



WeberknechtCMS User Manual

Matthias Püski

August 2009

Contents

1	Introduction	3
1.1	Weberknecht Components	3
1.2	Enterprise or standard	4
2	Installation	5
2.1	Prerequisites	5
2.2	Configuration	5
2.2.1	Using PHP	6
2.3	Server startup and installation of the web UI	6
2.4	Post installation steps	9
2.5	Developer installation	9
2.5.1	Introduction	9
2.5.2	Prerequisites	10
2.5.3	Java installation	10
2.5.4	Eclipse Installation	10
2.5.5	Tomcat installation	11
2.5.6	MySQL installation	11
2.5.7	Get the code	11
2.5.8	Final adjustments	13
3	First Steps	17
3.1	Creating contents	18
4	Using plugins	25
4.1	Introduction	25
4.2	Using your first plugin	25

<i>CONTENTS</i>	1
5 Using templates	29
5.1 Introduction	29
5.2 Creating your first template	31
5.3 Defining the page header	32
5.4 Adding the body part	32
6 Managing page menus	35
7 Import and export of data	37

Chapter 1

Introduction

Hello and welcome to the Weberknecht User Manual.

First of all I should clarify, what WeberknechtCMS actually is.

WeberknechtCMS is in fact a toolsuite, which helps the web enthusiast to create manage and develop a more or less dynamic site. While WeberknechtCMS Manager allows you to create, manage and keep track of your content and your internet site, Weberknecht Server ist the server side part of the suite.

WeberknechtCMS ist an Open Source CMS, which has been developed on the base of Apache Struts 1.27. Since today the size of the source code has been grown to more than 150 000 lines of code. Development has begun 6 years ago and it started as a fun project. (As most of them do.)

1.1 Weberknecht Components

Weberknecht contains of two modules.

1. Weberkecht Server

The server is the core of the system. It contains the database, the page processing and the web administration page. The image below shows the global architecture of the server. Actually the system is far more complex than this picture can show, but like this you get an idea how this stuff works.

2.Weberknecht Manager

This is the Graphical User Interface (GUI) for the System. It basically runs standalone and serves as an integrated development environment for static and dynamic webpages. Thus

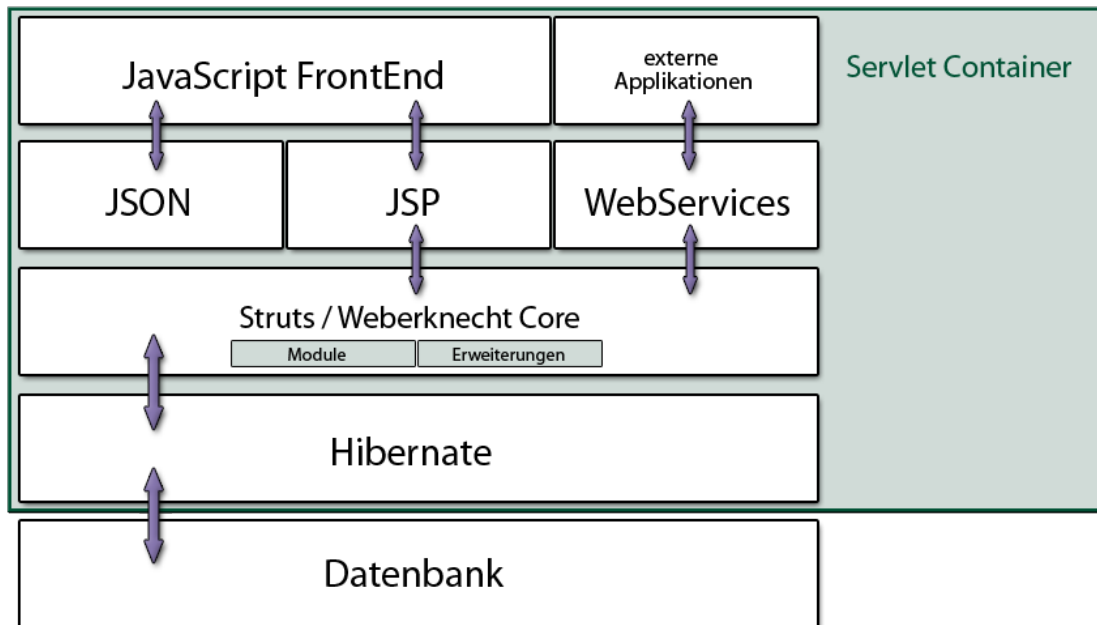


Figure 1.1: Weberknecht architecture

you may create and edit HTML pages locally and save them either as ordinary static HTML page or publish them to the server.

