUs. CCountryIso holds an array of two-letter ISO codes and the matching country name. The country names are available in German, English, French, Italian and Spanish. The `GetList()` returns a list of all countries, while the method `GetListEU()` returns only the 25 member states of the European Union. Single ISO codes can be matched with `GetCountry()` to the corresponding country name. The CStateUs maintains a list of the states of the USA. The names of this states are only available in English.

CCsv

Two dimensional arrays can be stored as a CSV file with this class. Existing data in the CSV format can be read into an array of rows. When column names are used, they are written as first line into the file. Each value can be enclosed by double quotes. By default, field names are not written, a `;` is used as separator and values are enclosed by double quotes.

CDate

This class is used to format date values. It provides names for weekdays and months in both German and English. Date values can be converted from string (e.g. results from database queries) to Unix time stamps, which can be formated and returned as strings. Each supported date format has a unique number. When calling the format method, either a number of a predefined format or a string, which is used as a parameter for the PHP `date()` function, has to be provided.

CImage

Images can be resized with this class. It supports ImageMagick[73] and the PHP library GD version 1 and 2. When using GD, the class provides a cubic resize method, which provides better results than the GD resize function, but is way slower and not recommended.

CNumber

Similar to CDate, this class is used to format numbers. Different predefined formats are identified by a number, or the format can be supplied as a string using the *printf* syntax.

CTimer

A tiny class to measure the execution time of a script. An instance can have multiple timers. The result can be printed using PHP `echo()`, or embedded into the page as HTML comments or sent to a syslog service.

Support functions

In this section support scripts are described, which can be used for the frontend of the Web site.

Counter

This script is used to increment counter values. The URL can be used as source of a tracking image.

[72] `http://www.phpclasses.org/`

[73] A set of tools for command line image manipulation. `http://www.imagemagick.org`

Download

The download script provides a secure method to download files from the Web server. Only files in the `static` directory can be downloaded. The script can set a user friendly file name in the HTTP header to replace the internal CSM file name. Each download can be recorded using the counter module.

Popup

The `index.php` script is used to display an image in a separate window. The `play.php` script is used to display audio and video files.

Redirect

The script is used to count redirects to external Web sites. The URL and some information for the counter object have to be provided as HTTP parameters to the script.

Support files

The files described here are provided by oPage as a base for the frontend developer.

Images

oPage provides default images for buttons (*more*, *back*, *zoom*, *topofpage*, *print*) which can be replaced by custom images representing the design of the website. Also icons for office documents (*doc*, *xls*, *odt*, *pdf*, *ppt*, *rtf*, *zip*), images (*gif*, *jpg*, *png*, *psd*, *tif*) and audio and video files (*avi*, *mov*, *mp3*, *mpg*, *ogg*, *wav*) are available.

Resources

Each resource file contains language dependent strings. These strings can be used in the templates e.g. as labels for form fields. Each string can be overruled by an entry in a custom resource file. See section 3.5.8 on page 59 for a detailed description of resources.

Template library

The template library is a collection of files which can be parameterized and included in templates: Files providing various HTML and XHTML DOCTYPEs, HTML header statements, different layouts for pager controls, various fragments for displaying text, images, download links and inline audio and video players.

Content modules

The following classes are used to manage the content of the Web site. Some of them are more generic (like CCategory, CContent) then others (like CFaq, CLinks). In the factory class a module can be configured and (re)used as a "virtually" derived class.

CArticle

An article can have a date, a title, a sub title, a text, an author and a small image. Each article can have multiple paragraphs (using the CArticleParagraph module, which is virtually derived from CParagraph) and each paragraph can have many pictures (CArticleParagraphPicture, which is a virtually derived class of the CPicture module). A paragraph offers fields like title, abstract, text and attachment (with title, filename and file) and template (used to assign a specific layout to a paragraph).

CCategory

The category module is used to group entries of other modules. For example each shop item can be assigned to a category like shoes, socks, trousers or t-shirts. Each category can have sub categories. If entries have to be assigned to more then one category, a relation module has to be used (to store the m:n relation). The category class is not used directly, instead virtual classes (e.g. CContentCategory, CLinksCategory) are used, which are created with the factory. If needed, each content module can have an associated category module.

CContent

The class CContent is a generic module and is used to store all kinds of content. It supports fields like title, abstract, image and up to five groups of fields like subtitle, text, additional text, image (small, medium and big) and fields for attachments. Content entries can be grouped with the CContentCategory class.

CFaq

Questions and their answers can be managed with the frequently asked questions (FAQ) class. Additionaly to the title, the question and its answer also the name and email address of the person who asked the question and the name of the person who answered the question can be stored. If needed, entries can be grouped with the (virtual) category module CFaqCategory.

CIssue

This module manages issues of e.g. newspapers. Each issue has a title, a release date, a title image, a front image and a small image can be stored. The CIssue class is usually used in combination with the CArticle module.

CLexicon

The lexicon is used to store terms and definitions. A brief and a long description and its source can be retained.

CLinks

This module is used for a link collection to other Web sites. For each URL a title, a description, a small and a big image can be stored. Links can be grouped with the category module CLinksCategory.

CNews

The news module is similar to the content module, but supports a lot more fields by default, and therefore is a bit slower then CContent.

CMetatags

Metatags are an (invisible) part of HTML pages and are important for search engines. Possible values are the author, information when to revisit the page, abstract, description and keywords of the page.

CPerson

The person module is a generic module to store personal data like firstname, lastname, address, telephone numbers and email addresses. It is used as CNewsletterRecipient or CCustomer (which both are virtual classes). It also provides methods to create HTML forms and handle post backs to add, update and delete persons.

CPersonIntro

This class is used in combination with the CPerson module and provides different forms of salutations. For CCustomer the according name of the virtual class is CCustomerIntro.

Community modules

The modules described in this section all have some kind of user interaction at the front side of the Web site. To create the forms where the user can enter the data, methods of the base class CModul are used.

CComment

The comment module is used to store user comments to other module entries. With it, comments to news or shop items can be managed. Each entry can have a type (e.g. `news` or `shop_item`) and an identifier (like the value of `news_id` of the corresponding news entry).

CEcard

This class is used to store ecards. An ecard has fields like title, abstract, text, `image_small` and `image_big`. Ecards can be displayed on the Web site and linked to a form, where the user can enter name and email address of a recipient. With methods of CModul, the ecard is merged with a template and sent using CMail.

CForum

Forum entries are stored as comments (CForum is a virtually derived class of CComment). Other forum classes are used to group forum entries (CForumCategory, which is a virtually class derived from CCategory) or to store information about forum users: CForumUser (CPerson), CForumUserIntro (CPersonIntro), CForumUserCategory (CCategory)[74]

CGallery

In a gallery collections of pictures are managed. Each gallery can have a date, a title, some description, information about location and photographer and a small image, which is used as a preview. The pictures are stored in the (virtual) module CGalleryPicture (see CPicture).

CGuestbook

This module stores guestbook entries, which users of the Web site have added. Each entry can have a title, text and the name and address of the writer.

CNewsletter

The newsletter module is used to store the content of a newsletter. oPage also provides a script, which sends the newsletter as personalised email to each recipient. The CNewsletter class is a virtual class, in reality the CNews module is used. The recipients are managed by CNewsletterRecipient (which is CPerson). The script, which sends the newsletter writes its actions into CNewsletterProtocol (a virtually class derived of CProtocol).

CPicture

A picture entry can have a title, some text, a unique picture code, a small and a big image and information about the location and the photographer. Each picture can be assigned to a customer category. This can be used to display pictures depending on the category of the currently logged in user. Mostly, the module is used in combination with a gallery entry (the according virtual class is named CGalleryPicture) or with a paragraph (as CParagraphPicture or CArticleParagraphPicture).

CPoll

A poll is a question with many possible answers. The poll has a title, an abstract, a text and a begin and end date. It can be specified, that only one or more than one answers can be selected. The number of votes are counted. A required minimum number of votes can be specified, the current result gets is displayed only after the limit has been reached. Each selection of an answer is counted in the answer module. For each answer, the next question, which should come, if the answer was selected, can be set. Also see CSurvey in paragraph 3.3 on the next page.

CQuiz

A quiz has a title, an abstract, an introduction and a final text, which is displayed when the quiz has been finished. Each quiz can have multiple questions (CQuizQuestion), each having a title, the question, a text for right and wrong answer and a number of points, when the question is answered correctly. Each question can have several possible answers (CQuizAnswers), each with a title and a flag, if this answer is a correct one.

[74] As you can see, with virtually derived classes the number of class files can be reduced a lot.

CSurvey

With this module complex surveys can be done. A survey has a title, an abstract, a start and a finish text. Also a begin and end date can be set. Each survey can have multiple questions (CSurveyPoll) and each question many answers (CSurveyPollAnswer). Also see CPoll and CPollAnswer.

E-Commerce modules

The modules in this section are all used in an online shop. Some modules are more generic then others and can be used alone (CItem) or in combination with other modules for other purposes (CCart in combination with CNews to create a basket of news items). Modules like CShipping can only be used in combination with CShippingCosts and CShippingPayment.

CCart

The shopping cart is always used in combination with another module, e.g. CShopItem. It stores an id, the quantity and optionally some ad

CCalculate

This module does not store any information in the database. It provides methods to calculate the item price in the default and an alternative currency. Also item and order discounts, shipping costs, cash discounts and the total order amount is computed. The shipping costs can have a fixed value and a value depending on the total weight or the total item number. All row and total values are calculated as net and gross sum, are assigned to the template and are also available via class methods.

CCurrency

Currency codes and their exchange rate are stored in this module. A short title (e.g. EUR or ATS), a normal title (Euro, Österreichische Schilling) and the exchange rate (1, 13.7603) can be stored.

CDiscount

This module provides a convenient way to promote shop items. It is similar to CNews and can manage stories with references to and discounts for shop items. The class provides fields for title, abstract and up to ten groups of sub title, text, another text, an URL, a small and a big image. Also a file can be attached. Numbers of items, which should be displayed along with the text, can be set (separated by comma). The discount can either be a number or a percentage and can be set for category of items, for items of a certain region, or for a list of item numbers or just for new items.

CItem

Item fields are title, short title, abstract and text, three images (each in small, medium and large) and properties like type, color, size, length, width, height, gross and net weight, unit, price, discounted price and begin and end date. Flags mark the item if it is new, old, available soon, orderable, already sold, in stock, on sale. If needed, up to three files can be attached. A reference to an instance of a shopping cart can be set to supply the quantity the user has selected. The item class evaluates the discounts, but does not calculate row amounts or total amounts. These calculations are done by CCalculate.

COrder

All order specific data is stored with this class. This includes order number, shipping and payment method, credit card data, total weight, amount, discount, effective shipping costs, cash discount percentage and amount, the customer number, name and address along with an optional shipping name and address. In the online shop this class is used as CShopOrder.

COrderItem

The order item module is similar to the item module, but has not the calculation methods. When the order is submitted, the values of the shop items are copied to the order item object to retain the correct item data

as they where when the order was created. For the same reason the title (instead of a reference id) of the shop item category is stored. In the online shop, this class is used as CShopOrderItem.

CShipping

This module stores the effective offered shipping and payment options. For a number of countries (a string of two-character ISO codes separated by a blank) a payment method, payment costs (e.g. base costs for cash on delivery) and a shipping method can be stored. The total of the order has to be higher then an optional minimum limit, and shipping is free, when the free limit is exceeded.

CShippingCosts

For each shipping method and country combination, shipping costs for different weight classes can be stored. Fixed costs (independent of the weight) and variable costs per weight unit can be set. To reduce the number of rows the country field can store multiple two-character ISO codes separated by a blank.

CShippingMethod

With this module the supported shipping methods like Post, Post-Express, DHL, by land, by air etc. are managed. For each entry a unique name and a title is stored. This class is a virtually derived class of CCategory.

CShippingPayment

Various payment methods like cash on delivery, payment in advance or credit card can be defined in this module. Each entry has a unique name, title, description and a text with payment instructions, which is displayed to the user, after the order was completed. Optionally a cash discount percentage can be set.

3.4 The factory class

The factory is used to create instances of classes. No class should be created with the new[75] operator of PHP.

Customization

When the same module (which is a class from the programmers point of view) is used in two different projects or differently in the same project, module specific customizations have to be applied every time a new instance of the class is created.

Each project has its own custom factory, where the objects can be customized as needed. Using a factory, the creation and the configuration of objects can be centralised.

Virtual classes

Imagine two similar category modules like one for shop items (CShopItemCateogry) and one for link categories (CLinkCategory). One option is to derive both from a common category module (CCategory). This results in two additional source files.

Using the factory to create an instance of a class called CLinksCateogory, always an instance of CCategory is created and then named and customized to look like a derived class. When needed the factory can also be configured to use a real class derived from CCategory.

Derived classes

It is obvious that it is not possible to add new methods to virtually derived classes. But for a real derived class, a new unique class name has to be used instead of the name of the original class. For instance, the admin script for the module would have to be changed or another admin script would be needed, to keep the old script compatible with other projects using the same module.

[75] http://www.php.net/manual/en/language.types.object.php

The class CLinksCategory, which is derived from CCategory, can be extended as CLinksCategoryX. Every time CLinksCategory is requested, the factory returns an instance of CLinksCategoryX.

The main task of the factory is to create objects. All framework classes like modules, controls, CDb or CTemplate can be instanced by the factory.

Listing 3 shows a fragment of the factory class. Two virtual classes CCustomer and CLinksCategory are instanced using the real classes CPerson and CCategory. The instance of CLinksCategory gets customized: the field `title_menu` is not used and the field `title_navigation` should be displayed in the list view of the administration interface. Similar methods are used to create instances of controls like menus or forms. Other classes like the template engine, the class for image manipulation or the data access layer are instanced using the factory.

Listing 3: *Factory.php*

```
1  <?php
2
3  Class CFactory
4  {
5      // Public method to instance a module
6      Function Modul( $sClass, $sName = Null, $fInclude = Null )
7      {
8          $oObject = $this->_Modul( $sClass, $sName, $fInclude );
9
10         if ( !is_object( $oObject ) )
11             return( $oObject );
12
13         $oObject->SetClass( $sClass );
14         $this->_ModulConfigure( $oObject );
15
16         return( $oObject );
17     }
18
19     // This method includes the class file and creates the instance. It can be overruled in
       the derived custom factory.
20     Function _Modul( $sClass, $sName = Null, $fInclude = Null )
21     {
22         // path to the project root directory
23         global $sRoot;
24
25         if ( is_null( $fInclude ) )
26             $fInclude = true;
27
28         switch( $sClass )
29         {
30             case 'CCustomer':
31                 if ( is_null( $sName ) )
32                     $sName = 'customer';
33                 $sClass = 'CPerson';
34             break;
35
36             case 'CLinksCategory':
37                 if ( is_null( $sName ) )
38                     $sName = 'links_category';
39                 $sClass = 'CCategory';
40             break;
41         }
42
```

```
43          if ( $fInclude )
44          {
45              $sInclude = $sRoot . 'include/modul/' . strtolower( substr( $sClass, 1 ) ) . '.
                    php';
46              if ( !file_exists( $sInclude ) )
47                  return( null );
48              include_once $sInclude;
49          }
50
51          if ( is_null( $sName ) )
52              $oObject = new $sClass();
53          else
54              $oObject = new $sClass( $sName );
55
56          return( $oObject );
57      }
58
59      // In this method the object gets configured. If needed _ModulConfigure can be overruled
            in a derived class.
60      Function _ModulConfigure( &$oObject, $sClass = null )
61      {
62          if ( is_null( $sClass ) )
63              $sClass = $oObject->GetClass();
64
65          switch( $sClass  )
66          {
67              case 'CLinksCategory':
68                  $oObject->SetFieldUseAdminList( 'title_navigation' );
69                  $oObject->SetFieldUse( 'title_menu', false );
70              break;
71          }
72      }
73
74      // This is the factory method to create the template engine object.
75      Function Template( $sFilename, $sBlock = null, $aParameter = null, $sLanguage = null,
            $sName = null )
76      {
77          global $sRoot;
78          global $oPage;
79
80          $sInclude = $sRoot . 'include/template.php';
81          if ( !file_exists( $sInclude ) )
82              return( null );
83          include_once $sInclude;
84
85          $oTemplate = new CTemplate( $sFilename, $sBlock, $aParameter, $sLanguage, $sName );
86          if ( is_object( $oPage ) && is_object( $oPage->Template() ) )
87              $oTemplate->SetVars( $oPage->GetTemplateParameter() );
88
89          return( $oTemplate );
90      }
91  }
92
93  ?>
```

3.5 The template engine

One of the important base components of oPage is the template engine. It was originally developed as *XTPL* by Barnabás Debreceni and published under the GPL. It has been completely rewritten for oPage but the original template syntax has been preserved. The main code is located in the class CTpl (`include/tpl/tpl.php`) and is independent from the framework and can be used in other PHP projects. This class is extended and integrated into oPage with the class CTemplate (`include/template.php`).

The main task of the template engine is to merge data with a template to create the final output. The template contains placeholders which are replaced with data by the template engine.

With the template engine all kinds of text based output can be created. Support for different character-encodings for HTML, XML, RTF or plain text is provided.

Loading and analysing a template requires a lot of effort. The engine provides ways to cache compiled templates in the file system. If the template has been changed, the engine automatically updates the cached file.

The template engine is heavily used by the classes CModul and CControl (and their derived classes) and understanding the template syntax is crucial to create and modify Web sites using oPages.

3.5.1 Replacing variables

Variables are placeholders which are filled by the template engine with values. The names of the variables are case sensitive and must be enclosed by `{}`.

The examples listing 4 and listing 5 on the following page illustrate the usage of the engine. An instance of the template engine is created and the template file gets loaded. Then values are assigned in different ways and merged with the template. The (main block of the) template gets parsed and the text is sent to the client.

Listing 4: *Template Engine sample1.php*

```php
<?php

include_once 'include/tpl/tpl.php';

// create an instance of the template engine
$oTemplate = new CTpl( 'sample1.tpl' );

// assign a value for the title of the page
$oTemplate->Assign( 'title', 'Welcome to sample 1' );

// create and assign an array of values
$aValue[ 'firstname' ] = 'Hannes';
$aValue[ 'lastname' ] = 'Dorn';
$aValue[ 'city' ] = 'Vienna';
$oTemplate->Assign( $aValue );

// create and assign another array of values
$aData[ 'title' ] = 'Sample 1';
$aData[ 'filename' ] = 'sample1.tpl';
$aData[ 'text' ] = 'Shows how to use replacement of variables.';
$oTemplate->Assign( 'data', $aData );

// parse main block
$oTemplate->Parse();

// output created content
```

```
27 echo $oTemplate->Text();
28
29 ?>
```

Listing 5: *Template Engine sample1.tpl*

```
1  <!-- BEGIN: main -->
2  <html>
3
4  <head>
5      <title>{title}</title>
6  </head>
7
8  <body>
9
10 <h1>{title}</h1>
11
12 <p>
13     <strong>Firstname:</strong> {firstname}<br>
14     <strong>Lastname:</strong> {lastname}<br>
15     <strong>City:</strong> {city}
16 </p>
17
18 <p>
19     <strong>Title:</strong> {data.title}<br>
20     <strong>Filename:</strong> {data.filename}<br>
21     <strong>Text:</strong> {data.text}
22 </p>
23
24 </body>
25
26 </html>
27 <!-- END: main -->
```

Variable names can be strings with the following characters: `a-z`, `A-Z`, `0-9`, `äöüÄÖÜß` and `_.␣=<>!#:|&%,/-`.

3.5.2 Parsing of blocks

Blocks are used to divide a template into smaller parts. They are defined as HTML comments and start with `<!- BEGIN: blockname ->` and end with `<!- END: blockname ->`. Block names are case sensitive and have to be unique. A template starts always with the block `main` and can be divided into smaller parts by defining sub-blocks. Each sub-block can be parsed multiple. Single blocks can be referenced by their full qualified block name. The full name of the block used to parse `field1` in listing 6 on the next page and listing 7 on the facing page is `main.DATA.field1`.

Blocks can be combined by logic operators. Currently `&` (and) and `|` (or) operators are supported. Only one type of operator is allowed at a time.

```
1 <!-- BEGIN: block1&block2 -->
2 This text is displayed, when block1 and block2 are both parsed.
3 <!-- END: block1&block2 -->
```

For each combination of block names, the block extended by `#header`, `#footer`, `#begin` and `#end` are parsed as well. This is just to have more blocks to be used by module and control classes.

```
1 <!-- BEGIN: block1&block2&#header -->
2 This text is displayed, if block1 and block2 are both parsed.
```

```
<!-- END: block1&block2&#header -->
```

Listing 6: *Template Engine sample2.php*

```
<?php

include_once 'include/tpl/tpl.php';

// create an instance of the template engine
$oTemplate = new CTpl( 'sample2.tpl' );

// assign a value for the title of the page
$oTemplate->Assign( 'title', 'Welcome to sample 1' );

// create and assign array of values
for( $i = 0; $i < 10; ++$i )
{
    // clear data array
    $aData = array();

    // some data for each row
    $aData[ 'row' ] = $i;

    if ( $i % 2 == 0 )
        $aData[ 'field1' ] = 'Value ' . $i;
    else
        $aData[ 'field2' ] = 'Value ' . $i;

    // assign data to template
    $oTemplate->Assign( 'data', $aData );

    // parse subblock for variable
    foreach( array_keys( $aData ) as $sName )
        $oTemplate->Parse( 'main.DATA.' . $sName );

    // parse subblock for the row
    $oTemplate->Parse( 'main.DATA' );
}

// parse main block
$oTemplate->Parse();

// output created content
echo $oTemplate->Text();

?>
```

Listing 7: *Template Engine sample2.tpl*

```
<!-- BEGIN: main -->
<html>

<head>
    <title>{title}</title>
</head>

<body>

```

```
10 <h1>{title}</h1>
11
12 <!-- BEGIN: DATA -->
13 <p>
14     <!-- BEGIN: row --><strong>Row:</strong> {data.row}<br><!-- END: row -->
15     <!-- BEGIN: field1 --><strong>Field 1:</strong> {data.field1}<!-- END: field1 -->
16     <!-- BEGIN: field2 --><strong>Field 2:</strong> {data.field2}<!-- END: field2 -->
17 </p>
18 <!-- END: DATA -->
19
20 </body>
21
22 </html>
23 <!-- END: main -->
```

Block names can be strings with the following characters: `a-z`, `A-Z`, `0-9`, `äöüÄÖÜß` and `_=<>!#|&-`.

3.5.3 Global and local values

Global and local (see table 2) values are variables provided by the template engine. Local values are related to the current template file.

Table 2: *Global and local values*

name	description
`{global.now}`	current date and time as unix timestamp
`{global.sessionname}`	current session name
`{global.sessionid}`	current session id
`{global.upload_max_filesize}`	maximum filesize, which can be uploaded to the server
`{local.template}`	current template filename
`{local.template_dirname}`	directory where the template file is stored
`{local.template_basename}`	filename of the template without path
`{local.template_basename_filename}`	filename without path and extension
`{local.template_basename_extension}`	extension of the template filename (usually `.tpl`)

The oPage specific class CTemplate, which is derived from CTpl provides additional global values like date, time, weekday, client IP, server name, application directory, current directory, script name or the current language (see table 3).

Table 3: *oPage specific global and local values*

name	description
`{global.date}`	current date in a language depending format
`{global.time}`	current time in a language depending format
`{global.weekday}`	day of week as language depending string
`{global.clientip}`	IP address of the client computer
`{global.servername}`	name of the server like www.domain.ext
`{global.queryseparator}`	usually `?`, but can be set to `/` in `custom/config.php` to optimize the page for search engines
`{global.paramseparator}`	usually `&`, but can be set to `/` in `custom/config.php` to optimize the page for search engines
`{global.http}`	address of the web server as set in `custom/config.php`
`{global.https}`	address of the SSL web server as set in `custom/config.php`

continued on next page

Table 3: *oPage specific global and local values* continued

name	description
{global.root}	the root directory of the web site as set in custom/config.php
{global.path}	relative path to the root directory from the directory where the current script is stored
{global.dir}	current sub directory (below the {global.root} including a closing /
{global.language}	short name of the current language as defined in custom/config.php
{global.language_id}	language ID as defined in custom/config.php
{global.language_full}	full name of the active language as set in custom/config.php
{global.language_postfix}	language filename and field extension as defined in custom/config.php
{global.parameter}	an array of the current HTTP get or post parameter values as return by $oApp->GetParameters()
{global.scriptname}	name of the active script
{global.requesturi}	full request string with scriptname and parameters

3.5.4 encoding values

The engine offers ways to encode data for different output formats like XML, URL encoded, plain text with and without new lines. This is done simply by appending the variable names with an encoding extension e.g. {variable#xml}. If necessary, it is possible to add a #latex or #rtf encoding extension for creating LaTeX or RTF[76] output.

encoding

{variable#encoded} applys standard URL encoding to the value[77]. With {variable#doubleencoded} URL encoding is applied twice (this is used to encode URL parameters on search engine optimized Web sites, which use / instead of ? and & in queries).

{variable#html} creates HTML encoded output. All characters which have HTML entity equivalents are translated into these. Usually this is not necessary, but in some situations it may be required. {variable#xml} encodes the value to be used in XML output. With {variable#text} all HTML tags are removed an only plain text including newlines remains. {variable#plain} is the same as #text, but without the newline characters. {variable#slashed} is like #plain, but single quotes ' are escaped with a / (this encoding is used for JavaScript strings). {variable#nbsp} is also plain text, but blanks are replaced with .

word wrap

With {variable#text76} text can be limited to lines with a maximum length of 76 characters, which is very useful for emails (of course, 76 is only a sample, any other lengths can be used).

align text

The command {tpl.align} is used to align text to a certain column: some text{tpl.align#20}some other text will insert additinal 10 spaces after some text to align some other text at column 20.

3.5.5 formating values

For each variable a format string can be provided using the printf syntax. The format options are appended before the encoding extension and are separated from the variable name with |: {number|%4d#xml} will format the date and encode the result for XML. Additionaly to the printf formats, date, number, size and list options can be used. It is obvious, that # and | can not be used in format strings. Custom format handlers can be added in a derived class of CTpl.

[76] Rich Text Format. Can be read by your word processor.
[77] http://www.php.net/urlencode

printf

`{variable|%4d}` is the same as `{variable|printf %4d}` and formats an integer to a length of at least 4 characters. For a full description of the `printf()` format see the PHP manual[78].

date

`{variable|date H:i:s}` formats a time value like `23:59:59`. For a full description of the `date()` format options have a look into the PHP manual[79].

number

`{variable|number 2, }` is equal to PHP function call `number_format( value, 2, ',', ' ' )` and results in something like `12 345,67`. Once again, the full description of the function `number_format()` can be found in the PHP manual[80].

size

In oPage the size of files is measured in bytes. To display those values in a more readable format, the size option can be used. `{variable|size KB}` or `{variable|size MB}` formats a number to KB (the value is divided by 1024) or to MB (the value is devided by 1024x1024). With `{variable|size XX}` the best representation of the value is used (Bytes, KB, MB, GB or TB). `{variable|size XX 2, }` can be used to apply a number format to the calculated value. This option is as well available for the first two size options and uses the same format as the PHP function `number_format()`.

list

`{variable|list /}` splits the value separated by `/` into HTML `<option value="value">value</option>` entries.

3.5.6 Parameters

Each template can have a block with default parameters. When the template is included in another template, this parameters can be overruled by a value provided with the `FILE` statement. Parameters can be used within the template with the `{PARAM.name}` statement.

```
1  <!-- BEGIN: param -->
2  style="float: left"
3  width="100%"
4  <!-- END: param -->
5
6  <!-- BEGIN: main -->
7  <div style="{PARAM.style}">
8      <table border="0" cellpadding="0" cellspacing="0" width="{PARAM.width}">
9      <tr valign="top">
10         <td>Name:</td>
11         <td>Value:</td>
12     </tr>
13     </table>
14 </div>
15 <!-- END: main -->
```

[78] http://www.php.net/sprintf
[79] http://www.php.net/date
[80] http://www.php.net/number_format

3.5.7 Subtemplates

Templates can be included in other templates with the FILE command: {FILE "filename.tpl"}. Common template fragments are available in include/template/ for HTML header statements, displaying a pager control, file downloading, displaying images, music and video playing. The administration interface makes use of subtemplates (see listing 31 on page 133 and listing 32 on page 134). Along with the FILE command parameters can be provided for the subtemplate which overwrite the default parameter values of the included template. For example {FILE "filename.tpl" parameter1="value1" parameter2="values2"}. This is a very powerful feature in combination with standard template fragments and the PARAM statement.

```
1 {FILE "admin/custom/detail/edit_textandimage.tpl" row=""}
2 {FILE "admin/custom/detail/edit_urlandtarget.tpl" row=""}
3 {FILE "admin/custom/detail/edit_filenameandtitle.tpl" row=""}
4 {FILE "admin/custom/detail/edit_textandimage.tpl" row="2"}
5 {FILE "admin/custom/detail/edit_urlandtarget.tpl" row="2"}
6 {FILE "admin/custom/detail/edit_filenameandtitle.tpl" row="2"}
```

The above sample includes template fragments of the backend to create a user interface for tow groups of fields: text, image, URL, target and file attachments.

3.5.8 Resources

To support multiple languages it is either necessary to have a template for each supported language or to have all static texts in a resource file. oPage supports both options, the use of resources is recommended.

The class CTpl searches for a resource file named like the template file, extended by the language short text and the extension .res. For shop/index.tpl the resource file shop/index#en.res is used. The language extension is only used when a language has been set.

The structure of a resource file is very simple. In the first line there is the name of the resource, in the second line there is the according text. This text has to be in one line, line breaks can be done with the
 tag. All HTML tags are allowed to format the text. Even variable placeholders can be used which are later replaced with values by the template engine. Any blank line found will be ignored.

```
1 opage_common_noframes
2 This website uses frames. Please use a browser, which supports frames.
3 opage_common_topofpage
4 Top of Page
5
6 opage_common_back
7 back
8 opage_common_next
9 next
```

Resource names can be strings of the following characters: a-z, A-Z, 0-9, äöüÄÖÜß and _.#␣.

In the template, resources are referenced by {RES.name_of_the_resource}. Resource strings are provided as is, URL encoded, double URL encoded, as text, plain or slashed (as described in section 3.5.4 on page 57).

The extended class CTemplate loads a global resource file stored in include/resource/xx.res (xx is a language abbreviation as defined in custom/config.php like en or de). For the administration interface, additional resource files for each language are available under admin/custom/resource/xx.res. These resource files are intended to be changed only by the developers of oPage. Custom resource strings can be stored in custom/resource/xx.res and custom/admin/resource/xx.res.

In CTemplate a name for the template can be set, which is used to load a template specific resource file from include/resource/, admin/custom/resource/, custom/resource/ or custom/admin/resource/, where ever a file exists.

3.5.9 Caching

In CTpl caching is not active by default. If caching is activated, take care, that the cache directory exists or can be created by the template engine and is writable by the web server. The created cache file contains all sub templates, and all resources are replaced. To get a unique filename, the real path of the template is extended by the current language and the name of the top level block if it is not main. The directory for the cache files is `cache/` and is by default a sub directory of the document root. The part of the template filename which is below the document root, is prepended by the cache directory name and used as file name for the cache file. Missing sub directories are created automatically.

Every time, a template is loaded, the date of the cached file is checked. If the original file has been changed, the template is recreated. Changes of sub templates are not automatically checked, because a template can contain a number of sub templates. Checking all of them at every call would decrease the performance.

In CTemplate caching is active by default and the cache directory for the templates is set to the sub directory `cache/templates/` of the Web sites root directory (as defined in `custom/config.php`).

3.6 CContent, a generic content module

The class CContent (see listing 8) is a generic module and can be used to store all kinds of content. It supports fields like title, abstract, image and up to five groups containing fields like subtitle, text, additional text, image (small, medium and big) and a field for an attachment.

Listing 8: *module CContent*

```
1  <?php
2
3  include_once $sRoot . 'include/modul.php';
4
5  Class CContent extends CModul
6  {
7      // Constructor
8      Function CContent( $sName = null )
9      {
10         // Set default name
11         $this->SetDefault( 'content' );
12
13         // Initialize base class
14         $this->CModul( $sName );
15
16         // Set default filters
17         $this->SetView( 'active' );
18         $this->SetFilter( 'display', 'n', '<>' );
19
20         // Set default sort orders
21         $this->SetOrder( 'sort' );
22         $this->SetOrder( 'date', 'desc' );
23
24         // Disable unused default fiels
25         $this->SetFieldUse( 'category_id', false );
26
27         // Enabled default fields for listview in the administration interface
28         $this->SetFieldUseAdminList( 'id' );
29         $this->SetFieldUseAdminList( 'date' );
30         $this->SetFieldUseAdminList( 'title' );
31         $this->SetFieldUseAdminList( 'sort' );
32         $this->SetFieldUseAdminList( 'display' );
```

```
33
34          // Enable automatic translation of urls to hyperlinks
35          $this->SetTextPrepareUrls();
36      }
37
38      // Set field definitions
39      Function SetFields()
40      {
41          $this->SetField( 'id', 'ID' );
42          $this->SetField( 'date', 'DATETIME' );
43          $this->SetField( 'title', Null, true );
44
45          ...
46
47          parent::SetFields();
48      }
49
50      // Set field properties for list view in administration
51      Function AdminListProperties( &$oAdmin, $sName )
52      {
53          global $oParameter;
54
55          parent::AdminListProperties( $oAdmin, $sName );
56
57          switch( $sName )
58          {
59              case 'title':
60                  $oAdmin->SetReadonly( $sName, true );
61                  if ( is_object( $oParameter ) )
62                      $oAdmin->SetMaxlength( $sName, $oParameter->Get( $this->GetName() . '
                            _title_maxlength', $oParameter->Get( 'title_maxlength', 255 ) ) );
63              break;
64
65              case 'title_menu':
66              case 'title_navigation':
67                  $oAdmin->SetReadonly( $sName, true );
68                  $oAdmin->SetSyncMaster( $sName, 'title' );
69              break;
70
71              case 'abstract':
72                  $oAdmin->SetLimit( $sName, 150 );
73              break;
74          }
75      }
76
77      // Set field properties for detail view in administration
78      Function AdminProperties( &$oAdmin, $sName )
79      {
80          global $oParameter;
81
82          parent::AdminProperties( $oAdmin, $sName );
83
84          switch( $sName )
85          {
86              case 'title':
87                  if ( is_object( $oParameter ) )
```

```
88                  $oAdmin->SetMaxlength( $sName, $oParameter->Get( $this->GetName() . '
                        _title_maxlength', $oParameter->Get( 'title_maxlength', 255 ) ) );
89          break;
90
91          case 'title_menu':
92          case 'title_navigation':
93              $oAdmin->SetSyncMaster( $sName, 'title' );
94          break;
95
96          case 'date_title':
97              $oAdmin->SetWidth( $sName, $this->GetFieldWidthAdmin( $sName, 20 ) );
98          break;
99
100         case 'abstract':
101             $oAdmin->SetWidth( $sName, 137 );
102             $oAdmin->SetHeight( $sName, $this->GetFieldHeightAdmin( $sName, 5 ) );
103         break;
104
105         case 'more':
106             $oAdmin->SetWidth( $sName, 21 );
107         break;
108
109         case 'title_sub':
110         case 'title_sub2':
111         case 'title_sub3':
112         case 'title_sub4':
113         case 'title_sub5':
114             $oAdmin->SetWidth( $sName, $this->GetFieldWidthAdmin( $sName, 106 ) );
115         break;
116
117         case 'text':
118         case 'text2':
119         case 'text3':
120         case 'text4':
121         case 'text5':
122             $oAdmin->SetWidth( $sName, $this->GetFieldWidthAdmin( $sName, 108 ) );
123             $oAdmin->SetHeight( $sName, 18 );
124         break;
125
126         case 'text_ishtml':
127         case 'text2_ishtml':
128         case 'text3_ishtml':
129         case 'text4_ishtml':
130         case 'text5_ishtml':
131             $oAdmin->SetDefault( $sName, 'y' );
132         break;
133     }
134 }
135
136 // Prepare row data for output
137 Function PrepareRow( &$aRow )
138 {
139     global $oParameter;
140     global $oApp;
141     global $oFactory;
142
143     $aFields = $this->GetFieldList();
```

```
144         foreach( $aFields as $sName )
145             switch( $sName )
146             {
147                 case 'title':
148                 case 'title_sub':
149                 case 'title_sub2':
150                 case 'abstract':
151                 case 'text':
152                 case 'text2':
153                     $this->PrepareText( $aRow, $sName );
154                 break;
155 
156                 case 'image_small':
157                 case 'image_small2':
158                 case 'image_big':
159                 case 'image_big2':
160                     $this->PrepareImage( $aRow, $sName );
161                 break;
162 
163                 case 'date':
164               case 'date_from':
165               case 'date_till':
166                     $this->PrepareDate( $aRow, $sName );
167                 break;
168             }
169 
170         return( parent::PrepareRow( $aRow ) );
171     }
172 }
173 
174 ?>
```

A module works like a report generator: Data is queried from the related table and merged with a template to create the output (see figure 14 on page 37). A complete description of all methods provided by a module as in section 3.8 on the following page.

A module is always derived from the base class CModul, which provides the basic methods common to all modules. In the constructor `CContent()` the default name of the module is set. This name is used as name for the related database table and to identify the module's block in the template file. Then the base class is initialized which also calls the `SetFields()` method. When a module supports `date_from` and `date_till` fields, the default view is set to `active`. Only elements will be showed where the current time falls between `date_from` and `date_till` (if `date_from` and/or `date_till` are set). A filter is set for the field `display` to query rows, where the value of `display` is not `n`. Next the fields are set which are used to sort the queried records. The CContent module provides additional fields to associate rows with entries of a category module, but this fields are not used by default, so they are deactivated.

As seen in listing 1 on page 37, an instance of a module is created by calling the `Module()` method of the factory object. It could be created by calling the new operater, but this is not recommended. The factory not only creates an instance of the class, it initializes and customizes the new object (see section 3.4 on page 50 for details on CFactory).

3.7 CApp

The class CApp is a system abstraction and configuration layer. It provides a simple and consistent interface to retrieve system, Web server, PHP and Web site specific information like hostname, document and application root directory, browser name and version, client IP address, language and query parameters.

Four different arrays with configuration values can be provided to the constructor of the class:

- **Production**: This set of parameters is for the real active Web site and gets used, if no other settings apply.
- **Staging**: The staging server is a testing platform, where the Web site can be tested by the customer before it is moved to the production site.
- **Testing**: The testing site is an internal server of the development company.
- **Development**: This settings are used, when the Web site is running on the computer of a software developer and is automatically selected, when the hostname is `localhost`.

Each parameter set is defined in `custom/config.php` and can have a list of hostnames for which the parameter should be used. A typical `config.php` looks like listing 9.

Listing 9: *config.php sample*

```php
<?php

$aProduction = array
(
    'install' => false,
    'setup' => true,
    'server' => 'example.com,www.example.com,mail.example.com',
    'http' => 'http://www.example.com',
    'https' => 'https://www.example.com',
    'Query_Separator' => '/',
    'Param_Separator' => '/',
    'Root' => '/',
    'ErrorHandling' => 'no',
    'DB' => 'mysql',
    'DB_Server' => 'mysql.example.com',
    'DB_User' => 'example',
    'DB_Password' => 'password',
    'DB_Database' => 'example_com',
    'DB_Table_Prefix' => '',
    'Mail_Server' => 'mail.example.com',
    'Language' => array( '0' => 'Deutsch', '1' => 'English' ),
    'Language_Short' => array( '0' => 'de', '1' => 'en' ),
    'Language_Postfix' => array( '0' => '', '1' => '_1' ),
    'Language_Default' => 0,
    'Admin_Language' => array( '0' => 'Deutsch' ),
    'Admin_Language_Short' => array( '0' => 'de' ),
    'Admin_Language_Postfix' => array( '0' => '' ),
    'Admin_Language_Default' => 0
);

?>
```

This class also manages the language settings for the frontend and the backend Web site. The active language is retrieved from a cookie, if the cookie is not set, the default language of the browser is used. If no valid language is available, the default language as set in `config.php` is used.

Also some search engine optimization is done here: `Query_Separator` and `Param_Separator` can be set to `/` instead of `?` and `&`. This makes a query look like an ordinary path which can result in better search engine positions.

3.8 CModul

The class CModul, located in the `include` directory, is the base class of all modules and provides a wide range of methods:

- Base
- Table
- Fields
- Query
- Output
- Prepare
- Database

A module is associated with data. This is usually a database table, but can also be data from the session object or from some other source. Each derived class provides a list of field names and their types. To query data, the user of a module can provide parameters for the different query clauses. For each field flags can be set, which define if the field should be used in various scenarios. Additional tables can be joined. Complex filters to return specific data can be set. Also the order of the returned rows can be defined. To limit the returned result set the number of the first row and a maximum number of rows can be assigned. All rows returned are stored as name value pairs in a list of two-dimensional arrays.

With the output methods, the module works like a report generator. It queries data and merges the records with a template. All blocks parsed are beneath a block named after the modules name. If the data set is empty, only the block `empty` gets parsed. If records are available, the `HEADER` gets parsed once, for each row the `DATA` section is applied and finally the `FOOTER`. For an example see listing 1 on page 37, listing 2 on page 38 and listing 10.

Each record gets prepared before it is merged with the template. Date and number values are formated, IDs are replaced with values from associated modules (which are queried separately) and URLs in text fields are transformed into click able hyperlinks. Fields containing file names of images and other binary files, properties like the path name, file size and image dimensions are made available to the template through special fields.

Listing 10: *index.tpl sample*

```
1  <!-- BEGIN: main -->
2
3  <!-- BEGIN: content -->
4
5      <!-- BEGIN: empty -->
6      <p>Nothing found!</p>
7      <!-- END: empty -->
8
9      <!-- BEGIN: HEADER -->
10     <p>This text is above the records.</p>
11     <!-- END: HEADER -->
12
13     <!-- BEGIN: DATA -->
14         <!-- BEGIN: title --><h2>{DATA.title}</h2><!-- END: title -->
15         <!-- BEGIN: text --><p>{DATA.text}</p><!-- END: text -->
16     <!-- END: DATA -->
17
18     <!-- BEGIN: FOOTER -->
19     <p>This text is beneath the records.</p>
20     <!-- END: FOOTER -->
21
22 <!-- END: content -->
23
24 <!-- END: main -->
```

3.8.1 Base

If no module name is specified by the constructor, the default name gets used, which has to be set in the derived class. Then the value for the field name postfix of the active language is set. This postfix is appended to all multilanguage fields. A global table prefix can be used to separate oPage tables from other projects. A global language postfix and table prefix can be set in `custom/config.php`. The module name is also used as name for the database table. Then `SetFields()` is called, which adds the fields to the module. This method is usually extended in the derived module classes.

The module name should represent a hierarchy. For instance the category module of CContent should have the class CContentCategory and the name `content_category`. This name is used to create the path for related files of the module. The images of the content category class are stored in `static/content/category/images` whereas other binary files are stored in `static/content/category/files`. Another path can be set, which is used e.g. by the newsletter tool to locate the various newsletter templates.

Since the number of binary files can get high and directory access may slow down, it is possible to set an upload level. By default, this level is one, which means the images are stored in a sub directory of the image directory. The name of this sub directory is created with the first letter of the the filename (which is a random MD5[81] value).

Modules can be virtually derived from standard modules (using the factory) like the above mentioned content category. For this classes, it is necessary to set the class name (e.g. CContentCategory), since the class itself would only report the name of the real class (CCategory). If two content classes CContent1 and CContent2 are used, and both classes share the same categories, a class group name can be set to CContent. So CContentCategory gets used by both classes (this is used in the administration part of oPage).

A module can have a number of sub modules (which are normal modules). This can be used to resolve 1:n and n:m relations. A newspaper article can be stored as an article with a number of paragraphs and each paragraph can have a number of pictures. First the class names of the sub modules are added with `AddSub()`. If the flag `$fSub` is set to true, the sub modules are created and added with `AddModul()` automatically when needed.

With `SetLog()` the logging flag can be turned on. If a global instance `$oLog` of the module CLog exists, every data modification action like insert, update or delete is recored in the log. By default logging is deactivated.

The trace parameter defines if the activities of the module and the required time have to be reported. Output can be sent to the browser, invisible as html comment or to the local syslog service of the server. By default tracing is disabled.

If the debug flag is set, the database queries and the returned data is displayed in the browser. This is useful when developing a Web site. When the explain flag is set, the database class sends each query with an EXPLAIN request to the database and displays detailed information, how the DBMS will execute the query.

All described settings can be adjusted differently in derived module classes, in the global and in the local factory as well as in the page scripts.

3.8.2 Table

A global prefix for table names can be defined in `custom/config.php` to separate oPage specific tables from other tables in the same database. This prefix is set with `SetTablePrefix( $sValue )` in the constructor. The name of the table itself is also set in the constructor with `SetTable( $sValue )`, where `$sValue` is filled with the name of the module.

With `AddTable( $sValue )` additional tables can be set and with `AddJoin( $sTable, $sJoin, $sOn )` the appropriate join parameters can be defined. Please note, that other ways of combining two tables are available: You can query data and set it with `SetReplacement()` (used to solve n:1 relations). Another

[81] MD5 is a method for creating hash values, containing 32 hexadecimal characters like `ffb2b2b0a963555a005cb1bd447ad44a`

option is to add submoduls with `AddSub( $sModul, $sId, $sForeignId )` to create a hierarchy of modules (to solve 1:n or m:n relations).

The complete table statement is generated by `GetTable( $iTable )`, where `$iTable` specifies either all tables (0) or a specific table (>0).

3.8.3 Fields

The base constructor calls the `SetFields()` method which adds the module fields by calling `SetField( $sName, $sType = Null, $fMultilanguage = Null, $sField = Null )`. The parameter `$sName` is the internal name of the field, as it will be used in the templates. With the parameter `$sType` the field type can be specified. Only types as defined in `SetFieldType()` are allowed. Example types are `TEXT` (which is the default type), `ID`, `FOREIGNID`, `FOREIGNNAME`, `TEXTBOX`, `IP`, `CHECKBOX`, `RADIO`, `YESNO`, `DISPLAY`, `USERNAME`, `PASSWORD`, `EMAIL`, `DATETIME`, `IMAGE`, `FILE`, `REAL`, `INTEGER` and a couple more. By default, `$fMultilanguage` is `false`. If it is `true`, the database field name is extended with the language postfix (as defined in `custom/config.php`). `$sField` specifies the name of the field as it is in the database table and can be omitted, if it is the same as the internal name.

The method in the base class adds fields like username, IP address and timestamps to track when the record has been created and when it has been last modified.

Similar to the table prefix, a global field prefix can be set, which is automatically prepended to each field. This is needed when additional tables are joined with the modules main table.

With `SetFieldUse( $sName, false )` a field can be deactivated completely. Some (optional) fields are added in `SetField()` but are deactivated in the constructor. Such fields can be activated with a call to `SetFieldUse( $sName )`. Other `SetFieldUse()` functions exist to control usage in admin list and detail views.

The complete field list statement is generated by `GetFields( $aFields = Null )`. If the parameter `$aFields` is an array, only the field names in the array are used, otherwise all active fields are used. A list of all available field names is returned by `GetFieldList()`.

3.8.4 Query

The method `QueryData()` is used to retrieve the content managed by the module from the database. It collects all the information needed and passes it to the database class, which creates the appropriate SQL statement and sends it to the DBMS. The queried records are returned to the module and stored in a two-dimensional array.

`QueryData()` prepares all elements of an SQL query:

- table (from, join)
- fields (columns)
- filter (where)
- groups (group by)
- order (order by)
- limit (limit)

`GetTable()` returns a string with all involved tables (at least one) and the matching join operators. `GetFields()` returns a list of active names (see section 3.8.3 for details) including the internal field name and the column name as it is in the table (if they are different). Analogous functions exist for filter, groups, order and limit.

If needed, uniqe rows can be forced by calling `SetDistinct()`.

With the `SetFilter()` method, a WHERE clause with elements like values and operators can be set for each field. Filter entries are combined with the `and` operator. An array of filter entries can be provided as parameter, together with compare operators (e.g. `=`, `<>` or `like`) and logical combine operators (`and` or `or`), which again can contain an array of filters.

The following source code shows an example for a complex filter:

```
1 $aFilter = array();
2 $aFilter[] = array( 'value' => 'y', 'compare' => '<>' );
3 $aFilter[] = array( 'compare' => 'is_null', 'escape' => false, 'slashes' => false );
4 $this->SetFilter( 'preview', $aFilter );
```

Similar to the filters, the group and order by clauses can be set. For group by, only the field name has to be set with `SetGroupBy()`, and the result can be retrieved with `GetGroupBys()`. The appropriate methods for order by are `SetOrder()` (with an optional parameter `desc`) and `GetOrders()`. These functions also check, if a field is used at all and only return needed field names.

Since it is not practical to display hundreds of records on one page, oPage offers ways to devide the result set into parts. `SetLimit()` sets an offset from where to start and the maximum number of rows to be returned. The `GetLimit()` method returns the appropriate SQL statement, which produces the limited result set. With `SetRows()` the number of rows per page can be set. The current page is set with `SetPage()` and can be retrieved from a pager control, which provides the user interface in combination with a template. `GetPage()` returns the current page and `GetPages()` returns the total number of available pages. If records are displayed in more than one column, the number of columns can be set with `SetCols()`.

```
1  // 10 rows per page
2  $oModul->SetRows( 10 );
3  // 3 columns per row
4  $oModul->SetCols( 3 );
5  // but never the first 5 records and only a maximum of 50 records
6  $oModul->SetLimit( 5, 50 );
7  // set the active page to 2
8  $oModul->SetPage( 2 );
9  // read the data
10 $oModul->QueryData();
11 // output the total number of pages: 2
12 echo $oModul->GetPages();
13 // output current page: 2
14 echo $oModul->GetPage();
15 // output number of records: 20
16 echo $oModul->CountRows();
```

In the above example, the records 36 (1 + 5 + 10 rows times 3 columns) to 65 (36 + 10 rows times 3 columns - 1) would be read. Because of the limit set, only the records 36 to 55 are queried.

3.8.5 Output

A number of output methods exist, but only `Output()` and `OutputTpl()` are intended to be called directly. `Output()` receives the filename of a template, creates a template object and forwards it to `OutputTpl()`.

`OutputTpl()` requires a certain block structure to merge the result set with the template. For a sample template see listing 10 on page 65, for a full description see listing 33 on page 135. In brief, `OutputTpl()` parses the module blocks `name of the modul#header`, `name of the modul` and `name of the modul#footer`. Beneath these blocks module blocks like `empty`, `HEADER`, `DATA` or `FOOTER` can be used to represent the desired layout. If the current `DATA` block contains a `BETWEEN` block, this block is parsed after each record except for the last record.

If the result set is empty, only the `empty` block is parsed. For the first available record, the `HEADER` block is parsed, then for each row the `DATA` block or, if a row specific `DATA{iRow}`[82] block exists, this block is used. The rows, which should be processed can be limited with `SetOutputRowFrom()` and `SetOutputRowTo()`.

[82] replace `{iRow}` with a one based row number

Independent from the database query, the rows in the result set can be grouped with `SetGroup()`. For each field, an optional array of possible values can be provided, so only rows containing those values are merged with the template, which has to provide a block named according to the value. If this array contains the value `_default_`, all rows with other values are merged against the `_default_` block. For each group an optional array of values can be used to create a combined one. E.g. `aCombine[ 'A' ] = array( 'A', 'Ä' );` assigns the letters `'A'` and `'Ä'` to the same group `'A'`. For each group value a header and a footer block are parsed: `GROUP_{group}_begin` and `GROUP_{group}_end`.

If the result set contains a numeric column named `level`, a block for each level can be used beneath the `DATA` block. If a maximum level is set with `SetLevels()`, a level block between `LEVEL1` and `LEVEL{iLevels}`is parsed. The field `level` can be retrieved from a database column or a calculated in `PrepareData()`.

If a section number higher then one is set with `SetSections()`, for each section a `ROW{iSection}` is parsed. The `iSection` is one based and is calculated using the row number: `$iSection = ( $iRow - 1 ) % $iSections + 1;`. If two sections are used, the template blocks `ROW1` and `ROW2` are parsed alternately.

`SetCols()` define, how many records are displayed in one row. If this number is higher than one, a `COL` block for each column is required. listing 11 illustrates a table with three columns.

Listing 11: *table.tpl sample*

```
<!-- BEGIN: main -->

    <!-- BEGIN: modul -->

        <!-- BEGIN: HEADER -->
            <table>
            <thead>
            <tr>
                <th>Column1</th>
                <th>Column2</th>
                <th>Column3</th>
            </tr>
            </thead>
            <tbody>
        <!-- END: HEADER -->

        <!-- BEGIN: DATA -->

            <!-- BEGIN: COL1 -->
            <tr>
                <td>{DATA.field}</td>
            <!-- END: COL1 -->

            <!-- BEGIN: COL2 -->
                <td>{DATA.field}</td>
            <!-- END: COL2 -->
            <!-- BEGIN: COL2_empty -->
                <td> </td>
            <!-- END: COL2_empty -->

            <!-- BEGIN: COL3 -->
                <td>{DATA.field}</td>
            </tr>
            <!-- END: COL3 -->
            <!-- BEGIN: COL3_empty -->
                <td> </td>
            </tr>
```

```
            <!-- END: COL3_empty -->

        <!-- END: DATA -->

        <!-- BEGIN: FOOTER -->
            </tbody>
            </table>
        <!-- END: FOOTER -->

    <!-- End: modul -->

<!-- END: main -->
```

If the result set contains the field `template` it is supposed that it contains the name of a template block. The value of the `template` field is extended with the name of the module. For example field `template` of the module `content` can contain one of the following values: `heading1`, `heading2` or `heading3`. Additionally to the main block a `#begin` and `#end` block is parsed.

The following listing provides an example:

```
<!-- BEGIN: DATA -->
    <!-- BEGIN: content_heading1 -->
    <h1>{DATA.title}</h1>
    <!-- END: content_heading1 -->
    <!-- BEGIN: content_heading2 -->
    <h2>{DATA.title}</h2>
    <!-- END: content_heading2 -->
    <!-- BEGIN: content_heading3 -->
    <h3>{DATA.title}</h3>
    <!-- END: content_heading3 -->
    <p>{DATA.text}</p>
<!-- END: DATA -->
```

For each field a number of blocks can be used. `field#empty` (replace `field` with the name of a field) is parsed if the field has no value. All other blocks are only parsed if the field contains a value. Value dependent blocks are possible with `field=value` where value has to be replaced with a specific value.