1.2 Enterprise or standard

The enterprise version of the system already contains a servlet container and the database which run on the local machine. The enterprise version thus allows several scenarios:

- Edit content locally
- Test content on the local webserver
- Publish the locally tested content to the server
- Generate static exports of dynamic contents

The standard version works completely standalone and helps you edit your pages and publish them via ftp for example.

Chapter 2

Installation

The first thing you need to do, is to decide which variant of the system you want to use. I will start to describe how the server is going to be installed. In the next section you will learn how to install the manager and connect it to your server.

2.1 Prerequisites

For the server installation your system must fullfil the following requirements:

- Java Runtime Environment 1.6.10 or later
- MySQL Server Version 4.1 or later, 5.0 is recommended
- Apache Tomcat Version 5.5.27 or later
- At least you will need 384MB of main memory on your machine, 512MB is recommended
- Mozilla Firefox 2.0 and later is mandatory, the web administration does not work with MS-IE

2.2 Configuration

If everything above is installed and configured, you may start with the installation process. First of all you need to download the server distribution. Unpack the downloaded zip file into the tomcat webapps folder.

Now you have to create your database, I assume you name it **wkcms**

the next thing to do is to configure the database connection. Navigate to the file **applicationContext.xml** inside the WEB-INF folder of the distribution.

Locate the following text:

```
<bean id="myDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" . . .
<property name="user" value="username" />
<property name="password" value="password" />
<property name="driverClass" value="com.mysql.jdbc.Driver" />
<property name="jdbcUrl" value="jdbc:mysql://localhost:3306/wkcms" />
```

Listing 2.1: Change the data source

Now change the two properties **username** and **password** for the database to your needs. If you did not name the database **wkcms** you must adjust the last line too.

2.2.1 Using PHP

Weberknecht server is capable of processing PHP pages. If you want to be able to use the image editor inside the web module, the PHP configuration is absolutely necessary.

Locate the file **web.xml** inside the WEB-INF folder of the distribution and open it.

Adjust the following lines

```
<init-param>
  <param-name>php.executable </param-name>
  <!--
    Enter the full qualified path to your php executable here
    e.g. c:/Programme/php/php.exe

    For php5 on windows please use the php-cgi.exe!
    -->
  <param-value>/usr/bin/php5-cgi </param-value>
</init-param>
```

Listing 2.2: Change the data source

and let the value `<param-value>` point to your php-cgi executable. Now you're done with the server configuration.

2.3 Server startup and installation of the web UI

Let's startup your tomcat and proceed with the last step of the installation process.

Open your Browser and point it to the following URL:

`http://yourserver.com:8080/wkcms`

Hit the **Install** button and the following screen will appear:

The image shows the 'Systemkonfiguration' (System Configuration) screen of the WeberknechtCMS. At the top, there is a banner image of a forest with the text 'WeberknechtCMS' overlaid. Below the banner is a wrench icon and the title 'Systemkonfiguration'. The main text explains that the user can configure the system here and that the system must be fully configured before use. Below this is a form with various configuration fields. The fields are: 'Seitenbezeichnung' (WeberknechtCMS), 'Hostname' (localhost:8080), 'Lokaler Pfad' (wkcms/), 'SMTP Server' (mail.gmx.net), 'Mail Absender' (pueski@gmx.de), 'SMTP Benutzername' (pueski@gmx.de), 'SMTP Passwort' (empty), 'Basispfad' (/home/mpue/devel/apache-tomcat-5.5.17/webapps/wkcms/), 'Admin eMail' (pueski@gmx.de), and 'Admin Passwort' (admin). At the bottom right is an 'INSTALLIEREN' button.

Seitenbezeichnung	<input type="text" value="WeberknechtCMS"/>
Hostname	<input type="text" value="localhost:8080"/>
Lokaler Pfad	<input type="text" value="wkcms/"/>
SMTP Server	<input type="text" value="mail.gmx.net"/>
Mail Absender	<input type="text" value="pueski@gmx.de"/>
SMTP Benutzername	<input type="text" value="pueski@gmx.de"/>
SMTP Passwort	<input type="text"/>
Basispfad	<input type="text" value="/home/mpue/devel/apache-tomcat-5.5.17/webapps/wkcms/"/>
Admin eMail	<input type="text" value="pueski@gmx.de"/>
Admin Passwort	<input type="text" value="admin"/>

INSTALLIEREN

Figure 2.1: Weberknecht installation

Just leave all values as they are, we'll change them later on. Since this screen is legacy it will be removed in the next release. After successful installation, hit the button **Hauptmenue** and you will be directed to the login screen. (Fig. 2.2)

Now you can login with username and password both **admin** and the screen in figure 2.3 will appear.