If the fields value is not 0, the block `field<>0` is parsed, if it is not 1, then `field<>1` is parsed. The `field#yes` and `field#no` blocks are only parsed for `YESNO` type fields.

```
<!-- BEGIN: DATA -->
    <!-- BEGIN: field#empty --><!-- END: field#empty -->

    <!-- BEGIN: field#header --><!-- END: field#header -->
    <!-- BEGIN: field#begin --><!-- END: field#begin -->

    <!-- BEGIN: field=value#begin --><!-- END: field=value#begin -->
    <!-- BEGIN: field=value --><!-- END: field=value -->
    <!-- BEGIN: field=value#end --><!-- END: field=value#end -->

    <!-- BEGIN: field<>0#begin --><!-- END: field<>0#begin -->
    <!-- BEGIN: field<>0 --><!-- END: field<>0 -->
    <!-- BEGIN: field<>0#end --><!-- END: field<>0#end -->

    <!-- BEGIN: field<>1 --><!-- END: field<>1 -->

    <!-- BEGIN: field --><!-- END: field -->

```

```
19      <!-- BEGIN: field#end --><!-- END: field#end -->
20      <!-- BEGIN: field#footer --><!-- END: field#footer -->
21
22      <!-- BEGIN: field#notempty --><!-- END: field#notempty -->
23
24      <!-- BEGIN: field#yes#begin --><!-- END: field#yes#begin -->
25      <!-- BEGIN: field#yes --><!-- END: field#yes -->
26      <!-- BEGIN: field#yes#end --><!-- END: field#yes#end -->
27
28      <!-- BEGIN: field#no#begin --><!-- END: field#no#begin -->
29      <!-- BEGIN: field#no --><!-- END: field#no -->
30      <!-- BEGIN: field#no#end --><!-- END: field#no#end -->
31  <!-- END: DATA -->
```

3.8.6 Prepare

For each field a format can be defined using `SetFormat( $sName, $Value )`, where `$sName` is the internal name of the field and `$Value` is the format. `$Value` can either be a string (for `TEXT` fields) or an integer (for `DATE` and `NUMBER` fields).

Automatic replacing of URLs with clickable hyperlinks can be globally turned on for all TEXT fields with `SetTextPrepareUrls()` or for an individual field with `SetPrepareUrls( $sName )`. Of course it is possible to turn it on for all fields and turn it off for specific fields. Usually a module is supposed to produce HTML output, with `SetTextMode()` all HTML tags get removed from the `TEXT` fields.

Categories to group records can be managed with the module CContentCategory. All IDs and category titles can be queried from this module and assigned to the CContent module with `SetReplacement( $sName, $aValue )`, where `$sName` is the name of the `FOREIGNID` field.

In the method `OutputPrepare( &$aRow )` the current row gets prepared. This method calls `Prepare Data1( $aRow )`, `PrepareReplace( $aRow )`, `PrepareData( $aRow )` and `PrepareRow( $aRow )`. Each of this function can be overrided in the derived module classes. See listing 8 on page 60 for an example of `PrepareRow()`. There the methods `PrepareText()`, `PrepareImage()` and `PrepareDate()` are used. Similar functions are `PrepareFile()` and `PrepareNumber()`. Others exist to calculate net and gross prices (`PreparePrice()`), to modify `URL` addresses (`PrepareUrl()`) or to update creation and modification timestamps (`PrepareRowEdit()`).

3.8.7 Database access

In the typical usage of oPage, it is not necessary to access the database tables directly. The modules base class provides methods to create, read, update and delete records.

The `Insert()` method creates a new record using data provided in a name/value array. The ID of the inserted row is returned in the array.

To read records from the modules table into an array, the `Query()` method can be used, which has only one parameter for a WHERE clause and returns all active fields from the table. The `Query2()` method has an additional parameter for the names of the fields, which should be read.

The `Update()` method is similar to the insert method, but has an additional (optional) parameter for the name of the ID field (which is `id` by default). The update method also takes care of binary files stored in the file system. When an IMAGE or FILE field is updated with a new value, the old file is deleted from the file system.

To delete records two methods are available. The `Delete()` method uses the filter parameter for the where clause of the delete SQL statement. The extended method `DeleteX()` is similar to the `Insert()` and `Update()` methods and uses a name/value array and an optional name for the ID field. Both methods delete the related binary files from the file system.

3.8.8 Sub modules (1:n relations)

In many cases content needs to be distributed over several database tables. For example a newspaper article can have several paragraphs and each paragraph can be embellished with pictures. Each article can be assigned to a number of categories. This results in five database tables:

- article
- paragraph
- picture
- category
- category_article

Here we have a 1:n relation between article and paragraph and between paragraph and picture. Between article and category there is a m:n relation, which is stored in the table category_article.

To represent these tables, five modules are used, one for each table:

- CArticle
- CArticleParagraph (derived from CParagraph)
- CArticleParagraphPicture (derived from CPicture)
- CArticleCategory (derived from CCategory)
- CArticleCategoryArticle (derived from CRelation)

Modules can be added to other modules as sub modules with `AddModul()`[83]:

```
1 $oPicture = $oFactory->Modul( 'CArticleParagraphPicture' );
2 $oParagraph = $oFactoy->Modul( 'CArticleParagraph' );
3 $oParagraph->AddModul( $oPicture, 'id', 'paragraph_id' );
4 $oArticle = $oFactory->Modul( 'CArticle' );
5 $oArticle->AddModul( $oParagraph, 'id', 'article_id' );
```

Now, when a article is merged with the template, all paragraphs are queried where the `article_id` is equal to the `id` of the article. And for each paragraph, the pictures are loaded where `paragraph_id` is equal to the `id` of the paragraph.

In the template the paragraph block has to be within the data block of the article.

```
1  <!-- BEGIN: main -->
2
3  <!-- BEGIN: article -->
4      <!-- BEGIN: DATA -->
5          <h1>{DATA.title}</h1>
6
7          <!-- BEGIN: paragraph -->
8              <!-- BEGIN: DATA -->
9              <p>{DATA.text}</p>
10
11                 <!-- BEGIN: picture -->
12                     <!-- BEGIN: HEADER -->
13                     <div>
14                     <!-- END: HEADER -->
15                     <!-- BEGIN: DATA -->
16                     <img src="{global.root}{DATA.image_small}" width="{DATA.image_small#
                           width}" height="{DATA.image_small#height}">
17                     <!-- END: DATA -->
18                     <!-- BEGIN: FOOTER -->
19                     </div>
```

[83] A good place to set sub modules is in the `custom/factory.php`, where all other configuration takes place.

```
                    <!-- END: FOOTER -->
                <!-- END: picture -->

            <!-- END: DATA -->
        <!-- END: paragraph -->

    <!-- END: DATA -->
<!-- END: article -->

<!-- END: main -->
```

An extended method is `AddSub()`. It adds the module as a sub module (by calling `AddModul()`), but also offers parameters to define, if the sub module should be used in the frontend, in the backend admin listview or detail view. If it is used m:n relations should be added with `AddRelation()`, which adds the module as sub module by calling `AddSub()` and also creates a form field in the admin to manage the entries in the relation table.

Figure 18 on the following page provides a screenshot of an article with two paragraphs each having two pictures. A more detailed description of the administration forms can be found in section 3.16.1 on page 85.

3.9 CPage

The content of a Web site is usually grouped into different sections. These can be news, product information, links or a guestbook. In oPage this content is managed in modules.

The module page is used to manage page entries. Each page has a unique name, a path, a target frame and four kinds of titles. The field `title_display` (e.g. `Homepage`) is used for internal purposes and is not displayed on the Web site. In the field `title` the official title of the page (e.g. `Welcome to my Web site`) is stored. The values in `title_navigation` and `title_menu` are used in horizontal (`title_navigation`) and vertical (`title_menu`) navigational controls (for more details see section 3.12 on page 75).

The path value of a page entry points to a PHP script, which creates the output of a Web page. This script is a controller[84] and can combine a number of modules. For example on the entry page of a Web site (the homepage) the top five entries of the news module are displayed along with two upcoming events and three featured products of the shop section (see also listing 1 on page 37).

3.10 CWebpage

In this context the CWebpage class is responsible for creating the surrounding environment for the other controls and modules (see figure 13 on page 36). The layout of the Web page is stored in a template which can be merged with the output of the modules and controls (which can use additional templates).

This class is typically used as follows:

```
$oPage = $oFactory->Webpage( 'home', 'CWebpageX' );
$oPage->Init();
$oPage->Begin();

$sContent = 'my content';

$oPage->Assign( 'content', $sContent );
$oPage->End();
```

The class CWebpageX is a derived class which extends the base class with project specific code. For example, a default page template is set with `SetTemplate()` in CWebpageX. In the initialization process the

[84] Model-View-Controller pattern

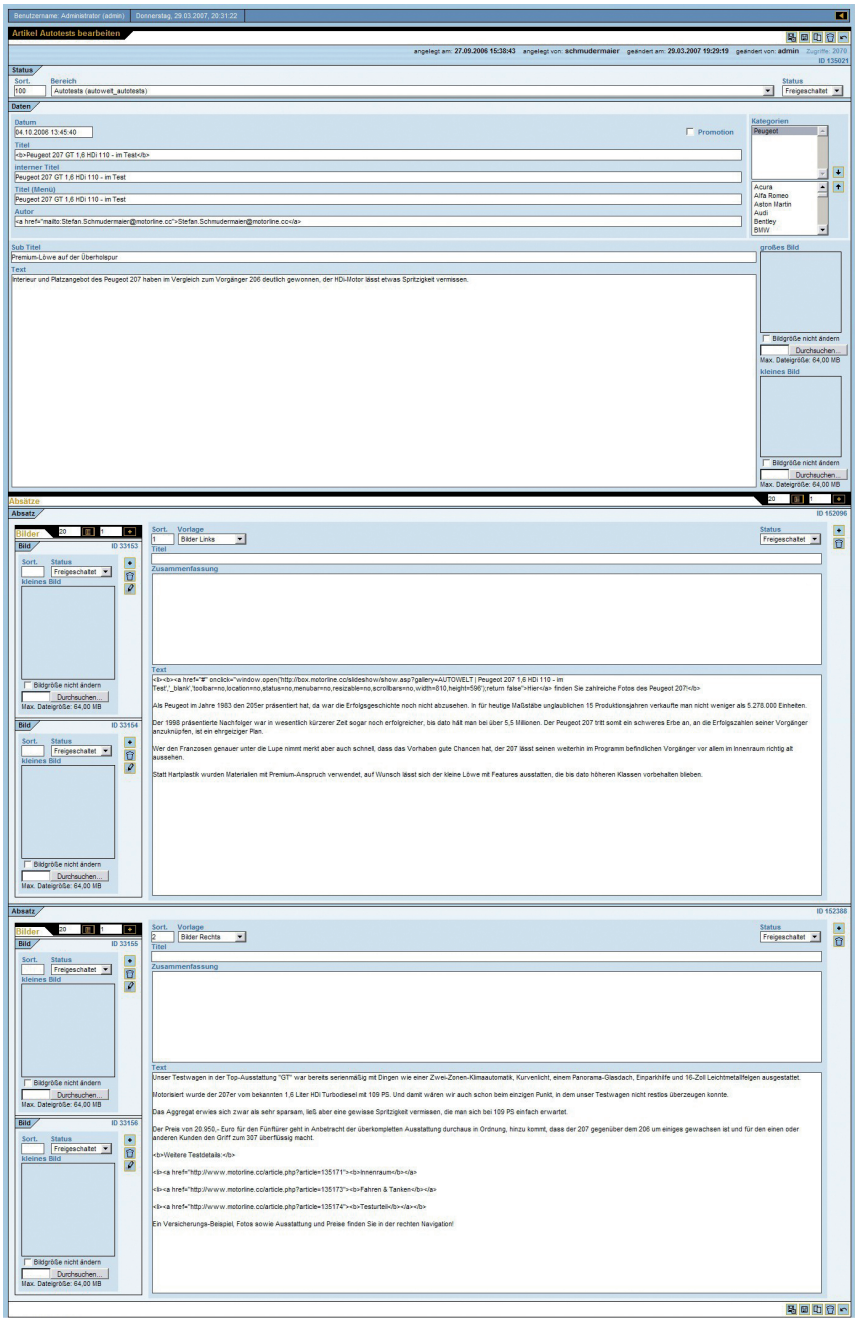


Figure 18: *Backend administration form for an article (detail view)*

data of the page named `home` are loaded (using the module CPage). With `Begin()` the template is loaded and parameters (see table 4) are assigned to the template (and are also passed along to the modules). In the `Begin()` method of the project specific class, the navigation controls can be created and filled with data. By calling `End()` the custom class calls the created navigation controls and the base class calls the template engine to prepare the final page. The result can be compressed, written to the cache or just sent to the client.

Table 4: *template parameters provided by CPageWeb*

parameter	description
`{query}`	Array with all parameter values sent with the query (e.g. ?id=1 becomes query.id with value 1)

continued on next page

Table 4: continued

variable	description
{global.backuri}	The URL of the current page including all relevant parameters; can be used as backlink parameter in links to other pages
{global.backurix}	The same as {global.backuri} but with special search engine optimized parameter format (basically it uses / instead of ? and &).
{global.printuri}	This variable has the same value as {global.backuri}, but with an additional parameter print=1 and is used to link to the printer friendly version of a page.
{global.printurix}	The search engine optimized version of {global.printuri}.
{page}	Array with all fields from the module page.
{page.robots}	If the parameter print is set to 1, this value contains noindex,nofollow to exclude the printer friendly version from being indexed, or index,follow otherwise.

3.11 CControl

CControl is the base class for controls. Similar to CModul a control always has a name which must relate to a block in the template. In the constructor GetParameter() is called, which can be overrided in the drived class, and is used to prepare the parameters for the control.

The method Output() loads the template and calls OutputTpl(), which merges the data of the control with the prepared template. Usually only the block name of the control is parsed, but if the blocks name of the control#header or name of the control#footer exist, the are parsed as well.

OutputValue() is not used in the base class, but can be called by the derived controls to parse blocks for every values.

```
<!-- BEGIN: someblock -->
  <!-- BEGIN: field#empty --><!-- END: field#empty -->

  <!-- BEGIN: field#header --><!-- END: field#header -->
  <!-- BEGIN: field#begin --><!-- END: field#begin -->

  <!-- BEGIN: field=value --><!-- END: field=value -->

  <!-- BEGIN: field<>0#begin --><!-- END: field<>0#begin -->
  <!-- BEGIN: field<>0 --><!-- END: field<>0 -->
  <!-- BEGIN: field<>0#end --><!-- END: field<>0#end -->

  <!-- BEGIN: field --><!-- END: field -->

  <!-- BEGIN: field#end --><!-- END: field#end -->
  <!-- BEGIN: field#footer --><!-- END: field#footer -->

  <!-- BEGIN: field#notempty --><!-- END: field#notempty -->
<!-- END: someblock -->
```

3.12 CNavigation

The control CNavigation is used to create navigational links.

One way to add entries is calling the Add() method directly. The usual way is to pass an instance of CNavigation to the AddNavigation() method of the Webpage() class, which loads data from the page module and calls the Add() method of CNavigation. The control currently supports two types: navigation (which is the default) and menu. Depending on the type set, either the page field title_navigation or title_menu are used.

With the factory method Control() both CNavigation and the virtual class CMenu can be created. While CNavigation uses `title_navigation`, CMenu uses `title_menu`.

The navigation control parses a number of blocks to offer a wide range of possible navigational elements:

```
<!-- BEGIN: control -->

    <!-- BEGIN: empty -->
    <!-- END: empty -->

    <!-- BEGIN: HEADER -->
    <!-- END: HEADER -->

    <!-- BEGIN: DATA -->

        <!-- BEGIN: FIRST -->
        <!-- END: FIRST -->

        <!-- BEGIN: BETWEEN -->
        <!-- END: BETWEEN -->

        <!-- BEGIN: LAST -->
        <!-- END: LAST -->

    <!-- END: DATA -->

    <!-- BEGIN: FOOTER -->
    <!-- END: FOOTER -->

<!-- END: control -->
```

Within `DATA.FIRST`, `DATA.BETWEEN`, `DATA.LAST` and `DATA` the following blocks are parsed:

```
  <!-- BEGIN: path#empty -->
  <!-- END: path#empty -->
  <!-- BEGIN: path -->
  <!-- END: path -->

  <!-- BEGIN: parameter#empty -->
  <!-- END: parameter#empty -->
  <!-- BEGIN: parameter#begin -->
  <!-- END: parameter#begin -->
  <!-- BEGIN: parameter -->
  <!-- END: parameter -->
  <!-- BEGIN: parameter#end -->
  <!-- END: parameter#end -->

  <!-- BEGIN: target#empty -->
  <!-- END: target#empty -->
  <!-- BEGIN: target -->
  <!-- END: target -->

  <!-- BEGIN: title#empty -->
  <!-- END: title#empty -->
  <!-- BEGIN: title -->
  <!-- END: title -->

```

```
<!-- BEGIN: isnew#empty -->
<!-- END: isnew#empty -->
<!-- BEGIN: isnew#yes -->
<!-- END: isnew#yes -->
<!-- BEGIN: isnew#no -->
<!-- END: isnew#no -->

<!-- BEGIN: onclick#empty -->
<!-- END: onclick#empty -->
<!-- BEGIN: onclick -->
<!-- END: onclick -->

<!-- BEGIN: sub_empty -->
<!-- END: sub_empty -->

<!-- BEGIN: sub_notempty -->
<!-- END: sub_notempty -->

<!-- BEGIN: sub_expanded -->
<!-- END: sub_expanded -->

<!-- BEGIN: sub_collapsed -->
<!-- END: sub_collapsed -->

<!-- BEGIN: sub -->
    <!-- BEGIN: DATA -->
        <!-- BEGIN: FIRST -->
        <!-- END: FIRST -->
        <!-- BEGIN: BETWEEN -->
        <!-- END: BETWEEN -->
        <!-- BEGIN: LAST -->
        <!-- END: LAST -->
    <!-- END: DATA -->
<!-- END: sub -->

<!-- BEGIN: sub_always -->
    <!-- BEGIN: DATA -->
        <!-- BEGIN: FIRST -->
        <!-- END: FIRST -->
        <!-- BEGIN: LAST -->
        <!-- END: LAST -->
    <!-- END: DATA -->
<!-- END: sub_always -->

<!-- BEGIN: separator -->
<!-- END: separator -->

<!-- BEGIN: break -->
<!-- END: break -->
```

Below the blocks `active` and `inactive` the full range of blocks described above can be used.

```
  <!-- BEGIN: active -->
  <!-- END: active -->
  <!-- BEGIN: inactive -->
  <!-- END: inactive -->
```

For each entry, an entry specific subblock is parsed (with all blocks described above), if available. If no entry specific block is available, a `_default_` block is parsed (with all blocks described here), if available.

```
    <!-- BEGIN: name -->
    <!-- END: name -->
    <!-- BEGIN: _default_ -->
    <!-- END: _default_ -->
```

Example navigational controls are horizontal or vertical link lists, pulldown menus or tree controls (or whatever comes up your mind).

3.12.1 CWebpage->AddNavigation()

The class CWebpage provides the method `AddNavigation()`, which adds an entry from the module CPage to a navigation control. The instance of the control and the name of the page have to be provided. To create a hierarchy (e.g. for a tree control), the name of the parent entry can be provided. Possible additional parameters are a query string and the height as well as the width of the entry.

3.12.2 CModul->FillNavigation()

This method queries data from the modules table and adds it to the navigation control. Optional an URL and a target frame name can be provided. If necessary, the name of a parent navigation entry can be supplied. By default, the field `title` is used as title, but it can be changed by providing a different fieldname. Filters can be set, but this filters are not used by default.

3.13 CMail

Sending email through a Web site is not only used by Web mail sites. A Web site should inform about new guestbook and forum entries or deliver new orders from the online shop. Another application is sending a newsletter or providing an interface to send ecards.

In oPage emails are sent with PHPMailer[85], which is a commonly used class to send emails via the PHP `mail()` function, sendmail or a SMTP server.

The PHPMailer is integrated into oPage by the CMail control class (using the facade[86] design pattern). An instance of the email class is usually created with the factory method `Control()`. The type of mail server can be set in `custom/config.php`. Important CMail methods (beside the obvious `SetFrom()`, `SetTo()`, `SetCc()`, `SetBcc()`, `SetSubject()` and `AddAttachment()`) are `SetValues()`, `Send()` and `SendTpl()`.

With `SetValues()` an array of name-value pairs are set, which will be later assignd to the template engine and merged with the email templates. `Send()` accepts up to two filenames of templates. The first one is mandatory and can either be a HTML or a plain text template (you have to call `SetHtml()` to tell the class). The second one is optional (but recommended, if the first template is a HTML template) and can only be a plain text template. `Send()` creates an instance of the template engine for each template and passes this objects to `SendTpl()`.

```
<!-- BEGIN: subject -->This is the subject<!-- END: subject -->
<!-- BEGIN: recipients -->hannes@dorn.cc|Hannes Dorn;office@dorn.cc<!-- END: recipients -->
<!-- BEGIN: main --><!-- BEGIN: mail -->
Firstname: {DATA.firstname}
Lastname: {DATA.lastname}
Text:
{DATA.text}
<!-- END: mail --><!-- END: main -->
```

[85] Originally written by Brent R. Matzelle and released under LGPL. See `http://phpmailer.sourceforge.net/`

[86] `http://en.wikipedia.org/wiki/Facade_pattern`

If no specific subject has been set with `SetSubject()`, `SendTpl()` parsed the subject block of the template by calling `OutputTpl()`. The blocks `from`, `recipients`, `recipients_cc` and `recipients_bcc` are parsed using `OutputTpl()` (this can be turned of by calling `SetFromFromTemplate()` and `Set RecipientsFromTemplate()`). If a from value has been set, it gets overwritten while the recipient addresses from the template are just appended to the existing recipient lists. Then the main block is parsed to produce the body for the email. If PGP encryption is enabled by `SetPgp()`, the body content is encrypted with through PGP before it is sent with PHPMailer.

`OutputTpl()` is similar to the same named method of CModul. First the `OutputTpl()` method of the parent class is called (see section 3.11 on page 75), then the controls values are assigned to the template engine and the template blocks are parsed.

3.13.1 CModule Mail

Consider the following scenario: An email should be sent if a new guestbook entry is made. The email should contain the newly added data. In the common way, the data from the form would be assigned to the mail control. A simplier way is to assign the guestbook module to the mail control with `SetModul()`.

In `SendTpl()` the from email address and the recipient (if not already set) are queried from the module (`GetEmailFrom()`, `GetEmailAdmin()`). `OutputTpl()` calls the `OutputTpl()` method of the module. All module blocks (see section 3.8 on page 64) can be used below the mail blocks `from`, `recipients`, `recipients_cc`, `recipients_bcc` and `mail`

```
1  <!-- BEGIN: subject -->Guestbook entry: <!-- BEGIN: mail --><!-- BEGIN: guestbook --><!--
       BEGIN: DATA -->{DATA.title}<!-- END: DATA --><!-- END: guestbook --><!-- END: mail -->
       <!-- END: subject -->
2  <!-- BEGIN: recipients --><!-- BEGIN: mail --><!-- BEGIN: guestbook --><!-- BEGIN: DATA -->{
       DATA.email}<!-- END: DATA --><!-- END: guestbook --><!-- END: mail --><!-- END:
       recipients -->
3  <!-- BEGIN: recipients_bcc -->office@dorn.cc<!-- END: recipients_bcc -->
4  <!-- BEGIN: main --><!-- BEGIN: mail --><!-- BEGIN: guestbook --><!-- BEGIN: DATA -->
5  <!-- BEGIN: firstname -->Firstname: {DATA.firstname}
6  <!-- END: firstname --><!-- BEGIN: lastname -->Lastname: {DATA.lastname}
7  <!-- END: lastname --><!-- BEGIN: text -->Text:
8  {DATA.text}
9  <!-- END: text -->
10 {DATA.date_created} {DATA.ip_created}<!-- END: DATA --><!-- END: guestbook --><!-- END: mail
       --><!-- END: main -->
```

As seen in the listing above, the field `title` of the guestbook module is used in the subject of the email. The `email` field is used as recipient and in the `main` block the guestbook module is used to produce the body of the email.

In CModul methods exist for sending the email: `MailToAdmin()` and `MailToUser()`.

If an action name (`add`, `confirm`, `update` or `delete`) is provided, `MailToAdmin()` checks, if an email should be sent (parameter entry `{name of the module}_emailtoadmin_{name of the action}`). Then an instance of CMail is created, the module is set with `SetModul()`, and the control is passed along to `MailSend()`, where the `Send()` method is called.

`MailToUser()` is similar to `MailToAdmin()`. If an action name (`add`, `confirm`, `update`, `delete`) is provided, the system looks into the parameter module, if an email should be sent (parameter `{name of the module}_emailtouser_{name of the action}`). Then it checks, if the field `email` is used and has a value. A display name for the recipient is created using the fields `firstname`, `lastname` and `company`. If none are used or all are empty, the email address is used as display name. An instance of CMail is created and the module and the recipient are set. Then the control is passed along to the `MailSend()` method, where the `Send()` method of the mail control is called.

The module is supposed to contain records already. Multiple records will result in multiple emails sent.

`MailSend()` is rather simple, but can be overruled in a derived module class. It is called with an instance of CMail, the filename of a template, the mode (possible values are `MailToAdmin`, `MailToUser` and `Null`) and optionally the name of the current action.

3.14 CForm

oPage provides a versatile form engine. The class CForm is derived from CControl and processes input types like plain and multiline text, date, number and currency, yes/no, tristate and radio buttons as well as images and other binary uploads. More specialized types exist like a country selector, an email address verifier or a creditcard validator. With a CAPTCHA field the form engine can check, that a human user has submitted the form.

The provided data is validated according to the field types. For each field, an error block in the template is parsed if a required field is empty or the content is invalid. For example, yes/no fields can only have the values `'y'` or `'n'`. If an invalid value is provided, the form control changes the value to `'n'`. If a field should contain an URL address, the validate method uses regular expressions to check, if the provided string looks like a valid address.

Date values are converted from the users native date format to the database format `'yyyy-mm-dd'`. Similar conversions are done for time and number values.

Uploaded images can be automatically resized. On Web sites, only GIF, JPEG or PNG images should be used, if the provided image is in a different format it gets automatically converted into a JPEG file.

oPage provides template fragments to build plain and tabular forms. The provided example shows the PHP script (listing 12) and the according template (listing 13 on the next page) for adding entries to a guestbook.