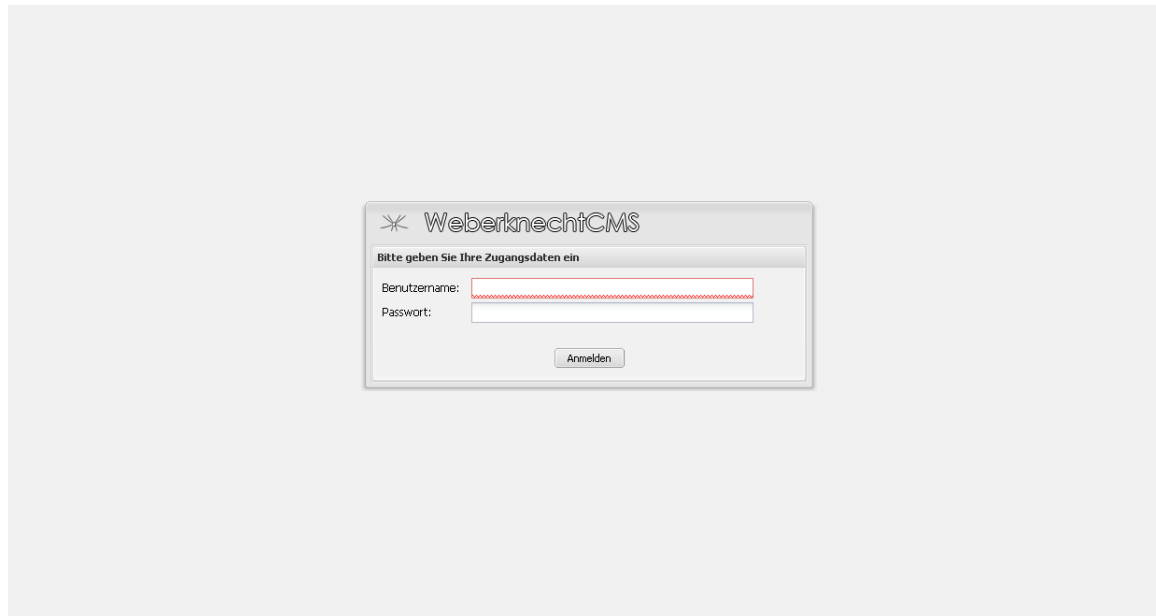


Figure 2.2: Weberknecht login

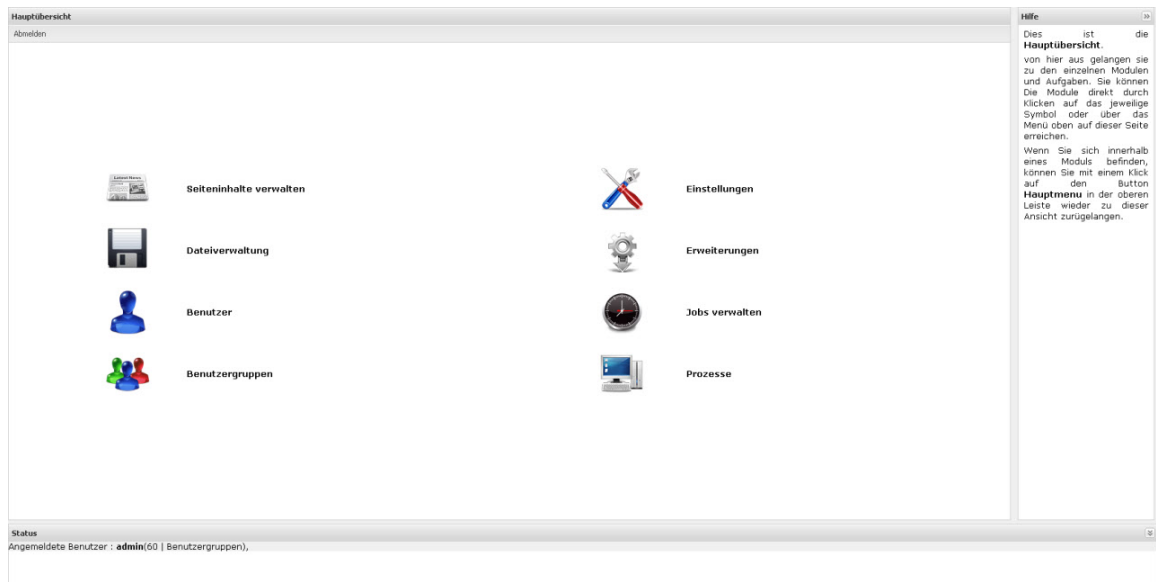


Figure 2.3: Weberknecht mainscreen

2.4 Post installation steps

It is very important, that, after successful installation the automatic database schema creation will be disabled, since the database would be created every time your tomcat starts up. To disable it open the applicationContext.xml again and locate the following lines:

```
<property name="hibernateProperties">
    .
    .
    <prop key="hibernate.hbm2ddl.auto">create </prop>
    .
```

Listing 2.3: Disable database autocreate

and comment the create line out, so that it looks like this

```
<property name="hibernateProperties">
    .
    .
    <!--
    <prop key="hibernate.hbm2ddl.auto">create </prop>
    -->
    .
```

Now your done. On next restart of the tomcat, it will not overwrite your existing database anymore.

2.5 Developer installation

2.5.1 Introduction

The first question you might ask is: “Why might I want to be a weberknecht developer?”?

There are several answers:

- You are a Java programmer and want to learn something about content management and Java web applications.
- You are a PHP developer and you want to programm web UI stuff and extensions for Weberknecht
- You want to write a Weberknecht plugin
- You want to contribute to the Weberknecht developers Team.

In any case and in order to be able to develop on WeberknechtCMS you will need some additional steps. First of all you will need a suitable development environment (IDE). The preferred IDE for the team is the well known Eclipse platform. Thus the next chapters fit for the installation with eclipse but other IDEs such as Netbeans or IntelliJ may suit as well. The installation steps may differ though. So let's start with the next chapter, which describes the developers installation in detail.

2.5.2 Prerequisites

As mentioned before you will need Eclipse and for completeness you will need the following components:

- Sun Java JDK 1.6.10 and later
- Eclipse IDE 3.4 and later
- Apache Tomcat 5.5.17 and later (Jetty might work as well)
- A MySQL(5.0 and later), PostgreSQL or a HSQL database(1.7 and later)

2.5.3 Java installation

Download the latest JDK for your platform from <http://java.sun.com> and install it. For simplifying purposes you should add some environment variables and extend your system path.

Depending on the operating system it differs how the variables must be set. In general you should do the following things:

- Add a new path entry which points to the `jdk\bin` folder so that you can type “`javac`” from the console and the java compiler will run
- add a new environment variable called `JAVA_HOME` which points to the jdk Folder (in windows its something like `SET JAVA_HOME=c:\path_to_your_jdk`)

2.5.4 Eclipse Installation

First, download eclipse. You will need to download the “Eclipse distribution for J2EE developers” for your platform. Just take a look at the <http://www.eclipse.org> site and jump right to the download site. Now extract the zip-archive to a folder of your choice and create a link to the eclipse executable. If you just don't get what I'm talking about, you might probably be reading the wrong manual.

2.5.5 Tomcat installation

Tomcat is the heart of the system. It's the servlet container which hosts the Weberknecht web application. You can get the latest version at

<http://apache.prosite.de/tomcat/tomcat-5/v5.5.28/bin/apache-tomcat-5.5.28.zip>

(Linux users may alternatively download the .tar.gz archive.)

After download unzip the tomcat archive to a folder of your choice and create two shortcuts to the startup and shutdown scripts. In windows these two are named startup.bat and shutdown.bat. Unixoid operating systems use the files named startup.sh and shutdown.sh. They are needed - as you might already have guessed - to start and to stop the tomcat. To start up the server use your shortcut. Windows makes it a little bit easier to see what the server is like to tell us, since it already pops up a console window when you click on the startup shortlink.

If you are using a Unix (clone) you might start the server from the console (If you don't know what a console ist, you might probably using the wrong OS). In order to view the server ouput, change to the tomcat\bin folder and type

tail -f ../logs/catalina.out &

The tail process will now spit out all new log messages from the tomcat.

2.5.6 MySQL installation

I will not describe the MySQL installation in detail. First of all you have to go to <http://www.mysql.com> and download the current version of the community server for your operating system. After succesfull installation create a new database for example **wkcms** and create an according user with has full access to the database. For details consult the MySQL reference manual.