Listing 12: *Guestbook add script*

```php
<?php

// include project settings and base classes
include_once '../include/opage.php';

// Create the page object
$oPage = $oFactory->WebPage( 'guestbook_add', 'CWebPageX' );
$oPage->Init();
$oPage->Begin();

// Create the template object
$oTemplate = $oFactory->Template( 'add.tpl' );

// Create the module and activate the fields, which should be used in the form
$oModul = $oFactory->Modul( 'CGuestbook' );
$oModul->SetFieldUseFormMust( 'firstname' );
$oModul->SetFieldUseForm( 'lastname' );
$oModul->SetFieldUseForm( 'city' );
$oModul->SetFieldUseForm( 'region' );
$oModul->SetFieldUseForm( 'country' );
$oModul->SetFieldUseFormMust( 'email' );
$oModul->SetFieldUseForm( 'url' );
$oModul->SetFieldUseForm( 'title' );
$oModul->SetFieldUseFormMust( 'text' );

// Create form
$oForm = $oFactory->Control( 'CForm', 'guestbook' );
$oModul->AddForm( $oForm );

```

```
30 // Add an additional field: captcha
31 // This field is not used by the module, but helps to prevent automatic guestbook spam
32 $oForm->AddField( 'captcha', 'CAPTCHA' );
33
34 // Process form
35 if ( !$oForm->IsSubmitted() )
36     $oForm->OutputTpl( $oTemplate );
37 elseif ( !$oForm->Validate() )
38     $oForm->OutputTpl( $oTemplate );
39 elseif ( !$oModul->AddAction( $oForm, $oTemplate, 'email.tpl', 'email_user.tpl' ) )
40     $oForm->OutputTpl( $oTemplate );
41 else
42 {
43     // Redirect the browser to the main guestbook page
44     Header( 'Location: ' . $oApp->GetRoot() . $oApp->GetDir() . 'index.php' );
45     exit();
46 }
47
48 // Insert content into main template
49 $oTemplate->Parse();
50 $oPage->Assign( 'content', $oTemplate->Text() );
51
52 // Finalize and output the generated page
53 $oPage->End();
54
55 ?>
```

Listing 13: *Guestbook add template*

```
1  <!-- BEGIN: main -->
2
3  <!-- BEGIN: submitted -->
4  <p>Thank you for your entry!</p>
5  <!-- END: submitted -->
6
7  <!-- BEGIN: error -->
8  <p>Ups, there is something wrong!</p>
9  <!-- END: error -->
10
11 <!-- BEGIN: guestbook -->
12
13     <p>Thank you for adding something to our guestbook. Please fill out the form and click ⤶
           save. Please note, that for security reasons, you can not write HTML code in our ⤶
           guestbook.</p>
14
15     {FILE "include/template/form/#begin.tpl" class="guestbook"}
16     {FILE "include/template/form/tabular/#begin.tpl" cellpadding="2" width="100%"}
17     <colgroup>
18         <col width="30%">
19         <col width="70%">
20     </colgroup>
21     {FILE "include/template/form/tabular/custom_begin.tpl"}
22         <!-- BEGIN: firstname#begin -->
23         {FILE "include/template/form/edit_label.tpl" name="firstname" title="Firstname"}
24         &amp;
25         <!-- END: firstname#begin -->
26         <!-- BEGIN: lastname#begin -->
```

```
27          {FILE "include/template/form/edit_label.tpl" name="lastname" title="Lastname:"}
28          <!-- END: lastname#begin -->
29      {FILE "include/template/form/tabular/custom_between.tpl"}
30          <!-- BEGIN: firstname -->
31          {FILE "include/template/form/edit_input.tpl" name="firstname" title="Firstname"
                width="20" style="width: 48%;"}
32          <!-- END: firstname -->
33          <!-- BEGIN: lastname -->
34          {FILE "include/template/form/edit_input.tpl" name="lastname" title="Lastname" width
                ="20" style="width: 48%;"}
35          <!-- END: lastname -->
36      {FILE "include/template/form/tabular/custom_end.tpl"}
37      {FILE "include/template/form/tabular/custom_begin.tpl"}
38          {FILE "include/template/form/edit_label.tpl" name="city" title="City"}
39          &amp;
40          {FILE "include/template/form/edit_label.tpl" name="region" title="Region:"}
41      {FILE "include/template/form/tabular/custom_between.tpl"}
42          {FILE "include/template/form/edit_input.tpl" name="city" title="City" width="20"
                style="width: 48%;"}
43          {FILE "include/template/form/edit_input.tpl" name="region" title="Region" width="20"
                 style="width: 48%;"}
44      {FILE "include/template/form/tabular/custom_end.tpl"}
45      {FILE "include/template/form/tabular/edit_select.tpl" name="country" title="Country:"
            style="width: 97%;"}
46      {FILE "include/template/form/tabular/edit_input.tpl" name="email" title="Email:" width
            ="50" style="width: 97%;"}
47      {FILE "include/template/form/tabular/edit_input.tpl" name="url" title="Url (without http
            ://):" width="50" style="width: 97%;"}
48      {FILE "include/template/form/tabular/edit_textarea.tpl" name="text" title="Your text:"
            width="60" height="10" style="width: 97%;"}
49      {FILE "include/template/form/tabular/edit_captcha.tpl" title="Captcha:" width="50"}
50      {FILE "include/template/form/tabular/edit_submit.tpl" title="save" width="50" style="
            width: 97%;"}
51      {FILE "include/template/form/tabular/#end.tpl"}
52      {FILE "include/template/form/#end.tpl"}
53
54  <!-- END: guestbook -->
55
56  <!-- END: main -->
```

3.14.1 CModul Form

The base class CModul provides a number of methods to simplify acquiring and processing of user input. For a full list of all form related methods have a look at listing 14.

Listing 14: *modul.php form methods*

```
1  Function AddAction( &$oForm, &$oTemplate, $sTplAdmin, $sTplUser, $fParse = null )
2  Function AddActionWithValues( $aValues, $sTplAdmin = null, $sTplUser = null )
3  Function AddActionGetValuesFromForm( $oForm )
4  Function AddActionPrepareValues( $aValues )
5  Function AddActionExists( $aValues )
6  Function AddActionInsert( &$aValues )
7  Function AddActionMailToAdmin( $sTemplate )
8  Function AddActionMailToUser( $sTemplate, $sField = null )
9
10 Function ConfirmCheck( &$oForm, &$oTemplate, $fParse = null )
```

```
11 Function ConfirmAction( &$oForm, &$oTemplate, $sTplAdminAdd, $sTplAdminUpdate, $fParse =
       null )
12 Function ConfirmActionGetValuesFromForm( $oForm )
13 Function ConfirmActionPrepareValues( $aValues )
14 Function ConfirmActionExists( $aValues )
15 Function ConfirmActionUpdate( &$aValues )
16 Function ConfirmActionMailToAdmin( $sTemplate )
17 Function ConfirmActionMailToUser( $sTemplate, $sField = null )
18
19 Function UpdateAction( &$oForm, &$oTemplate, $sTplAdmin, $sTplUser, $fParse = null )
20 Function UpdateActionGetValuesFromForm( $oForm )
21 Function UpdateActionPrepareValues( $aValues )
22 Function UpdateActionExists( $aValues )
23 Function UpdateActionUpdate( &$aValues )
24 Function UpdateActionUpdated()
25 Function UpdateActionMailToAdmin( $sTemplate )
26 Function UpdateActionMailToUser( $sTemplate, $sField = null )
27
28 Function RemoveAction( &$oForm, &$oTemplate, $sTplAdmin, $sTplUser, $fParse = null )
29 Function RemoveActionGetValuesFromForm( $oForm )
30 Function RemoveActionPrepareValues( $aValues )
31 Function RemoveActionExists( $aValues )
32 Function RemoveActionDelete( &$aValues )
33 Function RemoveActionMailToAdmin( $sTemplate )
34 Function RemoveActionMailToUser( $sTemplate, $sField = null )
```

A form to provide data for a module can be initialized with `AddForm()`, `ConfirmForm()`, `UpdateForm()` or `RemoveForm()`. Module fields, which should be used in the form, have to be activated in the module with `SetFieldUseForm()` before initializing the form. Fields, which are not used by the module are not used in the form, even if they where activated.

Action methods like `AddAction()`, `ConfirmAction()`, `UpdateAction()` and `RemoveAction()` offer support for typical user interaction. This methods extract the data from the provided form object, update the record in the database and send an email to the user and a notification to the owner of the Web site (email sending is configurable in the parameter module). Other action methods like `AddActionWithValues()`, `ConfirmActionWithValues()`, `UpdateActionWithValues()` and `RemoveActionWithValues()` provide similar operations and use an array of data instead of a form object.

All methods listed in listing 14 on the preceding page can be overrided in derived classes with custom implementations.

3.15 CPager

The CPager is a tool to divide the result set of a module into smaller parts, e.g. like Google does it with its search results.

A typical scenario looks as follows:

First the pager control looks for the active page number in the GET and POST parameters. This is done automatically when calling the base class constuctor of CControl. Then the number of rows, which should be displayed per page, is set for the module using `SetRows()`. With this value, the total number of pages can be calculated in the module and retrieved with `GetPages()`. This number is assigned to the pager with `SetPages()`. Then the active page can be retrieved from the control with `GetPage()` and assigned to the module with `SetPage()`.

The pager control automatically checks, if the page number is within the limits. If it is lower then one, it is set to one, if it is higher then the number of available pages, it is set to the maximum.

Listing 15: *pager.php sample*

```php
<?php

// ...

// Create the pager control
$oPager = $oFactory->Control( 'CPager' );

// Create the content module
$oModul = $oFactory->Modul( 'CGuestbook' );
$oModul->SetRows( $oParameter->GetValue( 'guestbook_rows', $oParameter->GetValue( 'rows' ) ) );
$oPager->SetPages( $oModul->GetPages() );
$oModul->SetPage( $oPager->GetPage() );
$oModul->QueryData();
$oModul->OutputTpl( $oTemplate );

// Output the pager
$oPager->OutputTpl( $oTemplate, null, '1' );
$oPager->OutputTpl( $oTemplate, null, '2' );

// ...

?>
```

Listing 16: *pager.tpl sample*

```
<!-- BEGIN: main -->

    <!-- BEGIN: pager1 -->
    {FILE "include/template/pager_small.tpl"}
    <!-- END: pager1 -->

    <!-- BEGIN: guestbook -->
        <!-- BEGIN: empty -->
        <b>Sorry, the guestbook is empty.</b>
        <!-- END: empty -->
        <!-- BEGIN: DATA -->
            <b>{modul.row_down}. <!-- BEGIN: text -->{DATA.text}<!-- END: text --></b><br>
            <!-- BEGIN: firstname -->{DATA.firstname}<!-- END: firstname -->
            <!-- BEGIN: lastname -->{DATA.lastname}<!-- END: lastname -->
            <!-- BEGIN: firstname|lastname --><br><!-- END: firstname|lastname -->
            <!-- BEGIN: email --><a href="mailto:{DATA.email}"><img src="email.gif" border="0"></a><!-- END: email -->
            <!-- BEGIN: BETWEEN --><hr><!-- END: BETWEEN -->
        <!-- END: DATA -->
    <!-- END: guestbook -->

    <!-- BEGIN: pager2 -->
    {FILE "include/template/pager_small.tpl"}
    <!-- END: pager2 -->

<!-- END: main -->
```

Listing 17: *pager_small.tpl template block sample*

```
<!-- BEGIN: PARAM -->
```

```
2  data="DATA."
3  style=""
4  <!-- END: PARAM -->
5
6  <div class="pager" style="{PARAM.style}">
7      <!-- BEGIN: first --><a href="{global.root}{global.scriptname}{{PARAM.data}param_first}"
           onmouseover="SetStatus( '{RES.opage_common_first}' );return( true );" onmouseout="
           SetStatus();return( true );"><span class="pager_first">&lt;&lt;</span></a> <!--
           END: first -->
8      <!-- BEGIN: previous --><a href="{global.root}{global.scriptname}{{PARAM.data}
           param_previous}" onmouseover="SetStatus( '{RES.opage_common_back}' );return( true )
           ;" onmouseout="SetStatus();return( true );"><span class="pager_previous">&lt;</span
           ></a> <!-- END: previous -->
9      <!-- BEGIN: site -->
10       <!-- BEGIN: active --><b>{{PARAM.data}page_site}</b><!-- END: active -->
11       <!-- BEGIN: inactive --><a href="{global.root}{global.scriptname}{{PARAM.data}
             param_site}" onmouseover="SetStatus( '{RES.opage_common_gotopage}' );return( true
             );" onmouseout="SetStatus();return( true );">{{PARAM.data}page_site}</a><!-- END:
             inactive -->
12     <!-- END: site -->
13     <!-- BEGIN: next --> <a href="{global.root}{global.scriptname}{{PARAM.data}
           param_next}" onmouseover="SetStatus( '{RES.opage_common_next}' );return( true );"
           onmouseout="SetStatus();return( true );"><span class="pager_next">&gt;</span></a><!-
           - END: next -->
14     <!-- BEGIN: last --> <a href="{global.root}{global.scriptname}{{PARAM.data}
           param_last}" onmouseover="SetStatus( '{RES.opage_common_last}' );return( true );"
           onmouseout="SetStatus();return( true );"><span class="pager_last">&gt;&gt;</span></a
           ><!-- END: last -->
15 </div>
```

If the pager control should be displayed more then once on a page (e.g. on top and at the bottom as in the example above), the `OutputTpl()` method can be called with an additional parameter `$sSub`, which is appended to the controls name, when parsing the main block. For an example see listing 15 on page 83.

The links created by the pager control contain all the GET and POST parameters provided by the global object `$oApp`, with some exceptions:

- All empty parameters are ignored.
- The parameters `datetime` and `timezone` are ignored.
- As obvious, the parameter `page` gets overwritten with the page number, which should be displayed.

oPage provides various pager templates in `include/template/`, which simplifies the use of the pager control. One example is shown in listing 17 on the preceding page.

3.16 Administration

3.16.1 CAdmin

oPage's backend is a custom Web site using the oPage framework. A derived CWebPage class and a `basic.tpl` template are stored in the `admin/custom/` directory. Besides of scripts like `admin/logon.php`, `admin/logoff.php` and `admin/navigation.php` the backend's main code is in `admin/custom/index.php` and `admin/custom/detail.php`. These files create the list and the detail view for the modules.

The illustration figure 19 on the following page shows the schema of the backend and the involved classes.

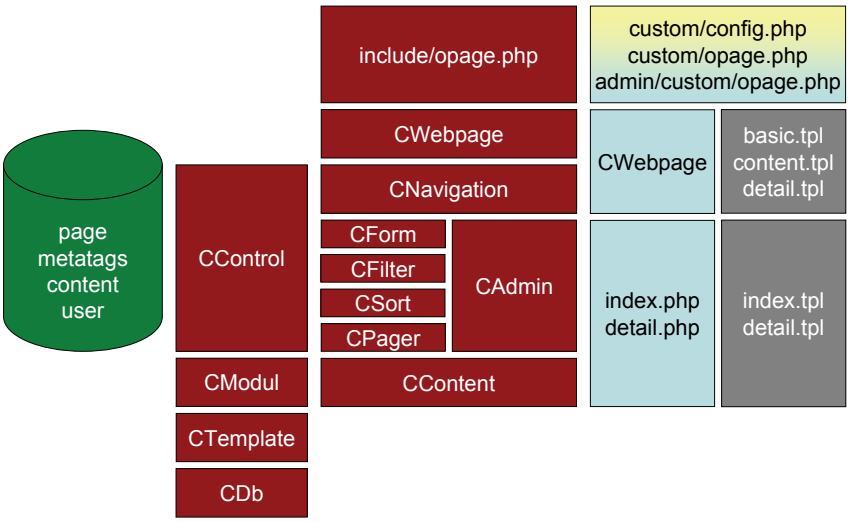


Figure 19: *oPage backend schema*

3.16.2 List View

The list view (see figure 20) displays a list of records of a module. These records can be filtered by entering criterias in the filter area. To order the result, the user can click on the appropriate sort buttons. The number of records which should be displayed can be changed, the default value can be set in the parameter module under `admin_{module name}_rows`, `admin_rows` and `rows`. If no value is defined, 20 rows are displayed. If more records are availabled, a pager control is displayed.

Some of the fields displayed can be changed in the list view. When a record has been modified, the box on the left side of the record is checked automatically. Rows with checked boxes can be stored (or deleted) with the buttons above the rows. At the same level, there is a button to add a new record. On the right side of each row, there is a button to open the record in the detail view and a button to delete the current record. Additional module specific buttons can be displayed as well.

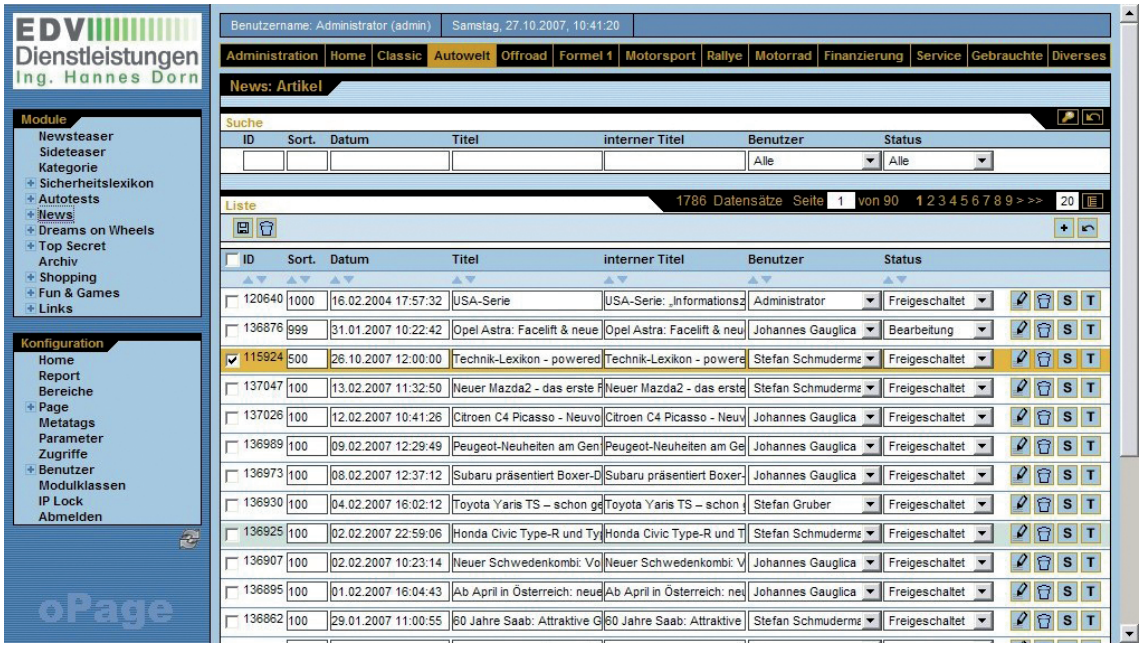


Figure 20: *Backend administration form for an article (list view)*

Listing listing 18 shows the `admin/custom/index.php` script, which creates the list view for a module.

Listing 18: *List View (admin/custom/index.php)*

```php
<?php

include_once '../../include/opage.php';

// get parameters
if ( !isset( $sSystemModul ) || $sSystemModul === null )
    $sSystemModul = $oApp->GetParameter( 'systemmodul' );
if ( !isset( $sSystemPage ) || $sSystemPage === null )
    $sSystemPage = $oApp->GetParameter( 'systempage' );
if ( !isset( $sSystemPath ) || $sSystemPath === null )
    $sSystemPath = $oApp->GetParameter( 'systempath' );

// die if parameters missing (or set default parameters)
if ( $sSystemModul === null )
    die( 'parameter_modul' );
if ( $sSystemPage === null )
    die( 'parameter_page' );
if ( $sSystemPath === null )
    die( 'parameter_path' );

// create page
$oPage = $oFactory->CWebPage( $sSystemPage, 'CWebPageX' );
$oPage->Init();
$oPage->UserCheck( $oApp->GetRoot() . 'admin/logon.php?filename=' . urlencode( $oApp->
    GetBackUrl() ) );
$oPage->RightCheck( substr( $sSystemPage, 2 ), 'read', $oApp->GetRoot() . 'admin/home/index.
    php' );
$oPage->Begin();

// create modul
$oModul = $oFactory->Modul( $sSystemModul );
$oModul->ClearFilters();
$oModul->SetView( '' );

// create admin
$oAdmin = $oFactory->Control( 'CAdmin' );
$oAdmin->SetTemplateParameter( $oPage->GetTemplateParameter() );
$oAdmin->SetModul( $oModul );
$oAdmin->SetFilter();
$oAdmin->SetSort();
$oAdmin->SetPager();

// add fields to admin
$oModul->AdminList( $oAdmin );

// run admin
$oAdmin->Action();

// output admin
$oPage->Assign( 'content', $oAdmin->Output( $sRoot . $sSystemPath . 'index.tpl' ) );

$oPage->End();

```

```
52 ?>
```

`admin/custom/index.php` creates an instance of CWebPage for the basic layout (stored in `admin/custom/basic.tpl`) of the page. This class provides methods to check, if the current user is logged on and has sufficient rights for this module. Then the module object and the CAdmin control are created. The module is assigned to the CAdmin control along with some other controls[87] used in the backend.

The CAdmin control is initialized by the module with the `AdminList()` method. This methods creates a form field for each module field that has been activated with `SetFieldUseAdminList()`.

The `Action()` method processes supplied data and updates the database by calling the modules `Insert()`, `Update()` and `Delete()` methods.

At last, the administration control is merged with the module specific template and the result is inserted into the page template.

To simplify and unify the development of the backend templates, oPage provides a wide range of template fragments for the various field types. These files are similar to the fragments described in section 3.14 on page 80, but are specific to the backend and must not be used by the frontend.

3.16.3 Detail View

While the list view is used to display a number of records, the detail view is used to edit one record. Figure 21 shows a screenshot of the detail view of a CArticle record.

At the top and at the bottom of the screen there are buttons to save, copy and delete the current record. If a preview URL has been defined for the module, a button is displayed which opens a copy of the current record and displays the preview page in a new window.

If the frontend script uses caching, update URLs can be set in the module configuration in the factory. Then the detail view displays a cache update button, which calls every URL to force a cache update.

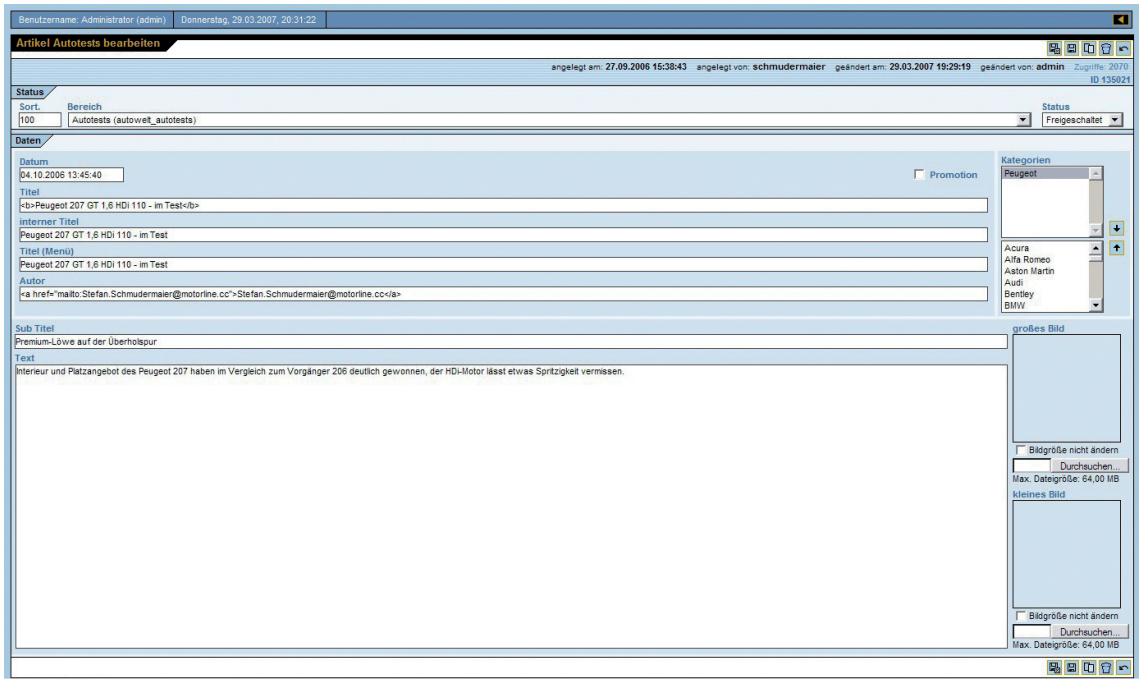


Figure 21: *Backend administration form for an article (detail view)*

[87] A custom control can be supplied to override the automatically created default CFilter, CSort or CPage control.

3.16.4 Sub Modules (1:n relations)

As described in section 3.8.8 on page 72 modules use other modules as sub modules to implement 1:n relations. In the backend both for the main module and for it's sub modules CAdmin controls are created. The main admin uses a sub admin for each sub module of the main module.

A sub module is defined using the `AddSub()` method of the main module. The instances of the sub modules are created automatically by the main module when needed. When a record is deleted in the backend, all the related records of the sub modules are deleted as well.

For each sub module a CAdmin control is created. These sub admins can again have other sub admins. All sub admins display their records in the list view node.

Figure 18 on page 74 shows the detail view of a CArticle record, which has multiple paragraphs and each paragraph having pictures. In the database the table article has a 1:n relation with `article_paragraph`, which is 1:n related with `article_paragraph_picture`.

3.16.5 Relations (m:n relations)

The module CRelation is used to represent a many to many relation. For instance, an article can be assigned to a number of categories and a category can have numerous articles. In this example, the CRelation module splits the m:n relation into two 1:n relations and stores the ID values. The module CArticle is used to manage the articles, CArticleCategory provides the categories. To manage the relations, two virtually derived classes are defined in the factory, one for each module. CArticleCategoryArticle manages the relation for CArticle class, while CArticleCategoryArticleCategory manages the relation for the CArticleCategory class.

Relations are added to the module in the factory with `AddRelation()`. Each relation module is automatically added by `AddRelation()` as a sub module with `AddSub()`.

A form control from the type SELECT_SWITCH is used to display the category in the detail view of the article. Assigned categories are listed in the top box. The available entries can be selected and moved between the boxes with the buttons on the right side.

When the article is stored, the administration control calls `StoreRelation()` of the CArticleCategoryArticle class with the ID of the article and a list of IDs of the assigned categories.

3.16.6 CModule Admin

The base class CModul provides the basic infrastructure to configure the administration control.

The backend creates an instances of CAdmin and one of the module. Then the module's `AdminList()` or `Admin()` method is called with a reference to the CAdmin instance. All activated fields are added to the admin form and configured with default settings depending on the field type.

For each m:n relation, an instance of the relation module is created and assigned to the admin instance. Also a SELECT_SWITCH field is created, where the relation data will be displayed.

Available sub modules are instanced and added to the main module with `AddModul()`. For each sub module a new instance of CAdmin is created and initialized by calling the `AdminList()` method of the sub module, and assigned to the main admin instance with `AddAdmin()`.

3.16.7 CContent Admin

The administration control can be configured within a module by implementing the methods `AdminListProperties()` and `AdminProperties()`. To prepare the user interface, the backend calls `AdminList()` for the list view and `Admin()` for the detail view. As described in section 3.16.6, this methods make some basic settings for each field depending on its type and calls the appropriate properties method.

Listing 8 on page 60 shows how the administration control of CContent is configured.

In `AdminListProperties()` the fields `title`, `title_menu` and `title_navigation` are set to read only. If the parameter `title_maxlength` or the module specific parameter `content_title_maxlength` is defined, the maximum length of the `title` field is limited to the given number. The `title` field is set as master field for `title_menu` and `title_navigation`. If one of these fields is empty, the value is copied from the master field.

The derived module can provide the `AdminList()` and the `Admin()` methods, which override the base class methods, to implement a specific behavior.

3.16.8 Import Data, Export Data

oPage offers in the list view of import and export buttons. Both buttons can be activated for a module by calling `SetImport()` and `SetExport()` in the factory class.

For the import, the field, which uniquely identifies a record can be changed from `id` to another field name with `SetImportId()`. On the basis of this field name, new, existing and deletable records are identified. `SetImportInsert()` allows Import to add new records. With `SetImportUpdate()`, existing records get updated, while `SetImportDelete()` forces oPage to delete existing records from the database, if they are not in the imported file. If `SetImportDeleteAll()` has been called, all existing records are deleted, before the import starts.

To save fields from being overwritten by update imports, these fields can be set on a leave-me-alone list with `SetImportLeave()`. If a column should not be exported, the field name can be put on the exclude-me list with `SetExportLeave()`.

The import and export format is CSV[88], a text file format where values are enclosed in quotes and separated by a semicolon.

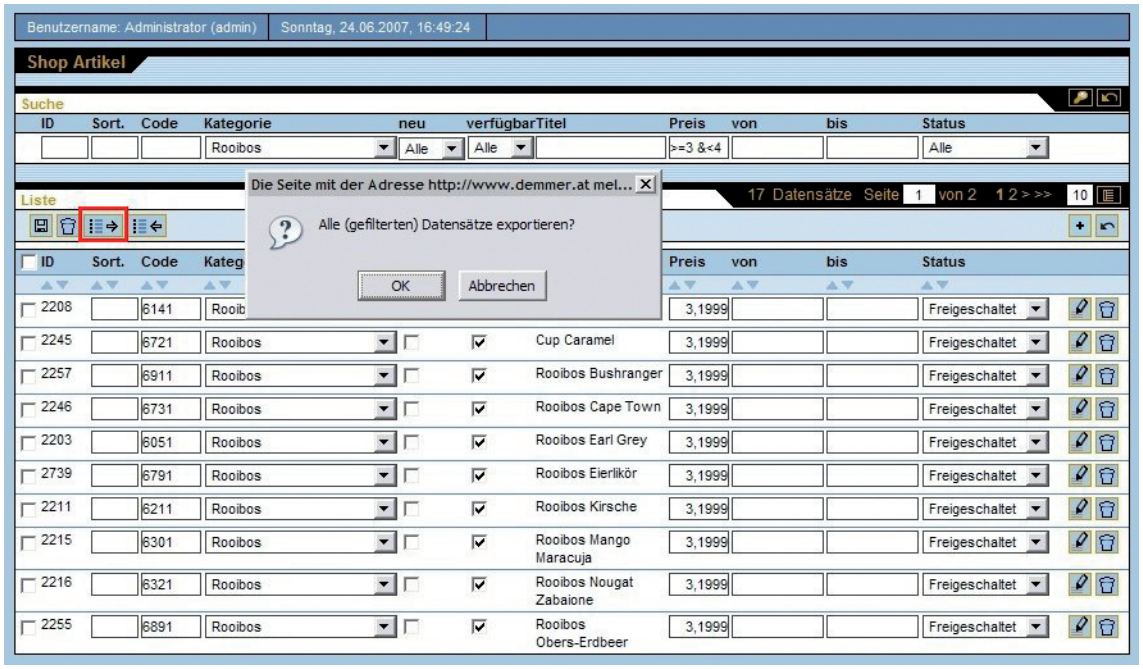


Figure 22: *data export*

When the export button is clicked (see figure 22), the administration control writes the data of the module with the modules `CsvWrite()` method into a temporary file in the `temp` folder and sends the file to the Web browser. When everything is finished, the temporary file is deleted. Only those records are exported, which apply to the current filter criterias, which the user has selected in the list view.

[88] `http://en.wikipedia.org/wiki/Comma-separated_values`

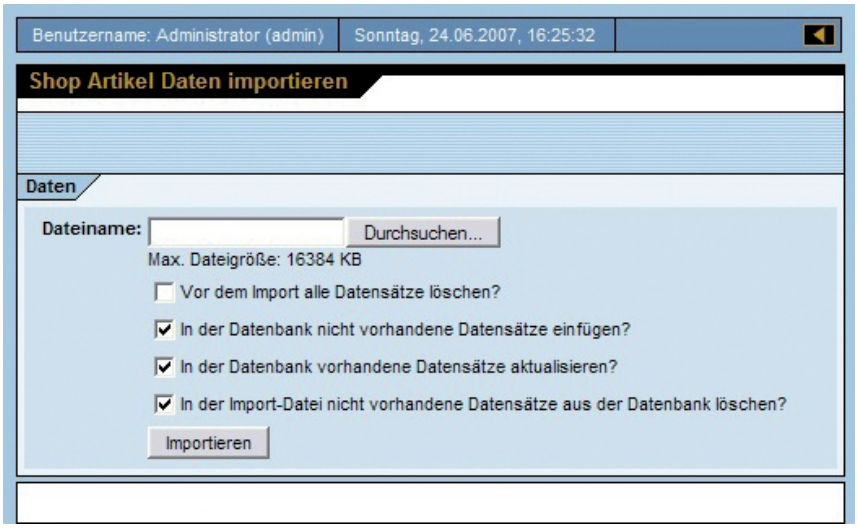


Figure 23: *import form*

The import procedure (`admin/custom/import.php`) opens a form, where the user can select a file and set the import options. When the form is submitted, the uploaded file is processed by the `CsvRead()` method of the module.

The import function can also be called directly. For example on an internal server, a script can create a CSV file with stock item data from a warehouse database. This script then makes a HTTP POST request with the appropriate parameters. The file is uploaded to the Web server and processed automatically without user intervention. To export data automatically, a similar function is available in oPage at `admin/custom/export.php`.

3.16.9 Preview

Users like to see, how the content looks on the Web site, even before it is saved and available to the public. This can be done with the preview mode. The preview button gets displayed in the detail window only if the preview mode has been activated for the module. oPage uses the frontend Web site to display the preview. Since this preview URL may be different on every Web site and for every module, the preview URL has to be set for each module in the factory.

To view a preview of a record, oPage creates a copy of the entry and then calls the frontend Web site to display it. This copy is marked with the preview flag (the field `preview` contains the value `'y'`) and will not be displayed on the regular Web site. Around the preview, an invisible frameset is loaded. When the frameset gets closed, it calls the backend URL `admin/preview/delete.php` to delete the preview record.

Creating the copy for the preview is tricky. The database record needs to be copied, but as well all records of the sub modules. All the data stored in the file system (like images and other binary objects) have to be copied. If the edited records contain files to be uploaded, the new data needs to be processed, e.g. the uploaded files need to be copied to their final destination and images need to be resized.

All this copying is done by the administration control for the main and the sub modules. For processing the form input and the uploaded files, the form control is used as usual.

When the preview record gets deleted, all the records of the sub modules get deleted as well, along with all external files belonging to this records.

Listing 19: *custom factory.php for preview*

```
1 <?php
2
```

```
3  include_once $sRoot . 'include/factory.php';
4
5  Class CFactoryX extends CFactory
6  {
7      Function _ModulConfigure( &$oObject, $sClass = null )
8      {
9          global $oApp;
10         global $oSession;
11         global $oCustomer;
12
13         if ( is_null( $sClass ) )
14             $sClass = $oObject->GetClass();
15
16         switch( $sClass )
17         {
18             case 'CNews':
19                 parent::_ModulConfigure( $oObject );
20                 $oObject->SetFieldUseAdmin( 'preview' );
21                 $oObject->SetPreview( 'news/detail.php', 'news' );
22             break;
23
24             default:
25                 parent::_ModulConfigure( $oObject );
26             break;
27         }
28
29         return( $oObject );
30     }
31 }
32
33 ?>
```

In listing 19 on the preceding page the news module is configured to provide preview support.

In the base class CModul, a filter is set to ignore the preview records. If a preview record ID is provided, the filter is extended, to allow the display of the desired record. When the preview button is clicked, the URL `news/detail.php` is called and the record id of the preview record is provided with the parameter `news`.

Listing 20: *news detail.php*

```
1  <?php
2
3  // include oPage main file
4  include_once '../include/opage.php';
5
6  // get parameter value for record id
7  $iId = $oApp->GetParameter( 'news' );
8
9  // create new page object
10 $oPage = $oFactory->Webpage( 'news', 'CWebpageX' );
11 $oPage->Init();
12 $oPage->Begin();
13
14 // create the content module
15 $oNews = $oFactory->Modul( 'CNews' );
16 $oNews->SetFilter( 'id', $iId );
17 $oNews->QueryData();
18
```

```
19 // output the content module using template 'detail.tpl' and assign result to page
20 $oPage->Assign( 'content', $oNews->Output( 'detail.tpl' ) );
21
22 // finish page and send it to the client
23 $oPage->End();
24
25 ?>
```

As shown in listing 20 on the preceding page, no special precautions have to be taken in the output script to support preview mode, since everything is done in the module base class.

Listing 21: *preview part in modul.php*

```
1  <?php
2
3  Class CModul
4  {
5      Function CModul( $sName )
6      {
7          global $oApp;
8
9          // If no name is provided, use default name
10         if ( is_null( $sName ) )
11             $sName = $this->GetDefault();
12
13         $this->SetFieldUse( 'preview', false );
14         $this->SetPreviewId( $oApp->GetParameter( $sName . '_preview' ) );
15         $this->SetFilter( 'preview', array( 'n', '', null ) );
16     }
17
18     Function Admin( &$oAdmin )
19     {
20         $oAdmin->SetButtonPreview( false );
21         if ( $this->GetFieldUse( 'preview' ) )
22         {
23             $this->SetFilter( 'preview', array( 'n', '', null ) );
24
25             if ( $this->GetFieldUseAdmin( 'preview' ) )
26             {
27                 $oAdmin->SetButtonPreview( true );
28                 if ( IsNull( $oAdmin->GetPreviewUrl() ) )
29                     $oAdmin->SetPreviewUrl( $this->GetPreviewUrl() );
30             }
31         }
32     }
33
34     /** Extends the provided sql where string with the preview filter, so that the preview ⤸
           record gets always displayed.
35     */
36     Function GetPreviewFilter( $sFilter = '' )
37     {
38         if ( !$this->GetFieldUse( 'preview' ) )
39             return( $sFilter );
40
41         $sId = $this->GetPreviewId();
42         if ( IsNull( $sId ) )
43             return( $sFilter );
```

```
44
45          if ( $sFilter != '' )
46              $sFilter = '(' . $sFilter . ')␣or␣';
47      $sFilter .= '(' . $this->GetField( 'preview' ) . "='y'␣and␣" . $this->GetField( 'id' ) . "=$sId␣)";
48
49          return( $sFilter );
50      }
51  }
52
53  ?>
```

3.16.10 Privileges

In this section, user and rights management for the backend is discussed.

By default, rights management is provided in a very simple manner. The module CUser manages backend user accounts, which all have administration privileges.

With every record edited, the name of the user is stored along with the current date. Also the last modified date and username is updated with every change.

If differentiated access rights are needed, per user rights management can be activated for each account by calling `$oApp->SetFeature( 'user', 'rights', 1 )` in `custom/opage.php`. For each account, multiple rights can be assigned. These records are managed by the modules CRight, CUserRight and CRightUser. The last two modules manage the m:n relation between CUser and CRight and are virtually derived classed from CRelation module (see paragraph 3.4 on page 50 as well as `include/factory.php`). These classes are needed to provided separate views for the same table: CUserRight provides the database field `user_id` as internal field `id`, CRightUser uses `right_id` as internal field `id`. In the administration form control, this classes are used to fill two listboxes, one with the selected values, the other with the available non-selected items. In the users detail dialog, rights can be assigned to users, in the detail window for the rights, users can be assigned to rights.

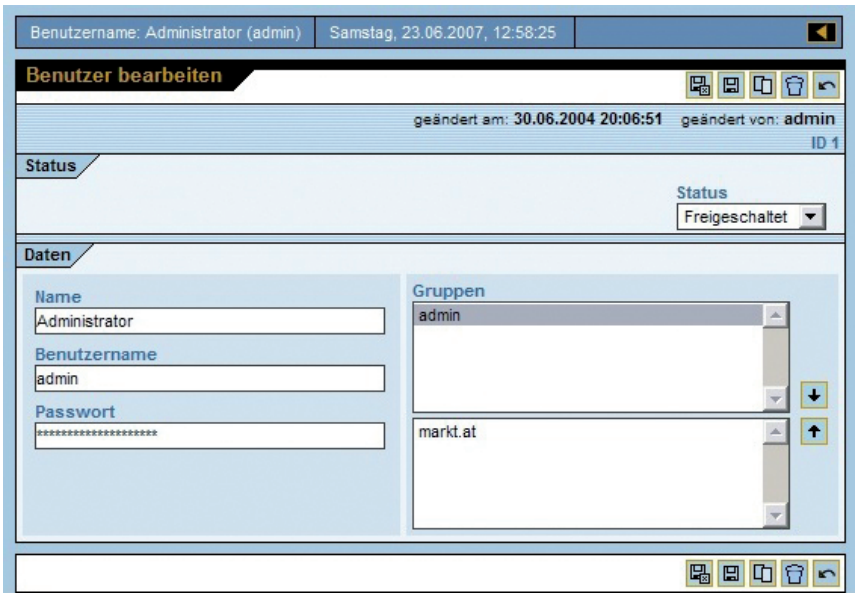


Figure 24: *user details*

Rights are currently managed per module, record level access lists are not available by now.

Users can also be assigned to groups, and groups can have rights. Here the module classes CUser, CRight,

CGroup and the relation classes CUserGroup, CGroupUser, CGroupRight and CRightGroup are used. This mode is activated with `$oApp->SetFeature( 'user', 'right', 2 )`.

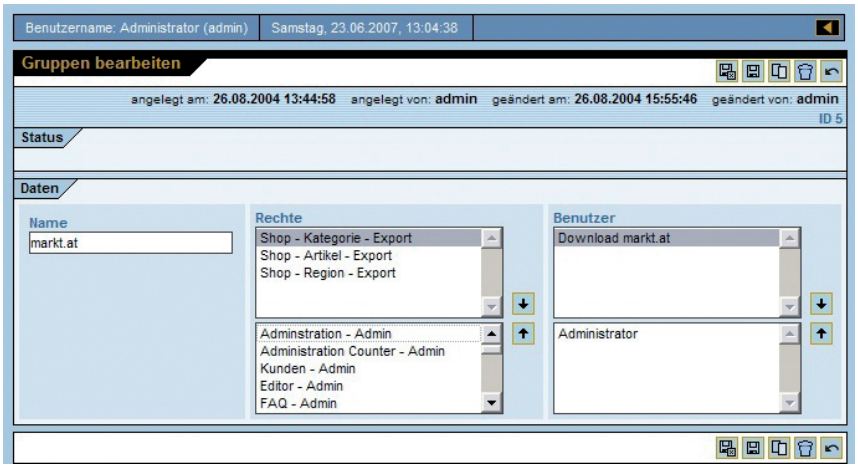


Figure 25: *group details*

While users and groups can be freely created, names of rights have to follow some rules. There are different levels of rights:

admin
This right allows everything and includes all other rights.

admin#import
With this right, adding new records is allowed (this right includes the `add`, `change` and `delete` right).

admin#add
With this right, adding new records is allowed (this right implicates the `read` right).

admin#change
This right allows to change existing records and includes the `read` right.

admin#delete
To delete existing records, this right is needed (and includes also the `read` right).

admin#export
With this right, the user can call the export function.

admin#read
With this right, all records are allowed to be read.

These rights are global and apply to every module. Module specific rights can be specified by prepending the name of the right with the name of the module.

***modulname*#admin**
***modulname*#import**
***modulname*#add**
***modulname*#change**
***modulname*#delete**
***modulname*#export**
***modulname*#read**

Using the rights module

When a user logs into the backend with the `Logon()` method of CUser, the assigned rights are loaded and stored in the session.

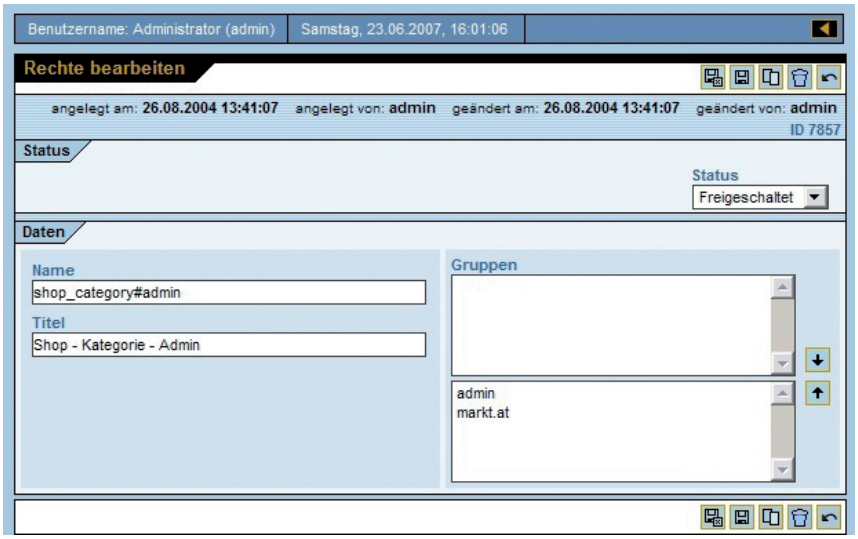


Figure 26: *right details*

To assure, that the user has sufficient rights, the method `Right(` $sModul$`,sRight )` is used. This method is called with the name of the module and the desired operation (`admin`, `export`, `add`, `change`, `delete`, `import` or `read`). The function checks, if the user has the privileges or a right, which includes the desired one, and returns true or false.

In `admin/navigation.php`, the backend navigation control is filled. If a user has no right to at least read a modules records, the menu item is not displayed at all. If an entry is a sub item of another menu item, e.g. CShopCategory is usually a sub entry of CShop, the user has to have at least the `read` right for CShop to access CShopCategory. If the user has no right for the parent entry, all sub menu items are hidden.

3.17 Advanced topics

3.17.1 Setup

Installation of oPage on a Web server is easy. All the files have to be transfered to the document directory or to a sub directory of it. Then the default `custom/config.php` has to be updated with the appropriate settings. The database connection to an existing database has to be set and if oPage is installed in a sub directory, the `root` path needs to be adjusted. Also the `install` setting needs to be changed to true to enable the install script.

The install script can then be called at `admin/install.php` and creates all necessary database tables and inserts some default records. The install script accepts the parameter `install` with one of the following values to invoke just a part of the installation procedure: `db`, which just checks the database connection, `session` to create the session table, `modul` to process the module specific installation, `parameter` to insert necessary default parameter values, and `user` to create the default user and assign administration privileges.

Now the backend at `admin/index.php` can be accessed with username and password `admin`. The first action should be changing the administration password in the user module. Also the default email address should be set in the parameter module.

To avoid other people running the install script, the script should be disabled in `custom/config.php`.

The setup script at `admin/setup.php` works similar, but requires a previous login to the backend, and can also be enabled and disabled in the oPage configuration. Setup only processes the module part of the installation procedure and creates missing tables and fields (and deletes deactivated fields) for each module. It also automatically adds backend navigation page entries for this modules. Additional modules can be added in

the module CModule. When a modules configuration was changed, setup has to be run to apply the changes to the database. See section 3.17.2 for details how to configure a module.

The install and the setup script can be called with the parameter `modul` to run the process just for one module.

With the test script at `admin/test.php`, some checks are run, to evaluate some basic Web site functions like sending an email or resizing an image. The test script can only be started, when setup is enabled.

The initial installation and the setup can be extended with modules, backend and frontend page and navigation entries in `custom/admin/setup/modul.php` and `custom/admin/setup/page.php`. Have a look at the enclosed examples listing 22 and listing 23.

Listing 22: *custom/admin/setup/modul.php*

```php
<?php

// Modules
$aModul[] = 'CNews';
$aModul[] = 'CLinks';

// Tools
$aTools[] = 'CMetatags';
$aTools[] = 'CPageNavigation';

?>
```

Listing 23: *custom/admin/setup/page.php*

```php
<?php

// create content page entries
SetupPageAdd( $sContent, $aPageContent, 'home', 'index.php', null, 'Home' );
SetupPageAdd( $sContent, $aPageContent, 'news', 'news/index.php', null, 'News' );
SetupPageAdd( $sContent, $aPageContent, 'links', 'links/index.php', null, 'Links' );

// add all content page entries to the navigation menu
$oObject = $oFactory->Modul( 'CPageNavigation' );
SetupPageMenu( $sContent, $oObject, $aPageContent );

?>
```

The base class CModul provides methods to setup a module, which are used by the installation scripts. The `Setup()` method creates a list of all active fields (including multiple fields for each language where required). Then either the `TableCreate()` or the `TableAlter()` method of the database access layer is called. These methods create SQL statements and executes them, to create and change the table and create indexes.

By overriding the `SetupData()` method, a module can add some default data, when oPage is installed. The base class method expects an array of database rows containing key value pairs.

3.17.2 Customization

One of the design goals of oPage was to make it highly customizable. The design is defined in the basic and in the content templates. In this section, the customization of the framework is described.

The basic configuration of oPage has been discussed in section 3.17.1 on the facing page. Other settings, like the size of images, can be changed in the parameter module within the backend.

Files in the directories `admin/`, `download/`, `include/`, `popup/` and `redirect/` contain the oPage system and must not be changed. There are other places to add custom behavior.

opage.php

To extend the system initialization (`include/opage.php`) a custom version can be created in `custom/opage.php`. In listing 24 first it is checked, if oPage is not in backend mode. For all frontend pages, a global session object is created, which is used to store logon information for the CCustomer module. The `LogonAuto()` method looks for username and password, initializes the customer module and stores the information in the session object. At last, the modus of the counter module, which has been created in `include/opage.php` is changed to the modus set in the parameter module.

Listing 24: *custom/opage.php*

```
1  <?php
2
3  if ( !$oApp->GetAdmin() )
4  {
5      // Create session object
6      $oSession = $oFactory->Session();
7
8      // Create customer modul
9      $oCustomer = $oFactory->Modul( 'CCustomer' );
10     $oCustomer->LogonAuto();
11
12     // Configure global counter modul
13     $oCounter->SetMode( $oParameter->GetValue( 'counter_mode', '' ) );
14 }
15
16 ?>
```

webpage.php

A derived class of CWebpage can be stored in `custom/webpage.php`. This class is used to provide the project specific code, which is common for each page. As shown in figure 13 on page 36, a Web page consists of common parts like navigational links, or logos, which are on every page of the Web site, and the content, which changes from page to page. The layout for this common parts is in `custom/basic.tpl` while the according code is in `custom/webpage.php`.

The navigation controls are created and filled with data. Also metatags are queried and assigned to the page template.

listing 25 shows an example `custom/webpage.php`. In the constructor, the parent constructor is called and the default page template is set. In the `Init()` method the access counter for the current page is increased. To get the page specific metatags (or the default), an instance of the CMetatags module is created and a filter is set for the current page. Then the result set is limited to one record, the data is queried and merged with the template. Also the back control is created and joined with the template.

The navigation control is created and filled with two example page entries. Navigation entries could also be inserted with the `FillNavigation()` method of a module like CCategory. For each navigation control a CMenu module can be used to manage the menu tree in the backend. In this example, the entries are provided by the module CPageNavigation (see also listing 26 on page 100).

After all entries have been added, the current page is set as active. The navigation control is merged later with the template (in the `End()` method), to allow a script to insert additional entries.

Listing 25: *custom/webpage.php*

```
1  <?php
2
3  include_once $sRoot . 'include/webpage.php';
4
5  Class CWebpageX extends CWebpageBasis
```

```
6  {
7      var $oNavigation;
8
9      Function CWebpageX()
10     {
11         parent::CWebpage();
12         $this->SetTemplate( 'custom/basic.tpl' );
13     }
14
15     Function Init()
16     {
17         global $oApp;
18         global $oCounter;
19
20         parent::Init();
21
22         // Increase counter for this page
23         $oCounter->Increase( 'page', $oApp->GetScriptName(), $this->oPage->Get( '
               title_display' ) );
24     }
25
26     Function Begin()
27     {
28         global $oFactory;
29
30         parent::Begin();
31
32         // Create and use the metatags module
33         $oMetatags = $oFactory->Modul( 'CMetatags' );
34         $oMetatags->SetFilter( 'name', $this->oPage->Get( 'name' ) );
35         $oMetatags->SetLimit( 1 );
36         $oMetatags->QueryData();
37         $oMetatags->OutputTpl( $this->oTpl );
38         unset( $oMetatags );
39
40         // Create and output back control
41         $oBack = $oFactory->Control( 'CBack' );
42         $oBack->OutputTpl( $this->oTpl );
43         unset( $oBack );
44
45         // Create navigation control
46         $this->oNavigation = $oFactory->Control( 'CNavigation' );
47         $this->AddNavigation( $this->oNavigation, 'home' );
48         $this->AddNavigation( $this->oNavigation, 'news' );
49
50         // Insert navigation entries from database
51         $oPageNavigation = $oFactory->Modul( 'CPageNavigation' );
52         $oPageModule->QueryData();
53         foreach( $oPageModule->GetResult() as $aRow )
54             $this->AddNavigation( $this->oNavigation, $aRow[ 'name' ] );
55
56         // Set current page active
57         $this->oNavigation->SetActive( $this->oPage->Get( 'name' ) );
58     }
59
60     Function End()
61     {
```

```
62          // Output navigation control
63          $this->oNavigation->OutputTpl( $this->oTpl );
64          return( parent::End() );
65      }
66  }
67
68  ?>
```

factory.php

The initialization script `include/opage.php` looks for a project specific factory class, which extends the standard factory. The factory class is used to create and customize modules, to create virtually derived modules, as well as to include additional project specific classes.

listing 26 shows an example custom factory class. A real derived class (CContentHomepage) and three virtually derived classes (CContentService, CPageNavigation and CServiceCategory).

The module CContentHomepage is described in listing 27 on page 102.

For each derived class, a new default name for the module is set, which overrules the default name of the parent class. The real class name is set, the class file is included, and the parent method `_Modul()` is called, where the instance is created. This parent method calls `_ModuleConfigure()`, where the object can be customized. A classname parameter can be provided to configure a module with different settings.

Both the CContentHomepage and the CContentService modules are derived from CContent. Settings, which apply to all CContent classes, are combined.

By enabling the `category_name` field, the CContentService module is used in combination with a category module. By default, the module would use the CContentServiceCategory module. By calling `SetClass Group( 'CService' )`, the CContentService module uses CServiceCategory instead.

Listing 26: *custom/factory.php*

```
1   <?php
2
3   include_once $sRoot . 'include/factory.php';
4
5   Class CFactoryX extends CFactory
6   {
7       Function _Modul( $sClass, $sName = Null, $fInclude = Null )
8       {
9           global $oApp;
10          global $sRoot;
11
12          switch( $sClass )
13          {
14              case 'CContentHomepage':
15                  if ( is_null( $sName ) )
16                      $sName = 'homepage';
17                  $sClass = 'CContentX';
18                  $sInclude = 'custom/modul/';
19              break;
20
21              case 'CContentService':
22                  if ( is_null( $sName ) )
23                      $sName = 'service';
24                  $sClass = 'CContent';
25              break;
26
```

```
27              case 'CPageNavigation':
28                  if ( is_null( $sName ) )
29                      $sName = 'page_navigation';
30                  $sClass = 'CMenu';
31              break;
32
33              case 'CServiceCategory':
34                  if ( is_null( $sName ) )
35                      $sName = 'servicecategory';
36                  $sClass = 'CCategory';
37              break;
38          }
39
40          if ( isset( $sInclude ) )
41          {
42              include_once $sRoot . $sInclude . strtolower( substr( $sClass, 1, -1 ) ) . '.php
                    ';
43              $fInclude = false;
44          }
45
46          $oObject = parent::_Modul( $sClass, $sName, $fInclude );
47
48          return( $oObject );
49      }
50
51      Function _ModulConfigure( &$oObject, $sClass = null )
52      {
53          if ( is_null( $sClass ) )
54              $sClass = $oObject->GetClass();
55
56          switch( $sClass )
57          {
58              case 'CContent':
59                  parent::_ModulConfigure( $oObject );
60                  $oObject->SetFieldUseAdminList( 'abstract', true );
61                  $oObject->SetFieldUse( 'date', false );
62                  $oObject->SetFieldUse( 'title', false );
63              break;
64
65              case 'CContentHomepage':
66                  $this->_ModulConfigure( $oObject, 'CContent' );
67                  parent::_ModulConfigure( $oObject );
68              break;
69
70              case 'CContentService':
71                  $this->_ModulConfigure( $oObject, 'CContent' );
72                  parent::_ModulConfigure( $oObject );
73                  $oObject->SetClassGroup( 'CService' );
74                  $oObject->SetFieldUseAdminList( 'category_name', true );
75              break;
76
77              case 'CPageNavigation':
78                  parent::_ModulConfigure( $oObject );
79                  $oObject->SetFieldUse( 'parent_name', false );
80              break;
81
82              case 'CServiceCategory':
```

```
                parent::_ModulConfigure( $oObject );
                $oObject->SetFieldUseAdminList( 'name' );
                $oObject->SetFieldUseAdminList( 'image_small' );
            break;

            default:
                parent::_ModulConfigure( $oObject );
            break;
        }
    }
}

?>
```

module/contenthomepage.php

A project specific module is usually stored in `custom/modul` (have a look at the include path `$sInclude` in listing 26 on page 100). This can be a new or a derived class from an existing module.