Now after everything has been set up, it's time to

2.5.7 Get the code

Right now there's no anonymous CVS access, thus you got a username and a password from the project admin, if not drop a line to pueski@gmx.de and ask to contribute.

Open your eclipse and switch to the CVS repository exploring perspective (Window -> Open perspective -> other -> CVS) you will see a screen as shown in figure 2.4

Right click inside the left window "CVS repositories" and choose "New->Repository location". (see figure 2.5)

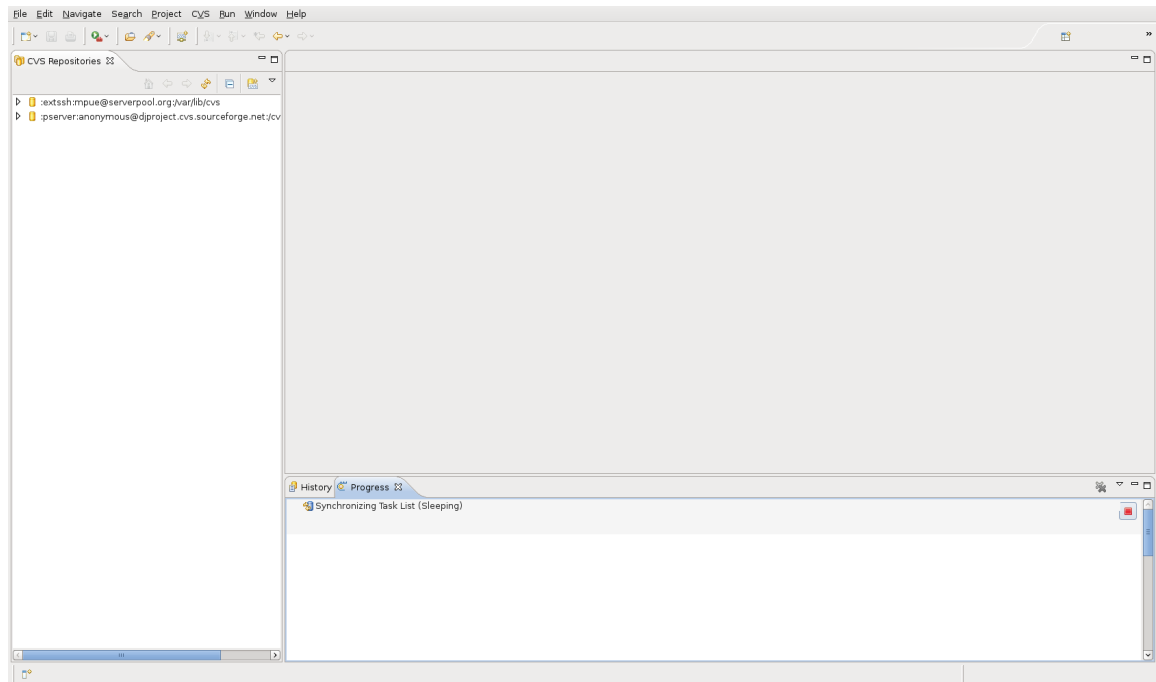



Figure 2.4: CVS Perspective

Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view



Location

Host:

Repository path:

Authentication

User:

Password:

Connection

Connection type:

☒ Use default port

☐ Use port:

☒ Validate connection on finish

☐ Save password (could trigger secure storage login)

[Configure connection preferences...](#)

Figure 2.5: Create a CVS Repository

Fill in the values as you were instructed from the eMail you got. Unless the eMail tells you something different you should enter the following:

- host : serverpool.org
- repository path : /var/lib/cvs
- connection type : extssh

and of course your username and password. Now click “finish” and you’re done.

The next thing to do is to check out the code. To do this, expand the newly created repository on the left side. If there’s nothing inside you might try right mouse->refresh repositories. If there’s still nothing I can’t help you. You should now see something like in figure 2.6

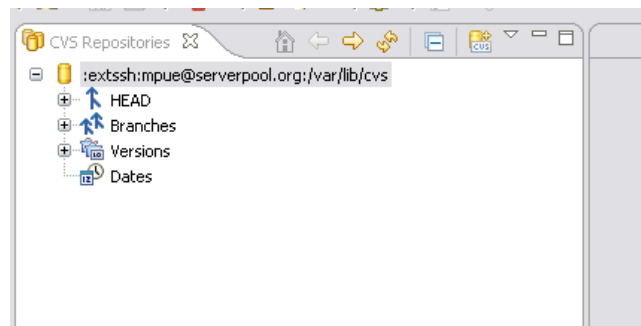


Figure 2.6: CVS branches

Expand the “HEAD” entry (click on plus) and you will see several projects. Find the entry “wkcms”, right click on it and choose “Check out as” and a dialog as in figure 2.7 will appear. Leave everything as it is and click on finish. Now go for a coffee, since the checkout might take some time.