The example in listing 27 shows how the CContent module is extended with the additional field `author`. If the additional field should be displayed in the backend, the administration templates `index.tpl` and `detail.tpl` have to be adapted. The original files have to be copied from `admin/content/` to `custom/admin/contenthomepage/` and can be edited there. The templates provided with oPage must not be changed. The path to the customized templates has to be set, e.g. in the constructor or in the factory.

Listing 27: *custom/modul/contenthomepage.php*

```
<?php

include_once $sRoot . 'include/modul/content.php';

Class CContentHomepageX extends CContent
{
    Function CContentHomepageX( $sName = Null )
    {
        if ( is_null( $sName ) )
            $sName = 'content_homepage';

        // Initialize base class
        parent::CContent( $sName );

        // Set path to admin templates index.tpl and detail.tpl: custom/admin/contenhomepage/
        $this->SetAdminPath( 'custom/admin/' . substr( get_class( $this ), 1, -1 ) . '/' );
    }

    Function SetFields()
    {
        parent::SetFields();
        $this->SetField( 'author', Null, true );
    }

    Function AdminListProperties( &$oAdmin, $sName )
    {
        parent::AdminListProperties( $oAdmin, $sName );
        switch( $sName )
        {
            case 'author':
```

```
31                  $oAdmin->SetWidth( $sName, 10 );
32              break;
33          }
34      }
35
36      Function AdminProperties( &$oAdmin, $sName )
37      {
38          parent::AdminProperties( $oAdmin, $sName );
39          switch( $sName )
40          {
41              case 'author':
42                  $oAdmin->SetWidth( $sName, 130 );
43              break;
44          }
45      }
46  }
47
48  ?>
```

hook.php

The initilization script `include/opage.php` creates a global hook object. If a custom hooking class CHookX is available at `custom/hook.php`, the file is included by the factory.

oPage throws events on various occasions. For each event, the `Execute()` method of the hook object is called. The provided example listing 28 is part of an online shop, where customers earn bonus points depending on the amount of their order.

Each customer has an account (CustomerPoints) where the points are accounted. The shopping script `commit_order.php` creates a new record in COrder and calls the hook object with the module's name `shop_order` and the action name `ordered`. If the customer has encashed some points or earned new points, the `Execute()` method of the custom CHook class inserts the according entries into the points account.

In a later step, the `order_commit.php` script sends a confirmation email to the customer and a notification to the order department. When the emails are prepared, the `mail` action is executed. The custom hook class creates an instance of CCustomerPoints, sets the filter to the current customer and appends the module to the COrder instance. When the order is merged with the email layout, the points entries are queried and merged with the template.

Listing 28: *custom/custom_hook.php*

```
1   <?php
2
3   include_once $sRoot . 'include/hook.php';
4
5   Class CHookX extends CHook
6   {
7       Function Execute( $sName, $sAction, $oObject )
8       {
9           global $oFactory;
10          global $oApp;
11
12          $sHook = $sName . '#' . $sAction;
13
14          switch( $sHook )
15          {
16              case 'shop_order#ordered':
17                  parent::Execute( $sName, $sAction, $oObject );
18                  $oOrder = &$oObject;
```

```
19
20              if ( $oOrder->Get( 'points' ) > 0 || $oOrder->Get( 'points_encashed' ) > 0 )
21              {
22                  $oCustomerPoints = $oFactory->Modul( 'CCustomerPoints' );
23
24                  // Get and format order data
25                  $aData = $oOrder->GetRow();
26                  $oOrder->PrepareRow( $aData );
27
28                  // Some points encashed?
29                  if ( $oOrder->Get( 'points_encashed' ) > 0 )
30                  {
31                      $aValues = array();
32                      $aValues[ 'customer_number' ] = $oOrder->Get( 'customer_number' );
33                      $aValues[ 'name' ] = 'order#' . $oOrder->Get( 'order_number' ) . '
                            _encashed';
34                      $aValues[ 'points' ] = $oOrder->Get( 'points_encashed' ) * -1;
35                      $oCustomerPoints->AddActionWithValues( $aValues );
36                  }
37
38                  // Some new points earned?
39                  if ( $oOrder->Get( 'points' ) > 0 )
40                  {
41                      $aValues = array();
42                      $aValues[ 'customer_number' ] = $oOrder->Get( 'customer_number' );
43                      $aValues[ 'name' ] = 'order#' . $oOrder->Get( 'order_number' );
44                      $aValues[ 'points' ] = $oOrder->Get( 'points' );
45                      $oCustomerPoints->AddActionWithValues( $aValues );
46                  }
47              }
48          break;
49
50          case 'shop_order#mail':
51              parent::Execute( $sName, $sAction, $oObject );
52
53              $oMail = &$oObject;
54              $oOrder = &$oMail->GetModul();
55
56              $oCustomerPoints = $oFactory->Modul( 'CCustomerPoints' );
57              $oCustomerPoints->SetFilter( 'customer_number', $oOrder->Get( '
                    customer_number' ) );
58
59              // Add the Points module as a submodule of the customer module
60              $oOrder->AddModul( $oCustomerPoints, null, null, $oCustomerPoints->GetName()
                    );
61          break;
62
63          default:
64              parent::Execute( $sName, $sAction, $oObject );
65          break;
66      }
67  }
68 }
```

3.17.3 Multilanguage

In this section, multilanguage support and internationalisation is described.

The available languages and the default language are set in `custom/config.php`. For each configuration the language for the content (frontend) and the administration interface (backend) can be set (see listing 9 on page 64). For each language, the long name, a short name and a postfix have to be set.

Each module field has a marker, if it is a multilanguage field. For each language, a field is created in the database table. The name of the field is extended with a language specific postfix as defined in `custom/config.php`.

The global application object provides methods to query the current active language. In the backend, the application object automatically uses the administration language settings. `GetLanguage()` returns the ID of the current active language, `GetLanguageShort()` returns the short name for a language ID, while `GetLanguageText()` provides the long name. `GetLanguagePostfix()` returns the field name extension for the active language, which is used by modules to identify the database columns for the current language. All this functions accept a parameter `$fContent`. If this parameter is set to true (by default it is false), `$oApp` is forced to use the language settings for the frontend. This is necessary in the backend to provide multilanguage support independently of the languages used in the administration interface.

The active languages for the frontend and the backend are stored in a cookie. To change the language, `SetLanguage()` can be called with the ID (as defined in `custom/config.php`) of the language. To store the settings in the cookie, `WriteLanguage()` can be called. The ID for a language can be queried with the long name and `GetLanguageID()` or with the short name and `GetLanguageShortID()`. `$oApp` automatically checks, if the GET or POST parameter `language` is available. If it has a valid value (either language long name or short name). The new active language is stored in the cookie. When a user logs on to the backend, the users language settings are also used.

System messages are defined in `include/resource.php` and are used reflecting the users to inform the backend user, when records where successfully stored or if an error occured.

Also numbers and date format functions make use of the language settings. The factory object initializes instances of CNumber and CDate with the current language. Currently resource strings, number and date formats are availabled in German and English. Additional language settings can be added.

To provide static content in different languages, separate HTML files have to be created for each language. When loading such a file, the correct filename can be created using `$oApp->GetLanguage Postfix()`. For an example see listing 1 on page 37.

For templates, this method also can be used. But it is sometimes more convenient to put all the text content into a resource file, such as `custom/resource/de.res`) and heaving just one template. The template engine uses the short language text to generate the filenames of the resource files. Have a look at section 3.5.8 on page 59.

3.17.4 Caching

There are different methods how Web pages are generated by content management systems. Some systems start the output right after a page has changed and store the result (static approach). Others (like oPage) create the page each time it is requested. This dynamic approach provides more flexibility, especially when information is used in the page creation process, that can change with every request like the name of the current user or the current date and time.

At the first request the Web page is created and stored in the cache. For the following requests, it is checked, if the content in the cache is still valid or has to be recreated. In this mode (*update before display*), the first user and users requesting an outdated page have to wait while the content is created. If possible, the page can be precreated when the content is stored. When the content is invalid, but available in the cache, then the outdated page can be sent to the client, and the update process is started afterwards (*update after display* mode). This reduces the response time of the Web site.

To cache content the CCache class is available. This class provides the infrastructure to store, validate and retrieve the content. A new instance of CCache can be created with the factory method `Cache()`.

The CWebPage uses CCache to store the generated pages. To cache content parts, the cache can be used directly.

Listing 29: *parts/news.php with content caching*

```php
<?php

defined( 'oPage' ) or die();

// Create and initialize the content cache
$oCache = $oFactory->Cache();
$oCache->SetScriptname( 'custom/parts/news.php' );
$oCache->Init();

// Output main teasers
if ( $oCache->GetCreate() )
{
    // Create content template
    $oTemplate = $oFactory->Template( $sRoot . 'custom/parts/news_content.tpl' );

    // Create the module
    $oTeaser = $oFactory->Modul( 'CTeaserNews' );
    $oTeaser->SetFilter( 'section_id', $oApp->GetParameter( 'section' ) );
    $oTeaser->QueryData();

    // Output records
    $oTeaser->OutputTpl( $oTemplate );

    // Get the content from the template
    $oTemplate->Parse();
    $sContent = $oTemplate->Text();

    // Store the content in the cache
    $oCache->SetContent( $sContent );
    $oCache->Write();
}
else
    // Get the content from the cache
    $sContent = $oCache->GetContent();

// Create main template
$oTemplate = $oFactory->Template( $sRoot . 'custom/parts/news.tpl' );

// Assign main teaser content
$oTemplate->Assign( 'content', $sContent );

// Output right teasers
include $sRoot . 'custom/fragments/teaserside.php';

// Set the content to the page
$oTemplate->Parse();
$oPage->Assign( 'content', $oTemplate->Text() );

?>
```

The provided example (listing 29 on the preceding page) creates a part of a page. The result of the module CTeaserNews is stored in the cache while the rest of the page is created on every request.

3.17.5 Search

The search method accepts a search string which can contain wild cards like * or %. The provided string is prepared with `StringDBFilter()` from `include/tools.php` where all * are replaced by %. Then an array of filters is created with every used `TEXT` or `TEXTBOX` field.

If any filter has been assigned, the filter array is assigned to the module and the `QueryData()` is called. If no text fields are used, the result set is cleared.

Internally the filter array is converted into sql `WHERE` parts which are combined by the `OR` operator and use the `LIKE` compare operator.

Listing 30: *modul.php search method*

```
/** Method for fulltext search in modules
    Search sets/clears $this->aResult
    @param search string
    @return void
*/
Function Search( $sSearch )
{
    // prepare search statement, $sCompare result is not used.
    $sSearch = StringDBFilter( $sSearch, $sCompare );

    $aFilter = array();
    foreach( $this->GetFieldList() as $sName )
        if ( $this->GetFieldUse( $sName ) )
            switch( $this->GetFieldType( $sName ) )
            {
                case 'TEXT':
                case 'TEXTBOX':
                    $aFilter[] = array( 'field' => $sName, 'value' => $sSearch );
                break;
            }

    if ( count( $aFilter ) > 0 )
    {
        $this->SetFilter( 'id', $aFilter, 'like' );
        $this->QueryData();
    }
    else
        $this->ClearResult();
}
```

3.18 Summary

In this chapter, the technical description of the oPage framework has been provided. oPage consists of the framework core, a reusable backend and a highly customizable frontend. The core and the backend program files can be shared between different Web sites.

The content is stored in database tables. oPage provides class files (modules) to read, insert, update and delete records from these tables. CModul is the base class for these classes and provides the necessary infrastructure. It is also used as a report generator to query, filter, sort and page records and merge the result with the template engine. A hierarchy of modules can be used to represent database relations.

The template engine loads a template from a text file, includes template fragments, replaces resource strings and stores the generated structure in its cache. The template is divided into smaller parts (blocks), which can be parsed individually. These blocks contain placeholders and instructions how to format the provided values. Blocks on the same level can be logically combined by *and* and *or* operators. The template engine in combination with the module class is a versatile report generator, which can create text output like HTML, XML, plain text, CSV, RTF, LATEX and alike.

A controller script creates the Web page. It includes the frameworks main file `include/opage.php`, which initializes the project environment. Then a customized CWebpage class is instanced, which loads the Web site layout and inserts common elements like navigation controls. Modules are instanced as desired and the output is assigned to the Web page object. This object puts all together and sends to result to the Web browser.

All objects in oPage are created by the factory. A project specific derived factory class can be automatically instanced by the oPage main include file, otherwise the default factory is used. The factory is a central place, where objects can be configured, before they get used. This pattern allows to create and use derived classes, without modifying existing source code.

oPage provides a versatile form engine, which processes plain and multiline text, date, number, currency, binary upload and other input types. oPage offers template fragments in the template library to build plain and tabular forms. The base class CModul provides a number of methods to simplify acquiring and processing of user input. Methods are provided to initialize, validate and process forms to add, update and delete records. According functions exist, which send appropriate emails for these actions using the processed data.

The adminstration control processes the backend forms. Two kinds of forms can be used: The list view, where multiple records can be edited at once, and the detail view, where just one record can be modified. The control retrieves field names and types from the assigned content module and initilizes the form engine. To reflect relations, a hierarchy of admin controls is created, each having an associated module. Before a record is finally stored, a preview of the resulting Web page can be created using a copy of the record. Records can be exported to and imported from a CSV file.

oPage can be customized and extended as needed. Existing objects can be configured in the factory. Additional and modified classes can be included. Extended actions can be started by catching events in a custom hook class. Additional languages can be activated as needed. To increase the performance, the generated pages or page fragments can be cached.

The oPage framework is a powerful base for building Web based content management systems.

4 Evaluation

In this chapter the goals of the oPage framework are matched against what has been accomplished. In section 2.6 on page 32 the requirements were listed.

4.1 Operating system

Web sites using oPage can run on every platform, where PHP and MySQL are available: Microsoft Windows, Linux, Mac OS X, Solaris, FreeBSD and others. ImageMagick is available for all mentioned operating systems. On platforms where ImageMagick is not available, the GD library of PHP is used instead.

Since all components are licensed as open source, it is (theoretically) possible to build binaries for unsupported operating systems.

oPage has been successfully installed on various Microsoft Windows and Linux systems running Microsoft Internet Information Server 6, Apache 1.3 and 2.0 and 2.2 and MySQL 4 and 5. Both PHP 4.x and PHP 5.x have been tested in CGI and fast CGI mode, and running as ISAPI and as Apache module. The differences of the platforms are hidden in the class CApp (see section 3.7 on page 63).

4.2 Installation

In section 3.17.1 on page 96 the steps for setting up the oPage framework have been described. This is easy enough for experienced users, but beginners may have troubles setting the right options in `custom/config.php`. A setup wizard and a tool to change the settings in the backend would make things easier and should be included in the future.

4.3 Web compatibility

The quality of Web pages depend heavily on the provided templates, but also on the content. The templates of the backend are standardized and used in every project. While creating these files, the resulting pages have been checked by the HTML Validator plugin of Firefox 2.

Frontend templates are created separately for each project. Although a set of template fragments is provided by the framework, the quality of the resulting Web page depends on the experience and accuracy of the Web developer. Authors may enter content containing HTML code, either directly or supported by the WYSIWYG editor. In both cases, typos and syntax errors may occur, which result in non compliant Web pages. Current browsers are fault-tolerant and manage to display the Web page even if it has errors. But less tolerant systems like screen readers or news readers, which often require valid XML code, may fail.

oPage provides a stable base of backend templates and template fragments for the frontend. Web sites using the framework have been successfully tested with a wide range of Web browsers like Microsoft Internet Explorer (4, 5, 6 and 7), Firefox (1 and 2), Opera (8 and 9) and Safari.

4.4 Search engine optimization

oPage offers a number of ways to optimize a Web site for search engines. There is a module to manage metatags for a Web site. URLs with parameters can be masqueraded as paths by changing the query and the parameter separator from `?` and `&` to `/`. Additional parameters like the title of the target page can be appended to an URL.

Further optimizations can be implemented by using a frontend controller and the Apache module *mod_rewrite*. TYPO3 comes with a more strict frontend and provides this feature. This has not been implemented in oPage so far since it would make oPage depend on Apache.

4.5 User interface

The user interface design in the backend is the same for all oPage projects. Template fragments provide the basic elements and are combined with page templates providing the common layout. For every button both the ALT and the TITLE attribute are set to explain their purpose. URLs in text blocks are identified automatically and converted into hyperlinks. The backend currently supports only German, but can be provided in different languages by creating the appropriate language resource files. For each text area, a WYSIWYG HTML editor can be activated.

Currently HTML input is not validated enough. To guarantee a consistent layout, which can not be broken, the system would have to detect errors in the content. It could be adequate to limit allowed tags and to check the content for XHTML validity.

4.6 User management

Management of backend user privileges is provided in three different modes. The default and simplest mode supports only one kind of users, which all have administration rights. In the second mode, global and per module rights can be assigned to users. Groups can be defined in the third variant, where users and rights are assigned to the defined groups.

For most oPage projects the simple mode has been sufficient. In a couple of projects e.g. `http://www.demmer.at` and `http://www.motorline.cc`, the variant with user, right and group management has been used. The same classes can also be used in the frontend.

Currently, rights are always assigned for a whole module. Record based access rights are currently not available, but would be a nice feature. A record should only be editable by the current owner. But this requires to change all backend form templates. One approach could be the use of AJAX in the list view. All the records are read-only by default. When the user tries to modify the record or wants to open the detail view, the browser makes a JavaScript call to check, if the user is allowed to do this operation. When the changes are submitted, the server has to check again, if the modifications are allowed.

4.7 Security

Web sites and Web applications have to be secure, only allowed operations should be possible and the content has to be secured against manipulation. Security concerns have to be taken into account on every level of the system stack, starting at the operating system up to the Web site's user frontend.

The operating system is usually managed by a system administrator, who is responsible to install and secure the system and its services, like the Web server and the database management system. Patches and updates need to be installed, permissions need to be set to block external intruders and to separate multiple Web sites from each other.

The rights in the file system have to be set to the minimum needed. On Web sites using oPage, the Web server only needs write permissions to the `cache/`, `static/` and the `temp/` directory. For any other files and directories, only read access is needed.

None of the security options can be changed within PHP scripts, the security functions of PHP should be turned on in `php.ini` or in the configuration of the Web server, where separate settings can be applied for each virtual server. oPage checks for PHP's *safe_mode* setting and acts appropriately.

For a long time (until PHP version 4.2.0), HTTP parameters have been converted into global variables by default. Using undeclared and uninitialized variables provided a nice way for intruders to manipulate a Web site. The PHP setting *register_globals* is usually off in current PHP versions, but it is always good to initialize variables with a default value. When oPage is used on a development system (see section 3.7 on page 63), the error reporting is turned on fully. When an uninitialized variable is detected, a warning is reported. If a critical error occurs, processing is stopped and the call stack is displayed.

Other security threats of Web applications are SQL-Injections[89] and cross-site scripting[90]. Most of the issues are caused by using unchecked parameter values within scripts of the Web site. In the provided guest book example (see listing 12 on page 80), the data submitted by the form is processed using the modules from action method, which returns an array of fields, that are used by the module, and values, where all HTML tags are stripped. All values, which are used in a database query, are automatically enclosed by single quotes and quotes are escaped with a slash.

4.8 Software architecture

oPage makes use of object oriented design patterns. The factory class is responsible for instancing the oPage classes and including the necessary source files. Foreign classes are integrated with a facade pattern. The system abstraction layer is instanced using the singleton pattern.

For each type of structured content, a module class is used, which is derived from a base class. It operates as a report generator and provides operations to create, retrieve, update and delete records. In a controller script, the various objects are tied together. The content is queried with the module and merged with a template through a template engine.

The database access is done with a database abstraction class (CDb). The system environment is also abstracted (CApp). Besides of the controller scripts, where procedural code is used, all other scripts use object oriented code.

Web sites using oPage are divided into the oPage core, the backend and the frontend. The core and the backend files can be shared between different projects, only the frontend is Web site specific. All customizations are usually done in the `custom/` directory.

oPage provides a number of template fragments with common elements, which can be used in the frontend to speed up development.

4.9 Application programming interface

oPage is used in a number of projects[91] of different complexity. Usually a Web designer creates a screen design, which a programmer takes and provides the configured and customized Web site using the oPage framework. Then a HTML developer modifies the default template files as needed, to meet the screen designs.

oPage can be customized at many places. The frontend layout is defined in a basic page template and in the module specific content templates. Modules can be configured in the custom factory. Additional project specific modules can be derived from standard modules or created from scratch. The project specific hook class can be used to perform additional actions when certain events occur.

oPage is designed to be flexible and adaptable for different needs. The framework has been extended and improved a lot since its first use in a project, nearly with every project new features where implemented, but still the concept itself proved to be flexible enough.

Integrating extensions is not very easy. Currently the files and the settings in the custom factory have to be installed by hand. Both the module files, the frontend controller and the administration interface for the backend could be packed together into one file and installed via a plugin interface in the backend.

4.10 Performance

A lot of effort has been made to get the best performance. Database queries have been reduced to a minimum, indexes have been created and the remaining queries where optimized. Analyzed templates are stored in the template cache. Fragments and complete pages can be cached as well.

[89] `http://en.wikipedia.org/wiki/SQL_Injection`
[90] `http://en.wikipedia.org/wiki/Cross-site_scripting`
[91] `http://www.opage.at`

For optimal performance, PHP should run either as an Apache module or as FastCGI module in Microsofts Internet Information Server, and be supported by a bytecode-cache like eAccelerator or Zend Optimizer to reduce the compileer overhead. The factory class includes source files only when needed (similar to the *autoloading* mode introduced with PHP 5), so only used files are loaded and have to be compiled.

On Web sites with performance issues, the trace mode can be activated, where slow database queries are documented. Another good place to look at is the slow query log of MySQL. Using PHP 5 and MySQL 5 instead of PHP 4 and MySQL 4 also brings a huge performance boost.

4.11 Lessons learned

At the first look the separation of logic, content and layout is not as easy as it seams. The difficult question is, how much logic is needed in the layout, speaking what kind of operations are put into the template engine and what stays in the controller script.

The template engine used in oPage replaces template variables with values. There are no operations to process an array of records, like in Smarty template engine[92] or PHPTAL[93]. In oPage the module assigns record per record to the template engine and calls the parse method for the required blocks. The controller script predefines the structure (the hierarchy of blocks, not the layout) of the template. For example the *article* module uses the sub modules *paragraph* and *picture*. In the template, the pictures are only available below the `paragraph` within the `article` block hierarchy (see section 3.8.8 on page 72 for an example). In some cases, where this concept is to inflexible, the controller script (the logic) can be modified to provide the data in a more convenient way.

The oPage's module hierarchy concept in combination with the template engine has been also implemented in C# and VBScript. The C# version is used on a Web site, which provides press and media review collections in a customer specific layout and design for more then 300 clients. The VBScript variant is used to query content from a database and to create HTML, XML, RTF, PDF, CSV and text files, which are further processed by other applications.

oPage has to be very flexible so that different types of frontends can be implemented. It should also support the reuse of existing features in new projects. This worked well so far in the backend, since the backend has the same structure in all projects. It's more difficult to reuse a guestbook or a forum and be flexible enough to implement project specific needs. A derived quest book module can implement project specific behavior by overwriting the form action methods of the base class (see section 3.14.1 on page 82). Module fields can be activated as needed in the factory. The existing form template fragments can be styled globally with cascading style sheets or with the style parameter of the `FILE` statement. A drawback of this flexibility is, that there are many places, where customizations can be implemented, in depth knowledge of the oPage framework is necessary to decide, which is the best solution.

The final architectural decision has to be made by the software developer, whether to use a central controller or a separate script for each area (like news, shop or guest book). Personally, I have switched to using a central controller script, which includes fragments containing the logic depending on the given parameters. It makes reusing of components a bit easier as well as URL rewriting for search engine optimization purposes.

[92] `http://www.smarty.net/`
[93] `http://phptal.motion-twin.com/`

5 Future Work

A software project like the oPage framework is usually never finished. New releases of the programming language PHP will offer advanced possibilities like aspect oriented programming[94]. Rising technologies like AJAX and Web services allow smoother user interfaces, but require a different application design.

Not all this cool new features can be used immediately, it takes some time till the updated infrastructure becomes available at the Internet service providers.

In this chapter some interesting points of improvement are described, which will make the oPage framework more useful.

5.1 Installation

Currently the installation of oPage has to by done manually. A zip archive has to be downloaded from the oPage distribution site and unzipped. Then the framework has to be configured by modifying `custom/config.php`. After all files have been uploaded to the Web space the correct permissions have to be applied.

A installation and configuration wizard could help users to install and configure the framework on their Web space.

First a PHP package (e.g. PHAR[95] or PHK[96]) can to be downloaded from the distribution Web site and uploaded to the users Web space. The archive can be opened in the browser, which starts the setup process. First it checks the system for compatibility, asks a view questions, installs the files and sets the correct file system permissions. After the installation is completed, the script should ask, if it should delete itself from the server. If something can not be done by the script (e.g. because of PHP security settings), the user has to be informed and the necessary commands have to be displayed.

In the backend a configuration Wizard could help editing the settings in `custom/config.php`. Therefor the Web server needs write access to the file, which may be a security issue. Another option may be a download link to the newly created file, which the user can use to replace the original file.

5.2 Template Engine

The current template engine supports logical operations on block names. Currently only `&` (and) and `|` (or) operators are supported and only one type of operator at a time. Here is space for improvement. The `!` (not) operator could be added to get blocks parsed when another block is not parsed.

```
1 <!-- !block1&!block2 -->
2 This text is displayed, if block1 and block2 are both *not* parsed.
3 <!-- END: !block1!&block2 -->}.
```

Combined logical operations would be useful as well.

```
1 <!-- block1&(block2|!block3) -->
2 This text is displayed, if block1 and block2 or block1 but not block3 are parsed.
3 <!-- END: block1&(block2|!block3) -->}.
```

The encoding and formating of values should be extended to provide a free use of combinations of encoding and formating parameters. `{variable|list /|size kb|number 2, |printf %10s#html}` would split the value into parts separated by `/`, which are later formated by the size operator and by the number operator. The result will be run through the printf statement before the result is returned to the list statement.

[94] http://phpaspect.org/
[95] http://www.php.net/phar/
[96] http://pecl.php.net/package/PHK/

Also a combination of `#text`, `#plain`, `#nbsp` and `#slashed`, etc. should be possible. `{variable#text#plain #slashed}` would convert the value into plain text, replace the newlines with spaces and escape the single quotes in the result with a slash.

5.3 Multilanguage resources

Currently the resource files are plain text files (section 3.5.8 on page 59). No support tools exist to create texts for different languages or to track changes and additional text entries. A proper standard format should be used instead, for which editor and management tools are available. Standards like gettext[97] or TMX[98] should be examined.

5.4 Automatic checking of external links

Every time a record is save, the system could look for links to external Web sites and run a check, if the target is available. The link, a reference to the content record, the HTTP result and a time stamp should be stored in a database table. When the content record is updated or deleted, the according rows in the external links table have to be updated as well. Based on the timestamp, checks can be limited to one per day. Once a week, all entries in the links table can be checked by a script started from a cron job or initiated by the frontend. The result could be sent by email to the Web master of the Web site (or to an email address set in the parameters table).

5.5 Access statistics

Currently the counter table contains only rudimental data, which are not properly analyzed.

The counter module could be extended to provide a graphical report for the collected data. It also could be extended to collect additional information like browser, client IP, access time and unique user ID. With this data reasonably charts for a page or the Web site can be created: Top 10 pages, number of unique users, visits, click paths, entry and exit pages, and in combination with GeoIP[99] even the location of the user can be reported.

A lot of this information is already provided by logfile analyzers like AWstats[100] or can be collected using Google Analytics[101], but the results would be more exact, when the framework tracks the access and creates the statistics.

5.6 Plugin System

Extensions to the framework could be provided as a PHAR file, which contains all the files and a setup script. A typical guestbook contains the module, templates for the backend list- and detail-view, resource files for language dependent texts, images, templates and a controller script for the frontend. A general setup script could be called to integrate the extension. It should be possible to extend the setup script by the plugin. An uninstall script should be provided as well. Also a wizard, which creates the basic structure of an extension would be interesting.

The module, the backend files and the frontend part should be separated and the dependences between and to other packages should be noted. When a package is installed, all the depending packages should be installed or updated as well. This makes it easier to create plugins, which reuse parts of other extensions. For example, the newletter system uses the person module to manage the recipients, the category module to group the entries and the news module for the newsletter content.

97 http://www.gnu.org/software/gettext/
98 http://www.lisa.org/standards/tmx/tmx.htm
99 http://www.maxmind.com/app/geolitecity
100 http://awstats.sourceforge.net/
101 http://www.google.com/analytics/

5.7 Workflow

Based on the user roles (see section 2.3 on page 13) according privileges could be set. A workflow could be defined for each module. E.g. an author is allowed to create a news entry, but does not have the permission to switch it online. With a new workflow button in the backend, the state of the record could be changed from new to review and owner of the record could be set to the next user in the workflow chain. The transition of the owner can be configured by rules, which depend on the data of the record. E.g. an editor may be only responsible for news items of a specific section of the Web site or for just a group of authors. All changes should be tracked in a workflow log.

5.8 Backend homepage

The backend start page is currently not properly used, only a welcome message is displayed. It would be a good location for Web site specific information: The number of users currently in the frontend, the number of backend users online, the number of active, perpared and inactive news items, the access statistics of the current day, week, month and year could be displayed. Also the open entries of a todo list or a list of new messages from the users inbox could be presented. The displayed blocks and the location on the screen should be customizable by each backend user and stored in the their profiles.

By using predefined rules, the system could display warnings on this page. E.g. “currently only two news items are displayed.”, “The newest news entry is older then five days.”, “There is one new order in the online shop.”, “You have five unhandled issues in the ticket system.”, “Seven responses to your messages are in the forum.”. Each of this warnings should be linked with the module page, where the relevant records are displayed.

5.9 Summery

With new projects new challenges will arise, which drive the development of oPage. The template engine syntax will become more powerful and the database layer may be changed to PHPs database objects (PDO[102]) library. The backend user interface may be redesigned and AJAX features should be added.

Currently, oPage still can be run using PHP4, since this version is widely used at internet service providers. With the discontinued development of PHP4, oPage will stop supporting PHP4. This will allow the use of all new object oriented features PHP5 is offering and will increase code quality and application security and stability.

[102] `http://www.php.net/pdo`

6 Conclusion

In this thesis I presented the oPage framework for Web based content management systems. It provides an infrastructure, which supports and speeds up the development process. It is more specific than a general purpose framework like ASP.NET, but more flexible than content management systems like TYPO3 or Joomla!.

oPage runs on any popular server platform, has a modular design and can be customized and extended in many ways. Logic, content and layout are separated. Web designers can create and modify layout templates without the need of PHP programming experience. Small PHP scripts provide the processing logic and glue together the framework objects. The page object provides the frame for the content objects, which are derived from the module and control base classes. All instances of classes are created by a factory. This is the first location to customize the framework, but many other places exist, where modifications and extensions are possible. The content is stored in a relational database (e.g. MySQL), each table is represented by a module class. Modules can be joined to a hierarchy to map database relations. Special modules exist to handle m:n relations. The frontend uses the modules to query and display the content. In the backend, the modules are used to configure the forms, and to create, retrieve, update and delete the records. The core and the backend files can be shared among different projects. The user interface in the backend is easy to use and similar for all modules.

A lot of effort has been made to get the best performance. By activating the cache, the performance can be increased even more.

Internationalization and localization of Web sites is supported. The framework provides an infrastructure to build multi language Web sites. The backend user interface can be made available in different languages by adding resource files.

oPage proved to be flexible. A number of Web sites have been created with the framework. All requirements could be fulfilled. Extensions to the framework have been made, but the basic concept stayed the same.

Further research need to be done to make oPage even more flexible. Installation is not as easy as it could be. The template engine could learn a few more tricks. External links should be checked regularly. The access statistics could have more details and should be presented on the start page of the backend. A plugin system and an extension manager, which handles installing, updating and removing of packages, would be a nice feature. Reusing modules and backend files is very easy, but the frontend scripts have to be adapted for every Web site. By defining a set of configurable features for frontend extensions would provide standardized packages, which could be reused in many projects. A configurable workflow engine would be a nice feature for enterprise Web sites.

A framework like oPage is never finished. New requirements arise, new technologies are invented and better design approaches are developed.

References

[AES02] Dave Addey, James Ellis, and Phil Suh. *Content Management Systems*. Peer Information Inc., 2002.

[AKM] AKM Autoren, Komponisten und Musikverleger. `http://www.akm.co.at`.

[Avg05] Paris Avgeriou. Architectural patterns revisited - a pattern language. In *Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 2005)*, pages D3 1–39, Irsee, Germany, July 2005.

[BHMH03] Dr. Peter Baumgartner, Mag. Hartmut Häfele, and Mag. Kornelia Maier-Häfele. Evaluation von content management systemen (studie kurzfassung). Österreichisches Bundesministerium für Bildung, Wissenschaft und Kultur, 2003.

[BL01] Paul Browning and Mike Lowndes. Content management systems. *JISC TechWatch Report*, September 2001.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[CDIW05] Jim Challenger, Paul Dantzig, Arun Iyengar, and Karen Witting. A fragment-based approach for efficiently creating dynamic web content. *ACM Transactions on Internet Technology (TOIT)*, 5(2), May 2005.

[Dem] Demmer Handelsgesellschaft. `http://www.demmer.at`.

[DH01] Yogesh Deshpande and Steve Hansen. Web engineering: creating a discipline among disciplines. *IEEE MultiMedia*, 8(2):82–87, April-May 2001.

[GM01] Athula Ginige and San Murugesan. Web engineering - an introduction. *IEEE MultiMedia*, 8(1):14–18, January-March 2001.

[Hac02] JoAnn T. Hackos. *Content Management for Dynamic Web Delivery*. Wiley Computer Publishing, 2002.

[JM02] Stefan Jablonski and Christian Meiler. Web-content-managementsysteme. *Springer Informatik-Spektrum*, 25(2):101–119, April 2002.

[KaC01] Othmar Kyas and Markus a Campo. *Internet Professionell*. mitp, 2001.

[Ker03] Clemens Kerer. *XGuide - Concurrent Web Development with Contracts*. PhD thesis, Technical University of Vienna, Austria, May 2003.

[KPRR04] Gerti Kappel, Birgit Pröll, Siegfried Reich, and Werner Retschitzegger. *Web Engineering*, chapter 1. Web Engineering. dpunkt.Verlag, 1 edition, 2004.

[Lei01] Helmut Leitner. Die Geschichte des Wiki Web. `http://www.wikiservice.at/wikiweb/wiki.cgi?DieGeschichteDesWikiWeb`, 2001. WikiWebAt.

[Low99] David Lowe. Web engineering or web gardening? *Webnet Journal: Column Engineering the Web*, 1(1), 1999.

[Mic04] Dimitrios Michelinakis. Open source content management systems: An argumentative approach. The University of Warwick, Warwick Manufacturing Group, 2004.

[MW94] Gunther Maier and Andreas Wildberger. *In 8 Sekunden um die Welt*. Addison Wesley, 1994.

[Nak01] Russell Nakano. *Web Content Management*. Addison-Wesley Professional, September 2001.

[oPa] oPage Homepage. `http://www.opage.at`.

[Ped04] Jan Keller Pedersen. Xoops comparison: A study of open source content management. IT University of Copenhagen, 2004.

[PHP] Php documentation. `http://www.php.net/manual/en/introduction.php`.

[RDF] Resource description framework (rdf). `http://www.w3.org/RDF/`. World Wide Web Consortium.

[RR01] Gunther Rothfuss and Christian Ried. *Content Management mit XML*. Springer, 2001.

[Wer05] Thomas Werres. Cms – potenziale und grenzen von typo3. Hausarbeit, Fachhochschule Koblenz RheinAhrCampus Remagen, 2005.

[Wöh04] Heiko Wöhr. *Web-Technologien*. dpunkt.Verlag, 2 edition, 2004.

[Wika] Wikipedia The Free Encyclopedia. Content management system. `http://en.wikipedia.org/wiki/Content_management_system`.

[Wikb] Wikipedia The Free Encyclopedia. Web Content management system. `http://en.wikipedia.org/wiki/Web_content_management_system`.

[WS00] Udo Winand and Jörg Schellhase. Content management systems. *wisu das wirtschaftsstudium*, 29(10):1334–1345, October 2000.

[ZTZ02] Oliver Zschau, Dennis Traub, and Rik Zahradka. *Web Content Management - Websites professionell planen und betreiben*. Galileo Business, 2 edition, 2002.

List of Tables

List of Figures

Listings

A Appendix

A.1 PHP

According to the PHP manual[PHP], "PHP (recursive acronym for 'PHP: Hypertext Preprocessor' or former 'Personal Home Page Tools') is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML."

Web sites:
Homepage: `http://www.php.net`
Description: `http://en.wikipedia.com/wiki/Php`

Usage:
As of July 2007, PHP is used at 21 million domains with 1.2 million IP addresses[103].

History:
The development of PHP started in late 1994 by Rasmus Lerdorf. Zeev Suraski and Andi Gutmans made a rewrite of the language engine and some of the most popular modules and helped to create PHP 3.0, which became broad available at ISPs (1998). Currently used are PHP Versions 4 and 5, developed mainly at ZEND.

Architecture:
PHP code is executed on the Web server. Both source code and precompiled code can be interpreted by the core language engine called ZEND engine. This engine can be called by the Web server as a module (SAPI) or as executable program using CGI. Many functions of PHP are part of the core engine, but more are implemented as extensions: e.g. database access, PDF functions, SSL, XML. PHP files use to have the file extension ".php".

Supported operating systems:
Linux, Mac OS X, Solaris, Unix, Windows

Supported Web servers:
Apache, Microsoft IIS, Caudium, Sun, iPlanet, Netscape servers and all Web servers supporting CGI.

Language features:
PHP can be embedded into HTML code. The language is very loose. Variables do not have to be declared and can hold any type of data. Object-oriented programming support was limited until version 4, with version 5 object-oriented functionality has been very much enhanced and is more robust and complete. PHP is fundamentally Web aware and has built-in support for accessing Web and FTP servers. Libraries for accessing SQL servers are also included.

Criticisms:
PHP is a scripting and dynamically typed language. It does not enforce the declaration and initialization of variables. The return value of the a function may be boolean (FALSE if the function fails) or integer (even 0 if the function succeeds, but which evaluates like FALSE). File system security is difficult, when PHP is used as a server module with Apache Web server.

A.2 Coding Standards

In reference to Pear Coding Standards[104], the following coding standards apply.

If it was hard to write it should be hard to understand... well. I like whitespace. I use spaces rather often. I also like new lines, empty lines and comments. I also have my own way treating default values for function arguments (see below).

[103] `http://www.php.net/usage.php`
[104] `http://pear.php.net/manual/en/standards.php`

These coding standards work for me since 20 years. I know there are other coding standards (e.g. Pear), but I prefer my own style, because I think the source code is more readable compared to Pear.

These are rather common coding conventions, which should be applied appropriately to the programming language used.

A.2.1 Indenting

I use an indent with 4 spaces, no tabs.

A.2.2 Control Structures

Please note the separate lines for "{", "}" and the extra spaces previous and after "(" and ")". This is in contrast to the pear standard, but it increases readability (at least in my opinion).

```
1  if ( ( condition1 ) || ( condition2 ) )
2  {
3      action1();
4      action2();
5  }
6  elseif ( ( condition3 ) && ( condition4 ) )
7      action3( 'with_parameter' );
8  else
9  {
10     echo 'using_default_action';
11     defaultaction();
12 }
```

Please note that "case" and "break" match at the same column and that there is a empty line between "break;" and the next "case".

```
1  switch ( condition )
2  {
3      case 1:
4          action1();
5      break;
6
7      case 2:
8          action2( 'param1', 'param2' );
9      break;
10
11     default:
12         defaultaction();
13     break;
14 }
```

A.2.3 Assignments

Although I love whitespace within source code, I don't like spaces to align assignments, because when a new assignment is added, which is longer then the ones already there, several lines have to be changed. Please note the spaces before and after the "=".

```
1  $iShort = 0x815;
2  $iLongVariable = 4711;
```

A.2.4 Function Calls

When a function is called, there is no extra space in front of "(" opening the arguments list. Since the ")" is the last character of the code, there is no extra space after ")".

```
1 $var = Foo( $bar, $baz, $quux );
```

A.2.5 Function Definitions

I use as default argument always null. This makes my life a lot easier when inheriting classes and overwriting methods. Please note the empty lines before the second "if" statement and the return statement. I treat the return statement like a function, so I call it with parentheses.

```
1  function Foo( $sArg1, $iArg2 = null )
2  {
3      if ( is_null( $iArg2 ) )
4          $iArg2 = 13;
5
6      if ( condition )
7      {
8          statement;
9      }
10
11     return( $val );
12 }
```

A.2.6 Comments

I use both single-line comments and multi-line comments. For automatic documentation creation, I also use PHPDoc style comments. To comment changes within the source I insert the current date and an abbreviation of my name consisting of the first two letters of my last name and the first letter of my first name.

```
1  /** PHPDoc comment for function declaration
2      @param: blahblah
3      @return: blahblah
4  */
5  Function Func( $sParam )
6
7  /** PHPDoc comment for variable declaration
8  */
9  var $sVar;
10
11 // Single-line inline comment
12
13 /*
14     2007.07.21 DOH:
15     Multi-line inline comment documenting some change
16 */
```

A.2.7 Header Comment Blocks

I don't like the idea of drawing frames around comments (like in Pear). It makes life a trap, when you rewrite the comment. One line of comment should not exceed around 76 to 80 characters. There's no hard

rule to determine when a new code contributor should be added to the list of authors for a given source file. In general, their changes should fall into the "substantial" category (meaning somewhere around 10% to 20% of code changes). Exceptions could be made for rewriting functions or contributing new logic.

Simple code reorganization or bug fixes would not justify the addition of a new individual to the list of authors.

All files should include modification comments to encourage consistency.

Use some tags to automatically track filenames and version numbers supplied by the source code repository software like Subversion, CVS or Microsoft Visual SourceSafe.

```
<?php
/* vim: set expandtab tabstop=4 shiftwidth=4: */

/*
    Program or library name
    Copyright (c) 1997-2007 by the copyright holder

    Some blah blah about the purpose of the file.

    Authors:
        2007-01-01 Original Author <author@example.com> (XXX)
        2007-07-01 Your Name <you@example.com> (YYY)

    $Archive: $
    $Author: $
    $Date: $
    $Revision: $
*/
?>
```

A.2.8 Include

I treat include statements as statements, not as functions. So the parentheses are omitted.

```
include_once 'filename1.php';
require_once 'filename2.php';
```

A.2.9 Code Tags

In PHP only <?php and ?> are used.

A.2.10 Example URLs

Use "example.com", "example.org" and "example.net" for all example URLs and email addresses, per RFC 2606[105].

A.2.11 Naming Conventions

Classes

Names of classes always start with the capital letter "C". To represent a hierarchy, use capital letters within the class name.

[105] http://www.faqs.org/rfcs/rfc2606

Here are some examples:

```
CPoll
CPollAnswer
CWebpage
CWebpageX
CWebpageCounter
```

Functions and Methods

Function names start with a capital letter. Private methods should start with a "_" to separate internal functions from public ones.

Constants

The names of constants should be capitalized.

```
define( 'CONST', 1 );
```

Variables

Variables (and function arguments) start with a lowercase letter identifying the the type of the variable. As far as I know this concept has been introduced as Hungarian notation at the beginning 90s in the last century. I adopted this concept as follows:

type	prefix	example
Array	a	aArray
Boolean	f	fYesNoTrueFalse
Constant	c	cConst
Date	d	dDate
Float	i	i
		iCurrency
Integer	i	i
		iNumber
Long	i	i
		iLongNumber
Object	o	oObject
Resources	o	oFile
String	s	sBuffer

Note, that I use a single letter "i" (or "j" or "k" if I need more) for loops, since "iI" or "iK" do not look that nice.

Global Variables

I use global variables only when necessary. E.g. for objects like oApp, oFactory, oDb or oParameter.

A.3 Directory structure

A.3.1 oPage core system

All files necessary files for both the administration system and the website are stored here. These files and directories are not intended to be changed by the Web master or editor. If you need to customize a module, you can do this in `custom/factory.php` and by creating a derived class in `custom/modul/`.

Table 6: *Directories of the core system*

directory	description
include/	Core system code
include/captcha/	Files to generate a CAPTCHA image (used in forms to avoid automated form submits - e.g. guestbook spam)
include/control/	Control classes like form, mail or pager
include/database/	Database classes for various database systems
include/email/	Class for sending emails
include/javascript/	JavaScript code for creating cookies, form validation or popup windows
include/modul/	Module classes like news or guest book
include/resource/	Language resources
include/template/	Template parts
include/tpl/	Template engine
cache/	Cached versions of templates and pages, must be writable by the Web server
counter/	Used to increase a counter
download/	Increase a counter and offer a file for download
popup/	Open an image in a separate window
redirect/	Increase a counter and redirect the user to another website
static/	All images and files related to modules, must be writable by the Web server
temp/	Import and export files are temporarily stored here

A.3.2 Administration system

This is the administration system for your website. These files and directories are not intended to be changed by the Web master or editor. If you want to add additional modules, you can change custom/admin/navigation.php.

Table 7: *Directories of the administration system*

directory	description
admin/	Administration system
admin/custom/	Base classes, template and style files
admin/custom/detail/	Template parts for detail pages
admin/custom/index/	Template parts for index pages
admin/custom/resource/	Language resources for administration system
admin/custom/template/	Template parts for pages
admin/home/	Default page of content frame
admin/images/	Images used in administration system
admin/images/buttons/	Buttons used in index and detail pages
admin/images/content/	Images used in index pages
admin/images/detail/	Images used in detail pages
admin/images/navigation/	Images used in navigation page
admin/images/preview/	Code for the preview system
admin/counter/	Administration pages for the standard module counter
admin/metatags/	Administration pages for the standard module metatags
admin/parameter/	Administration pages for the standard module parameter
admin/user/	Administration pages for the standard module user

A.3.3 Website

Feel free to create as many directories as you like as long as they do not interfere oPage core system or administration system directories. It is a good idea to use self-explaining directory names and separate

different sections of your website into different directories like `news/` or `guestbook/`.

Table 8: *Directories for the website*

directory	description
`custom/`	Base class, template and style files for the website
`custom/admin/`	Files to customize the administration system
`custom/error/`	Error page called in a state of emergency like no connection to the database
`custom/javascript/`	Website specific javascript files
`custom/modul/`	Derived or new modules
`custom/resource/`	Website specific language resources which can overrule core and administration resource entries
`images/`	Images for the website basic layout

A.4 Administration interface templates

Listing 31: *Sample administration template: Metatags (list view)*

```
<!-- BEGIN: main -->

<!-- BEGIN: admin -->
{FILE "admin/custom/index/#begin.tpl"}

<!-- BEGIN: admin_filter -->
{FILE "admin/custom/index/filter#begin.tpl"}
    {FILE "admin/custom/index/filter_title.tpl" name="id" align="left"}
    {FILE "admin/custom/index/filter_title.tpl" name="name" align="left"}
    {FILE "admin/custom/index/filter_title.tpl" name="abstract" align="left"}
    {FILE "admin/custom/index/filter_title.tpl" name="author" align="left"}
    {FILE "admin/custom/index/filter_title.tpl" name="revisit" align="left"}
{FILE "admin/custom/index/filter#between.tpl"}
    {FILE "admin/custom/index/filter_input.tpl" name="id" align="left"}
    {FILE "admin/custom/index/filter_input.tpl" name="name" align="left"}
    {FILE "admin/custom/index/filter_input.tpl" name="abstract" align="left"}
    {FILE "admin/custom/index/filter_input.tpl" name="author" align="left"}
    {FILE "admin/custom/index/filter_input.tpl" name="revisit" align="left"}
{FILE "admin/custom/index/filter#end.tpl"}
<!-- END: admin_filter -->

{FILE "admin/custom/index/#pager.tpl"}
{FILE "admin/custom/index/#buttons.tpl"}

{FILE "admin/custom/index/sort#begin.tpl"}
{FILE "admin/custom/index/#selectall.tpl"}
<!-- BEGIN: admin_sort -->
    {FILE "admin/custom/index/sort_title.tpl" name="id" align="left"}
    {FILE "admin/custom/index/sort_title.tpl" name="name" align="left"}
    {FILE "admin/custom/index/sort_title.tpl" name="abstract" align="left"}
    {FILE "admin/custom/index/sort_title.tpl" name="author" align="left"}
    {FILE "admin/custom/index/sort_title.tpl" name="revisit" align="left"}
{FILE "admin/custom/index/sort#between.tpl"}
    {FILE "admin/custom/index/sort_field.tpl" name="id" align="left"}
    {FILE "admin/custom/index/sort_field.tpl" name="name" align="left"}
    {FILE "admin/custom/index/sort_field.tpl" name="abstract" align="left"}
    {FILE "admin/custom/index/sort_field.tpl" name="author" align="left"}
    {FILE "admin/custom/index/sort_field.tpl" name="revisit" align="left"}
```

```
<!-- END: admin_sort -->
{FILE "admin/custom/index/sort#end.tpl"}

<!-- BEGIN: ROW -->
{FILE "admin/custom/index/edit#begin.tpl"}
    {FILE "admin/custom/index/edit_readonly.tpl" name="id" align="left" width=""}
    {FILE "admin/custom/index/edit_readonly.tpl" name="name" align="left" width=""}
    {FILE "admin/custom/index/edit_limited.tpl" name="abstract" align="left" width=""}
    {FILE "admin/custom/index/edit_input.tpl" name="author" align="left" width=""}
    {FILE "admin/custom/index/edit_input.tpl" name="revisit" align="left" width=""}
{FILE "admin/custom/index/edit#end.tpl"}
<!-- END: ROW -->

{FILE "admin/custom/index/#footer.tpl"}
{FILE "admin/custom/index/#end.tpl"}
<!-- END: admin -->

<!-- END: main -->
```

Listing 32: *Sample administration template: Metatags (detail view)*

```
<!-- BEGIN: main -->

<!-- BEGIN: admin -->
{FILE "admin/custom/detail/#begin.tpl"}

<!-- BEGIN: ROW -->
{FILE "admin/custom/detail/#header.tpl"}
    {FILE "admin/custom/detail/#language.tpl"}
    {FILE "admin/custom/detail/state#begin.tpl"}
    {FILE "admin/custom/detail/state#default.tpl"}
    {FILE "admin/custom/detail/state#end.tpl"}
    {FILE "admin/custom/detail/edit#begin.tpl"}
        <table border="0" cellpadding="4" cellspacing="4" width="100%" class="
            opage_fieldsection">
        <tr valign="top">
            <td class="opage_fieldarea">
                <table border="0" cellpadding="2" cellspacing="0">
                <tr valign="top">
          {FILE "admin/custom/detail/edit_input.tpl" name="name" colspan="2"}
                </tr>
                <tr valign="top">
          {FILE "admin/custom/detail/edit_input.tpl" name="author" width="100%" style="width
              : 100%;"}
          {FILE "admin/custom/detail/edit_input.tpl" name="revisit" style="width: 100%;"}
                </tr>
                </table>
            </td>
        </tr>
        <tr valign="top">
            <td class="opage_fieldarea">
                <table border="0" cellpadding="2" cellspacing="0" width="100%">
                <tr>
              {FILE "admin/custom/detail/edit_textarea#counter.tpl" name="abstract" style="
                  width: 100%;"}
                </tr>
                <tr>
```

```
34              {FILE "admin/custom/detail/edit_textarea#counter.tpl" name="description" style
                    ="width: 100%;"}
35                </tr>
36                <tr>
37              {FILE "admin/custom/detail/edit_textarea#counter.tpl" name="keywords" style="
                    width: 100%;"}
38                </tr>
39                </table>
40            </td>
41        </tr>
42        </table>
43    {FILE "admin/custom/detail/edit#end.tpl"}
44 {FILE "admin/custom/detail/#footer.tpl"}
45 <!-- END: ROW -->
46
47 {FILE "admin/custom/detail/#end.tpl"}
48 <!-- END: admin -->
49
50 <!-- END: main -->
```

Listing 33: *Modul block description*

```
1  <!-- BEGIN: main -->
2
3      <!-- BEGIN: modul#header -->
4          for subblocks see 'modul' block
5      <!-- END: modul#header -->
6
7      <!-- BEGIN: modul -->
8
9          <!-- BEGIN: empty -->
10         <!-- END: empty -->
11
12         <!-- BEGIN: GROUP_{group}_begin -->
13             <!-- BEGIN: _default_ -->
14             <!-- END: _default_ -->
15             <!-- BEGIN: {groupvalue} -->
16             <!-- END: {groupvalue} -->
17         <!-- END: GROUP_{group}_begin -->
18
19         <!-- BEGIN: HEADER -->
20         <!-- END: HEADER -->
21
22         <!-- BEGIN: DATA --> or <!-- BEGIN: DATA{iRow} -->
23
24             <!-- BEGIN: LEVEL{iLevel} -->
25
26                 <!-- BEGIN: ROW{iSection} -->
27
28                     <!-- BEGIN: COL{iCol} -->
29
30                         <!-- BEGIN: {modul_}{template}#begin -->
31                             for subblocks see '{modul_}{template}' block
32                         <!-- END: {modul_}{template}#begin -->
33
34                         <!-- BEGIN: {modul_}{template} -->
35
```

```
36                          <!-- BEGIN: {field}#empty --><!-- END: {field}#empty -->
37
38                          <!-- BEGIN: {field}#header --><!-- END: {field}#header -->
39                          <!-- BEGIN: {field}#begin --><!-- END: {field}#begin -->
40
41                          <!-- BEGIN: {field}={value}#begin --><!-- END: {field}={value}#
                                begin -->
42                          <!-- BEGIN: {field}={value} --><!-- END: {field}={value} -->
43                          <!-- BEGIN: {field}={value}#end --><!-- END: {field}={value}#end
                                 -->
44
45                          <!-- BEGIN: {field}<>0#begin --><!-- END: {field}<>0#begin -->
46                          <!-- BEGIN: {field}<>0 --><!-- END: {field}<>0 -->
47                          <!-- BEGIN: {field}<>0#end --><!-- END: {field}<>0#end -->
48
49                          <!-- BEGIN: {field}<>1 --><!-- END: {field}<>1 -->
50
51                          <!-- BEGIN: {field} --><!-- END: {field} -->
52
53                          <!-- BEGIN: {field}#end --><!-- END: {field}#end -->
54                          <!-- BEGIN: {field}#footer --><!-- END: {field}#footer -->
55
56                          <!-- BEGIN: {field}#notempty --><!-- END: {field}#notempty -->
57
58                          <!-- BEGIN: {field}#yes#begin --><!-- END: {field}#yes#begin -->
59                          <!-- BEGIN: {field}#yes --><!-- END: {field}#yes -->
60                          <!-- BEGIN: {field}#yes#end --><!-- END: {field}#yes#end -->
61
62                          <!-- BEGIN: {field}#no#begin --><!-- END: {field}#no#begin -->
63                          <!-- BEGIN: {field}#no --><!-- END: {field}#no -->
64                          <!-- BEGIN: {field}#no#end --><!-- END: {field}#no#end -->
65
66                      <!-- END: {modul_}{template} -->
67
68                      <!-- BEGIN: {modul_}{template}#end -->
69                          for subblocks see '{modul_}{template}' block
70                      <!-- END: {modul_}{template}#end -->
71
72                  <!-- END: COL{iCol} -->
73
74                  <!-- BEGIN: COL{iCol}_empty -->
75                  <!-- END: COL{iCol}_empty -->
76
77              <!-- END: ROW{iSection} -->
78
79          <!-- END: LEVEL{iLevel} -->
80
81          <!-- BEGIN: BETWEEN -->
82          <!-- END: BETWEEN -->
83
84      <!-- END: DATA --> or <!-- END: DATA{iRow} -->
85
86      <!-- BEGIN: FOOTER -->
87      <!-- END: FOOTER -->
88
89      <!-- BEGIN: GROUP_{group}_end -->
90          <!-- BEGIN: _default_ -->
```

```
91              <!-- END: _default_ -->
92              <!-- BEGIN: {groupvalue} -->
93              <!-- END: {groupvalue} -->
94          <!-- END: GROUP_{group}_end -->
95
96      <!-- End: modul -->
97
98      <!-- BEGIN: modul#footer -->
99          for subblocks see 'modul' block
100     <!-- END: modul#footer -->
101
102 <!-- END: main -->
```

A.5 CMS feature comparison

This table has been generated using data from CMS Matrix[106], a Web site dedicated to provide information on both commercial and open source content management systems.