After successful checkout, switch back to the “J2EE” perspective (Window->Open perspective->Other->Java J2EE) and your package explorer should now look like in figure 2.8

2.5.8 Final adjustments

The first thing we should now do is to **disable all validators**, sounds strange but we need to do it. In fact they don’t work at all, and slow the whole thing down. In my opinion the whole eclipse validation thing is crappy, so we turn this big mess OFF!

Go to “Window->Preferences” and type “Validation” into the text field on the top. Choose the first validation entry inside the list and you’ll see the screen as in figure 2.9

Now click “Disable all” and “Apply”, Eclipse now asks for a full rebuild, just click “OK” and you’re fine.

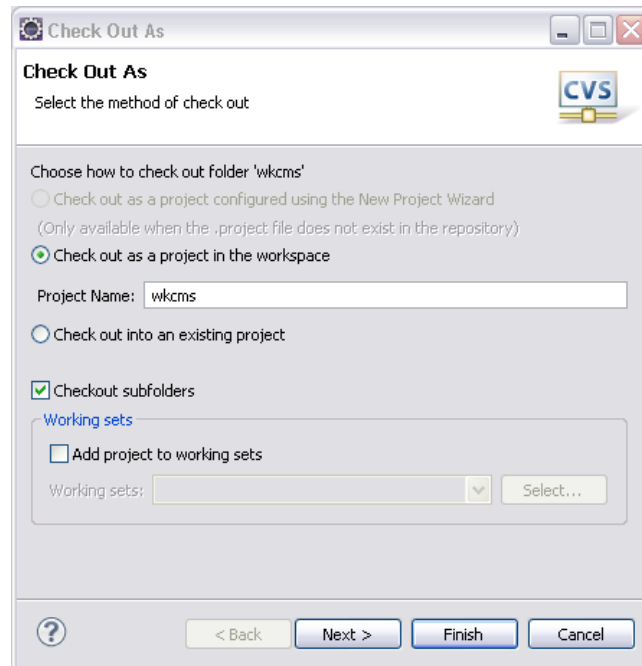


Figure 2.7: Checkout

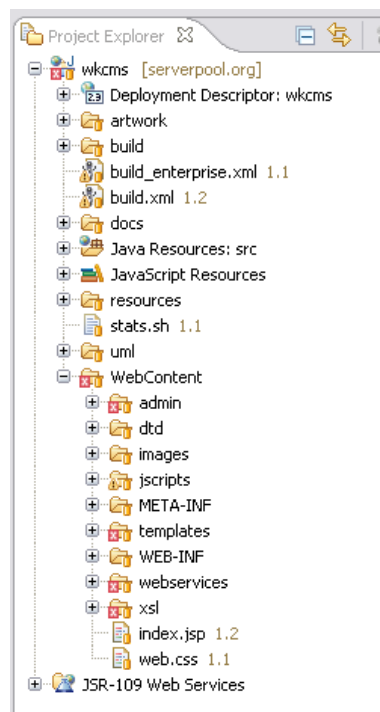


Figure 2.8: Project

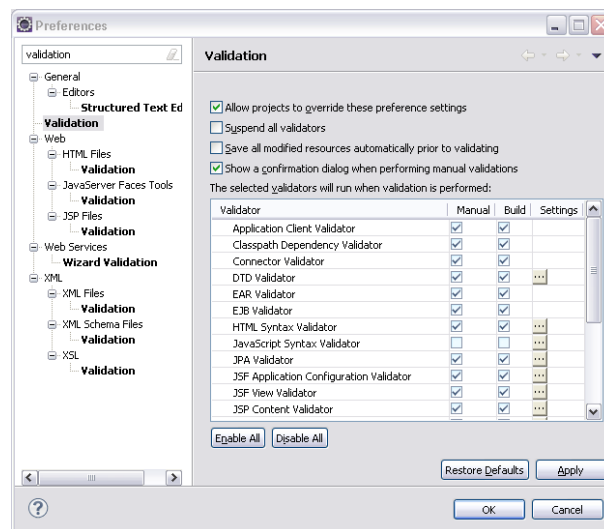


Figure 2.9: Validation

The final steps are now to adjust the build path and build script of the project, take a deep breath and we will proceed with the last few issues.