[106] http://www.cmsmatrix.org

Product	Ariadne 2.4rc2	Drupal 4.7.2	eZ publish 3.x	Joomla! 1.0.7	Mambo 4.5.3	phpCMS 1.2.1pl2	Plone 2.1.1	SpinPike Commerce 3.3.5	TYPO3 4.0	Xaraya 1.0	Xoops 2.06
Last Updated	8/26/2004	6/ 7/2006	5/10/2006	4/25/2006	1/28/2006	8/31/2005	12/ 1/2005	4/16/2004	5/15/2006	11/ 9/2005	4/16/2004
System Requirements	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Application Server The application server or application environment required to run this CMS.	mod_php	PHP 4.3.3+	None	Apache recommended, any server that supports PHP and MySQL	PHP 4.1.2+	not needed	Zope		PHP 4.3.0+		None
Approximate Cost The approximate licensing cost of this CMS. Note that there are almost always hard and soft costs beyond licensing costs for any CMS.		Free		$0	Free		Free	$795	Free	Free / GPL	Free
Database The database engine this CMS uses to store content and settings.	PostgreSQL, MySQL, Oracle	MySQL, Postgres	MySQL, PostGreSQL, Oracle, MSSQL	MySQL	MySQL	Flat file (no database needed)	Zope	MySQL	MySQL, PostGreSQL, Oracle, MSSQL	MySQL, MySQLi, PostgreSQL, SQLite	MySQL 4.23.xx or later
License The type of license this CMS is distributed under.	GNU GPL	GNU GPL	GNU GPL	GNU GPL	GNU GPL	GNU GPL	GNU GPL	Proprietary	GNU GPL	GNU GPL	GNU GPL
Operating System The operating systems this CMS is compatible with.	Windows, Unix	Any	Linux, Windows	Any	Any	OS Independent	Any	Unix, Linux	Any	Any	Any
Programming Language The programming language that the CMS is written in and/or can be extended using.	PHP	PHP	PHP	PHP	PHP	PHP4 or PHP 5	Python	PHP	PHP	PHP	PHP 4.1.0 or later
Root Access Is root (or administrator) access required to install this application?		No	No	No	Yes	No	No		No	No	
Shell Access Is shell access required to install this application? In other words, do you need to be able to log in to the machine (other than through FTP) in order to install this application?		No	No	No	Yes	No	No		No	No	
Web Server The web servers this CMS is compatible with.	Apache, IIS	Apache, IIS	Apache	Apache	Apache, IIS, any PHP enabled web server, but Apache Recommended	Any php enabled server	Apache, IIS, Zope	Apache	Apache, IIS	Any php enabled server	Apache, IIS
Security	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Audit Trail Does the system keep track of who made additions, updates, or deletions?	Limited	Yes	Yes	No	No	No	Yes	No	Yes	Free Add On	No
Captcha A challenge-response system designed to defeat bots from being able to use user-only features of a system.		Free Add On	Free Add On	Yes	No	No	No		Free Add On	Free Add On	

See http://en.wikipedia.org/wiki/Captcha for more information.											
Content Approval Does the system provide for some level of system-wide content approval?	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Free Add On	Yes
Email Verification Does the system send an activation key to users to make sure they\'ve entered a valid email address?		Yes	Yes	Yes	Yes	No	Limited		Yes	Yes	
Granular Privileges Does the system allow read and write privileges on a per page or per content item basis, as well as separate privileges for other system functions?	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	
Kerberos Authentication Does the system support authentication via Kerberos?	No	No	No	No	No	No	Free Add On	No	No	No	No
LDAP Authentication Does the system allow for LDAP-based authentication?	Yes	Free Add On	Yes	Free Add On	Yes	No	Free Add On	No	Free Add On	Free Add On	Yes
Login History Does the system keep track of who logged in and when? Such systems also usually keep track of things like what browser the user was using and what IP address they came from, as well as unsuccessful attempts.	No	Yes	No	Yes	Free Add On	No	Free Add On	Limited	Yes	No	Free Add On
NIS Authentication Does the system support authentication via NIS?	No	No	No	No	No	No	Free Add On	No	No	No	No
NTLM Authentication Does the system support authentication via NTLM?	No	No	No	No	No	No	Yes	No	Free Add On	No	No
Pluggable Authentication Does the system allow an administrator to plug in additional authentication schemes (from a vendor or homegrown) beyond the default proprietary authentication scheme and an LDAP authentication mechanism?	Yes	Yes	Yes	No	Yes	No	Yes	No	Free Add On	Yes	No
Problem Notification Does the system provide a mechanism for alerting administrators (email, instant messenger, cell phone, etc) when it detects a problem? (Logging does not count.)	No	No	No	No	No	No	No	No	Yes	No	Limited
Sandbox Does the system allow for a private area for content managers to try new content ideas without the worry of affecting the rest of the site?	Limited	No	Yes	No	No	No	Yes	No	Yes	Free Add On	Limited
Session Management Does the system provide some facility for an administrator to see who is logged in, what they are doing, and log them out if necessary?	No	Yes	Yes	Yes	Limited	No	Free Add On	Limited	Yes	Yes	Limited
SMB Authentication Does the system support authentication via SMB?	No	No	No	No	No	No	Free Add On	No	Free Add On	No	No

SSL Compatible Can this system be used with an SSL certificate on the web server?	Yes	Yes	Yes	No	No	No	Yes		Yes	Yes	
SSL Logins Can this system be configured to switch to SSL mode (HTTPS) for logins, and then back to normal HTTP after the login? This kind of functionality protects user login information from being sniffed.	Yes	No	Yes	No	No	No	No		Yes	Yes	
SSL Pages Can this system be configured to switch to SSL mode for certain pages (or sections), and then back to straight HTTP for other pages (or sections)? You may want this if the system is used partially for regular site content and partially to distribute confidential data such as customer invoices or medical records.	Yes	No	Yes	No	No	No	No		Free Add On	Yes	
Versioning Does the system provide for some level of system-wide content versioning?	No	Limited	Yes	Yes	Limited	No	Yes	Yes	Yes	Free Add On	Yes
Support	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Certification Program Is there a professional certification or degree program for this CMS?	No	No	No	No	No	No	No	No	No	No	Yes
Code Skeletons Does the system provide code skeletons or code templates to make it easy for new developers to write plugins for it?		No	Yes	No	No				Free Add On		
Commercial Manuals Are there books or other commercially available documentation for this CMS?	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Limited	Yes
Commercial Support Can support be purchased from a commercial organization with trained staff members?	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Commercial Training Can training be purchased from a commercial organization that has dedicated training staff for this CMS?	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No
Developer Community Is there a free online developer community specifically for this product?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Online Help Is there an integrated context-sensitive help system built in to the CMS?	Limited	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	
Pluggable API Can the system be extended through an open and documented application programming interface (API)?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Professional Hosting Is there a vendor supplied professionally tuned											

hosting environment (application service provider) or has a certified hosting partner program.	Yes	Yes	Yes	Yes	Yes	No	Yes	Costs Extra	Yes	Yes	Yes
Professional Services Are there commercially available professional services organizations to customize or provide administrative services for this CMS?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Free Add On	Yes	Yes	Yes
Public Forum Is there a publicly available forum or message board for the system?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Public Mailing List Is there a publicly available mailing list for the system?	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Test Framework Does the system have an automated test framework that can be used to test the codebase to ensure that it is functioning properly? This sort of framework is sometimes called Unit Tests or Smoke Tests.		Free Add On	No	No	No				Free Add On		
Third-Party Developers Are there third-party developers who manufacture plug-ins for this system?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Users Conference Is there an annual users conference for this system where it's users can get together, discuss ideas, get training, etc?	No	Yes	Yes	Yes	Yes	No	Yes		Yes	No	
Ease of Use	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Drag-N-Drop Content Does the product allow the user to position content in a drag and drop fashion?	No	No	No	No	No	No	Free Add On	No	Free Add On	No	No
Email To Discussion Can messages be emailed to the system so that they automatically appear in community discussions (forums/message boards)?	No	Free Add On	No	Free Add On	No	No	Free Add On	No	Free Add On	Free Add On	No
Friendly URLs Does the system have human-readable and search engine friendly URLs? (The alternative is that there are a bunch of symbols and numbers in the URL and the URLs are typically quite long.)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Free Add On
Image Resizing Is the system capable of allowing users to resize uploaded images so they need not mess around with an external image editor?		Free Add On	Yes	Yes	No	No	Free Add On		Yes	No	
Macro Language Is there a macro language that allows content managers to place powerful functionality (like auto-generated navigation systems) without any programming knowledge?	No	Free Add On	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Limited
Mass Upload Does the system have a way of uploading/importing		Free Add	Free Add On	No	No	Yes	Yes		Free Add On	Free Add On	

many images and other files all at once to save time?		On									
Prototyping Does the system allow the user to create custom default settings for the different kinds of content objects to save them extra clicks when creating those object types. For instance, if there is a particular template they like to use, can they set it as a default?		No	Yes	Yes	No	Yes	No		Free Add On	Yes	
Server Page Language Is there a server page language available like PHP, JSP, or ASP for easy one-off functionality?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Spell Checker Does the system have an integrated spell checker?		Free Add On	No	No	No	No	Free Add On		Yes	Free Add On	
Style Wizard Does the system have a wizard for generating styles/themes/templates or whatever else a look and feel might be called in the system? These wizards would allow the user to step by step choose colors, layouts, logos, etc to create their own look and feel without knowing anything about HTML/CSS.		No	No	No	No				Yes		
Subscriptions Can the user subscribe to various sections of the site and receive notifications on new/updated content? These types of functions are typically found in the news feeds, forums, and blogs.		Free Add On	Yes	No	No	No	No		Free Add On	Free Add On	
Template Language Is there a templating language for powerful layout controls? (Note that templating languages are different from server page languages in that they are not capable of full programmatic functions, and are fully HTML compliant.)	No	Limited	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
UI Levels Is there a means of "dumbing down" the publishing interface for less sophisticated publishers while keeping a more sophisticated interface for power users?	Yes	No	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes
Undo Does the system allow users to "undo" operations if they make a mistake?	No	Limited	Yes	No	Yes	No	Yes	No	Yes	Free Add On	No
WYSIWYG Editor Is there a web-based rich text editor to allow publishers to create formatted content without knowing HTML, CSS, XML, or XSL?	Yes	Free Add On	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Free Add On	Free Add On
Zip Archives Does the system allow a user to upload a zip (or other compressed file) full of static content, which is then published to the site? This sort of feature is used to do web-based mass uploads of static content.		No	No	No	No				Free Add On		
Performance	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**

Advanced Caching Does the system have advanced caching mechanisms that go beyond simple page caching? For instance, navigation, template, or content object caching?	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes	
Database Replication Can the system take advantage of database replication for better scalability? The system would need to be able to perform reads from slaves and writes to the database master.	No	No	Yes	No	No	No	Costs Extra		No	No	
Load Balancing Does the system allow you to put a load balancer in front of it to split the load between multiple servers? This would require that user sessions can be passed beween all the nodes transparently.	Yes	No	Yes	No	No	No	Yes	No	No	Yes	Yes
Page Caching Does the system have a mechanism for caching the contents of a page so that if it\'s requested again it can skip most of the work to create the page?	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes	
Static Content Export Does the system have the ability to export it\'s content as static HTML so it may be served up from regional cache servers, or from static HTML web servers?		No	Yes	No	No	Yes	Free Add On		Free Add On	No	
Management	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Advertising Management Does the CMS have a banner or other management system?	No	Free Add On	Free Add On	Yes	Yes	Free Add On	Free Add On	No	Free Add On	Free Add On	Yes
Asset Management Is there a central repository for uploading images and other files so they can be reused through-out the site?	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Free Add On	Yes
Clipboard Is there a clipboard system that allows publishers to easily cut and paste content from one area of the site to another?	No	No	No	No	No	Yes	Yes	No	Yes	No	Yes
Content Scheduling Does the system allow for content to be automatically added or removed from a site based upon date?	Yes	Free Add On	Yes	Yes	Yes	Free Add On	Yes	No	Yes	Free Add On	Yes
Content Staging Can content be created on one server and easily "pushed" to another server?	Costs Extra	No	Yes	No	No	Yes	Free Add On	No	Free Add On	Yes	Limited
Inline Administration Is content edited directly in the page that it will be placed? (The alternative is that there is a wholly separate interface for managing content.)	Limited	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Online Administration Can the system be completely managed through a											

web browser? (The alternative is that there is some sort of offline client software that must be used to manage at least some components.)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Package Deployment Can content and applications be packaged so that tedious repeditive publishing functions can be easily deployed time and time again without the repetition?	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Sub-sites / Roots Does the system allow for sub-sites within a site that are self-contained with their own navigation and content hierarchy?	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Free Add On
Themes / Skins Does the system have a mechanism to transport styles, templates, etc between sites so that you can create a theme on one site and then reuse it on many others?	Limited	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Trash Is there a trash system to allow administrators or publishers to recover content that has been removed from the site? Note that this is not the same as recovering items from a versioning archive.	No	No	Yes	Yes	Yes	No	Free Add On	No	Free Add On	Free Add On	No
Web Statistics Does the system have built in web site statistics reporting for things such as pages/content items viewed, number of users per time period, etc?	No	Yes	Free Add On	Yes	Yes	Yes	Free Add On	No	Free Add On	Free Add On	Free Add On
Web-based Style/Template Management Is there a web-based interface for adding styles and templates to the system for design and layout control?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Web-based Translation Management Can language translations be managed through and easy web-based interface?	No	Yes	Yes	Free Add On	Yes	No	Free Add On	No	Yes	Yes	Yes
Workflow Engine Is there a full-fledged workflow system integrated into the CMS that can be used for business process management (or other duties beyond just content approval)?	Limited	Limited	Yes	No	No	No	Yes	No	Limited	Free Add On	No
Interoperability	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Content Syndication (RSS) Can the system export RSS/XML feeds so that the content can be republished on other sites? Note that if it can only syndicate content from one function, such a "news" then it should be labeled as only limited support.	Limited	Yes	Yes	Yes	Yes	Limited	Yes	No	Yes	Yes	Yes
FTP Support Does the system allow users to upload internal content and/or files via FTP?	Yes	Limited	No	Free Add On	No	Yes	Yes	No	Yes	No	Yes
UTF-8 Support											

Does the system support UTF-8 character encoding to enable multi-lingual sites without the use of seperate code pages for each language?	Yes	Yes	Yes	Limited	No	Yes	Yes		Yes	Yes	
WAI Compliant Does the system follow the W3C specification for WAI compliance?	No	Limited	Yes	No	Limited	Yes	Yes	No	Free Add On	Limited	No
WebDAV Support Does the system allow users to upload internal content and/or files via WebDAV?	No	No	Yes	No	No	Limited	Yes	No	No	Yes	No
XHTML Compliant Does the system follow the W3C specification for XHTML compliance?	No	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No
Flexibility	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
CGI-mode Support Can the system run in CGI mode for development purposes or on low-end systems?	No	Yes	No	No	No	Yes	Free Add On	No	Yes	Yes	No
Content Reuse Does the system allow content to be mirrored (not copied, but reused) from one location to another on a site?	Yes	Limited	Yes	Yes	Limited	Yes	Yes	No	Yes	Yes	Free Add On
Extensible User Profiles Does the system provide a user profiling that can be extended with new profile properties through an administrative interface?	Yes	Yes	Yes	Yes	Free Add On	No	Yes	No	Free Add On	Yes	No
Interface Localization Is the system localized/internationalized so it can be translated into other languages and take locale preferences like date/time preferences into account?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Metadata Does the system support the adding of arbitrary metadata properties to all the content objects? The metadata is typically then used for profiling, indexing, or even auxillary display functions.	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes	
Multi-lingual Content Does the system support the creation of sites with multiple languages?	Yes	Yes	Yes	Free Add On	Free Add On	Free Add On	Free Add On		Yes	Yes	
Multi-lingual Content Integration Does the system support a multi-lingual version of each content object without republishing the content object. For example, if you create an FAQ in English, then all that needs to be done to display the FAQ in Spanish or another language is to translate the content, not create another page with another FAQ content object. Then depending upon user preferences it either shows one version or the other.	Yes	Free Add On	Yes	Free Add On	Free Add On	Free Add On	Free Add On		Yes	Yes	
Multi-Site Deployment Is the system capable of hosting multiple sites from one software deployment? This means you can install											

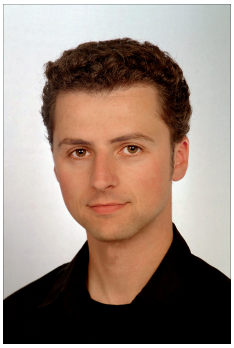
the software once and host as many sites as you want. It also means that when it comes time to upgrade you only need to upgrade the software in one place, not for each seperate site.	Yes	Yes	Yes	Free Add On	Free Add On	Yes	Yes		Yes	Yes	
URL Rewriting Is the system capable of rewriting URL or working with some other URL rewriting mechanism to provide shorter or friendlier URLs?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Wiki Aware Does the system support wiki or wiki-like functionality? Wiki provides online collaboration functionality as well as a simple text formatting language.	No	Free Add On	Yes	Free Add On	Limited	Free Add On	Free Add On		Free Add On	Limited	
Built-in Applications	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**
Blog Does the systemm have a blog or web log? (See slashdot.org for an example.)	Free Add On	Yes	Yes	Yes	Yes	Free Add On	Yes	No	Free Add On	Free Add On	Free Add On
Chat Does the system have an application for real-time online chat?	No	Free Add On	No	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	Free Add On	Free Add On
Classifieds Does the system have a classifieds application?	No	No	Free Add On	Free Add On	Free Add On	Free Add On	No	No	Free Add On	No	Free Add On
Contact Management Does the system have a contact management or rolodex type of application?	Yes	Free Add On	Yes	Yes	Yes	Free Add On	Free Add On	No	Free Add On	Free Add On	Limited
Data Entry Does the system have an application for creating arbitrary data entry applications?	No	Free Add On	No	Free Add On	Free Add On	Free Add On	Free Add On	No	Limited	Free Add On	Free Add On
Database Reports Does the system have an application for creating database reports?	No	No	Limited	Free Add On	Free Add On	Free Add On	Limited	No	Free Add On	Free Add On	Free Add On
Discussion / Forum Does the system have a message board?	Free Add On	Yes	Yes	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On	Yes
Document Management Does the system have an application for managing offline document storage and versioning?	No	Limited	No	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On	Free Add On
Events Calendar Does the system have an application for tracking events and displaying events calendars?	Yes	Free Add On	No	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On	Free Add On
Events Management Does the system have a way to create events and allow users to sign up for those events.		No	No	No					Free Add On		
Expense Reports Does the system have an application for tracking employee expense reports?	No	No	No	Free Add On	No	Free Add On	No	No	Free Add On	No	No
FAQ Management							Free				

Does the system have an application to organize frequently asked questions?	No	Yes	Free Add On	Yes	Yes	Free Add On	Add On	No	Free Add On	Free Add On	Yes
File Distribution Does the system have an application for distributing files including privileges for who is allowed to view/download those files?	Yes	Free Add On	Yes	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On	Free Add On
Graphs and Charts Does the system have an application that will allow the user to generate graphs and charts based upon some data set (SQL, text file, xml file, etc)?	No	No	Free Add On	Free Add On	Free Add On	Free Add On	No		Free Add On	Free Add On	
Groupware Does the system have email and calendaring (group scheduling) applications?	No	Free Add On	No	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	Free Add On	No
Guest Book Does the system have a guest book or graffiti application?	No	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	Free Add On	Free Add On
Help Desk / Bug Reporting Does the system have an application for trouble ticketing or bug reporting?	No	Free Add On	No	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	Free Add On	No
HTTP Proxy Does the system have a mechanism to proxy or mirror HTML and other content and applications from other web servers?	Limited	No	No	No	No	Free Add On	No	No	No	No	Yes
In/Out Board An intranet application that allows staff to post their status. In the building. Out for the day, be back tommorrow. Etc.	No	No	No	No	No	Free Add On	No		No	No	
Job Postings Does the system have a mechanism for posting job listings?	No	Free Add On	Yes	Free Add On	Free Add On	Free Add On	No	No	Free Add On	Free Add On	Free Add On
Link Management Does the system have an application to manage links?	No	Free Add On	Yes	Yes	Yes	Free Add On	Free Add On	No	Free Add On	Free Add On	Yes
Mail Form Does the CMS have an application for creating customizable 'contact us' type forms?	No	Free Add On	Yes	Yes	Yes	Free Add On	Free Add On	No	Yes	Free Add On	Yes
Matrix Does the system have a matrix application similar to what you see here on CMS Matrix?		No	No	No	No				Free Add On		
My Page / Dashboard Does the CMS have a dashboard application (sometimes called a portal)? (See my.yahoo.com for an example.)	No	Free Add On	Limited	No	No	Free Add On	Limited	No	Free Add On	Yes	Yes
Newsletter Does the system have the ability to allow users to add/delete themselves to/from a list so that they can be sent email from the system on various topics?		Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On		Yes	Free Add On	
Photo Gallery		Free Add					Free				Free Add

Does the system have an application for displaying a thumbnail / image repository?	Yes	On	Yes	Free Add On	Free Add On	Free Add On	Add On	No	Free Add On	Free Add On	On
Polls Does the system have an application for conducting simple single question polls?	No	Yes	Yes	Yes	Yes	Free Add On	Free Add On	No	Free Add On	Free Add On	Yes
Product Management Does the system have an application for displaying organized product information?	No	Free Add On	Yes	Yes	Free Add On	Free Add On	Yes	Yes	Free Add On	Free Add On	Free Add On
Project Tracking Does the system have an application for managing project tasks?	No	Free Add On	No	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	Free Add On	Free Add On
Search Engine Does the system have an integrated search engine that can index the managed content and allow the user to search the indexed content?	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Free Add On	Free Add On	
Site Map Can the system generate a tree showing all of the pages in the system dynamically so it doesn\'t have to be maintained seperately by the content managers?		Free Add On	Yes	Free Add On	Free Add On	Yes	Free Add On		Yes	Free Add On	
Stock Quotes Does the system have an application for displaying stock ticker information?		Free Add On	No	No	No				Free Add On		
Surveys Does the CMS have an application for conducting complex multi-question surveys?	No	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Costs Extra	No	Free Add On	Free Add On	No
Syndicated Content (RSS) Does the CMS have an application for retrieving and displaying RDF/RSS/XML syndicated content?	No	Yes	Yes	Yes	Yes	Free Add On	Free Add On	No	Free Add On	Free Add On	Yes
Tests / Quizzes Does the system have an application for administering tests and quizzes?	No	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	No	No
Time Tracking Does the system have an application for tracking employee time for payroll or billing purposes?	No	No	No	No	No	Free Add On	No	No	Free Add On	No	No
User Contributions Does the CMS have a system for allowing a user community to contribute stories and other content to the site?	Yes	Yes	Yes	Yes	Yes	Free Add On	Yes	No	Yes	Free Add On	Yes
Weather Does the system have a weather information system?		No	No	No	No				Free Add On		
Web Services Front End Does the system have an application for directly interfacing with arbitrary web services such as the Google API and the various available methods from X-Methods, and then creating a templated user interface without coding?	No	No	No	Free Add On	No	Free Add On	No	No	Free Add On	Yes	Free Add On
Commerce	**Ariadne**	**Drupal**	**eZ publish**	**Joomla!**	**Mambo**	**phpCMS**	**Plone**	**SpinPike Commerce**	**TYPO3**	**Xaraya**	**Xoops**

Affiliate Tracking Does the system have an affiliate tracking or referrals system for tracking partner sites that link in to the site?	No	Free Add On	No	Free Add On	No	No	No	No	Free Add On	Free Add On	Yes
Inventory Management Does the system provide the site owner a way to manage inventory levels (quantities of product on hand)?		Free Add On	No	Free Add On	No	No	No		Free Add On	No	
Pluggable Payments Does the system allow the site owner to plug in new payment gateways so they may use whatever payment processing mechanism (PayPal, PayFlowPro, 2checkout, iTransact, Authorize.net, etc) they choose?		Free Add On	Yes	Free Add On	No	No	Free Add On		Free Add On	Free Add On	
Pluggable Shipping Does the system allow the site owner to plug in their own shipping calculator so that they can charge based upon their needs (weight, price, UPS, USPS, FedEx, etc)?		Free Add On	Yes	Free Add On	No	No	Free Add On		Free Add On	No	
Pluggable Tax Does the system allow the site owner to plug in new tax calculators?		No	Yes	Free Add On	No	No	Free Add On		No	No	
Point of Sale Does the system tie into (or have built in) a point of sale system to enable site owners to run a brick and mortar store from the same inventory?		No	No	Free Add On	No	No	No		No	Free Add On	
Shopping Cart Does the system have a mechanism for allowing the user to create a list of adhoc items to purchase, and then purchase all of them at once?	No	Free Add On	Yes	Free Add On	Free Add On	No	Free Add On	Yes	Free Add On	Limited	Free Add On
Subscriptions Does the system have a way manage timed subscriptions that are tied into the commerce function? This requires recurring billing, billing cancellation, etc.		Free Add On	No	Free Add On	No	No	No		No	Limited	
Wish Lists Does the system allow users to create wish lists?		No	Yes	Free Add On	No	No	No		Free Add On	No	

B About the Author



Ing. Hannes Dorn
born 21st of November, 1972 in Amstetten, Austria
living and working in Vienna and Klagenfurt

B.1 Education

June 1992 — **Matura**, HTL St. Pölten, Austria
Secondary college for information technology and organization.
Degree with excellent success

March 2008 — **Master's degree in Business Informatics**
Technical University of Vienna, Austria

Master's Thesis:
"oPage - Framework for Web based Content Management Systems" carried out at the Technical University of Vienna, Distributed Systems Group

B.2 Professional career

1992 - 1999 — **SPARDAT Sparkassen Datendienst**, Vienna
IT services for Austrias mutual savings banks
Software engineer and manager of Intranet projects.

1999 - 2000 — **VIANET AG**, Vienna
Internet Service Provider
System maintenance and Web development.

2000 - 2002 — **IBIT GmbH**, Vienna
Web agency, software development and system maintenance
Technical director, Web development.

October 2002 — **EDV Dienstleistungen Dorn**, Vienna
`http://edv.dorn.cc`
Self-employed IT consultant and software engineer for system maintenance, software and Web development.
Building Web sites using the oPage framework.

This master thesis has been produced using LaTeX.