Right click on the root of the project and click “Properties”, choose “Java Build Path” on the left of the dialog and you should see something like in figure 2.10.

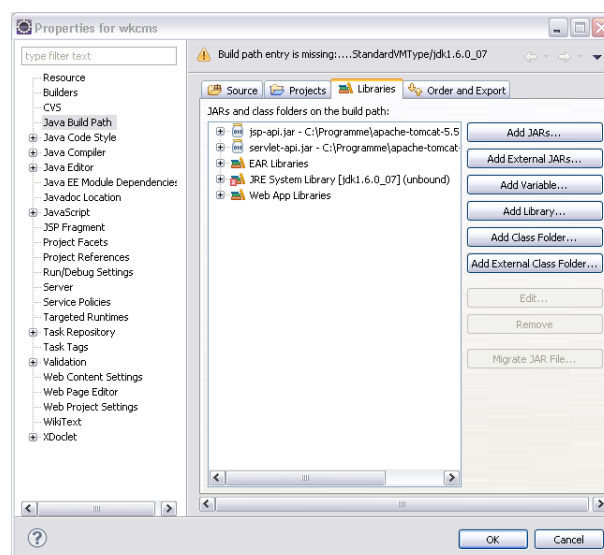


Figure 2.10: Build Path

Click on the tab “Libraries” and you’ll see the entry “JRE System library” marked with a red cross. Click on “Edit” end choose the workspace default JRE. Now we need to change the first two entries on which eclipse complains that they are missing. These are

- jsp-api.jar
- servlet-api.jar

These two libraries are located inside your tomcat\common\lib folder. Remove the two entries and click “Add external jars”. Navigate to your tomcat\common\lib folder and add those two jar files. Click on “Apply” and Eclipse will now start a build and the red crosses will disappear.

The last step is now to adjust the build.xml file. Open the file (it’s in the root of the project) and locate the following lines

```
<property name="catalina.home" value="/home/mpue/devel/apache-tomcat-5.5.17"/>
<property name="devel.home" value="/home/mpue/devel/j2ee_workspace"/>
```

Listing 2.4: Change the build options

Let the two entries point to your tomcat folder and your eclipse workspace folder and you’re done.

To check if everything is properly set up choose now “Window->Show View->Other->Ant->Ant” an new empty Ant view will appear. Just drag the build.xml into that view and the ant targets will appear inside that view. (see figure 2.11)

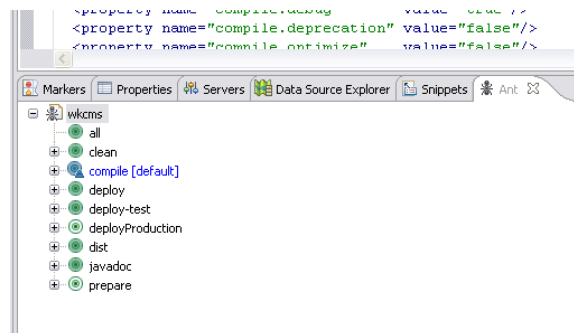


Figure 2.11: Ant

Double click on the “Deploy” target and Eclipse will start to compile the project. After successful compilation the binaries will be copied to the tomcat webapps folder and the console window will show a “BUILD SUCCESSFUL” message.

Congratulations, you have now a complete Weberknecht development environment!

You can now proceed with the steps as described in Section 2.1

Chapter 3

First Steps

As you might guess, this chapter deals with the first steps to take when using the weberknecht system. First of all, I will describe, how to use the web UI, how to create internet pages and how it fits all together.

Again open the following url in your browser:

`http://yourhostname.com:8080/wkcms/admin`

Where **yourhostname** is the server where you installed Weberknecht and we are assuming that you left all settings untouched during the installation process. The case that you've installed the system on a virtual domain in a different folder is a much more complicated case with which we will deal in a different chapter.

Now since you've already opened the page and the screen from figure 2.3 is being displayed, just click onto the first icon named "Manage contents" and the content editor will open (fig.3.1).

On the left side you will see an empty tree showing the term "Nodes". In WeberknechtCMS everything is a node, no matter what type of content lies behind the node. At the moment the type of the node can be:

- A content object (a single HTML page)
- a plugin (containing e.g. a gallery or a contact form)

Other types such as links, or external resources are linked via the plugin mechanism. In fact the only real type is a content, all other things are handled via the plugin mechanism. Refer the chapter 3.1 for more information.

In the next section I will show you how to create a content and put some images inside.

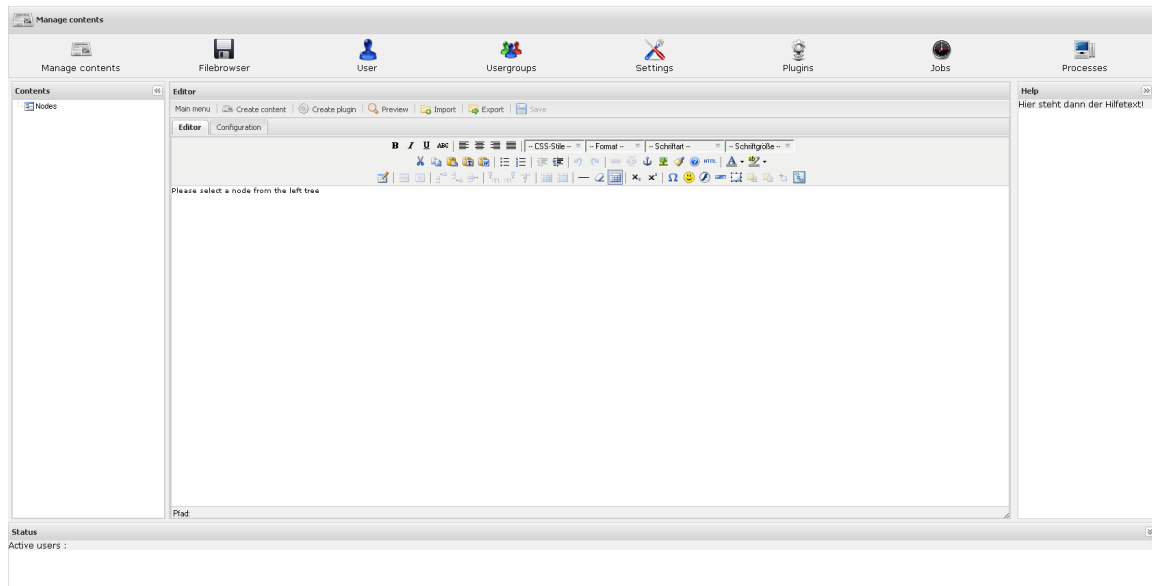


Figure 3.1: Content manager

3.1 Creating contents

First of all locate the menubar on top of the content editor window. It contains all actions which are available for the editor view.(fig. 3.2)

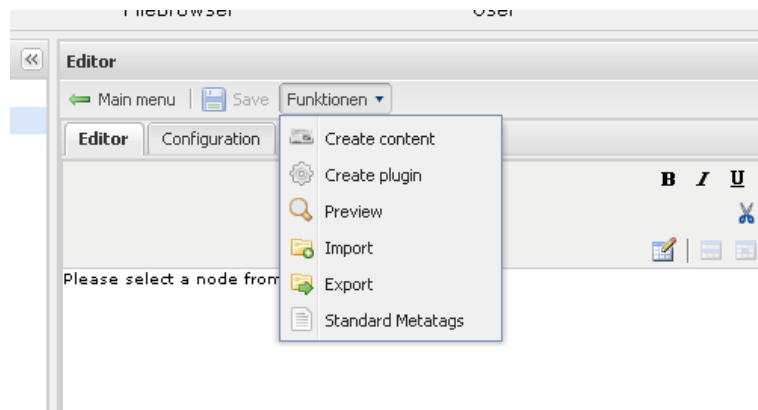


Figure 3.2: Content menubar

I will now shortly explain the meaning of the buttons and the menu entries from left to right.

Main menu

This will take you always to the main dashboard view from the beginning.

Save

Saves the currently edited content to the database.

The following items are visible, after clicking on the functions button:

Create content

Creates a new HTML content inside the main tree on the left side.

Create plugin

Creates a new plugin inside the content tree for special content purposes.(see chapter 3.1)

Preview

This will show a preview of the current page you are editing.

Import

This will import contents you backed up before. See chapter 6 for further details.

Export

This action will export all nodes of the content tree to an archive, which you can download to your computer.

Standard Metatags

This allows you to insert the standard set of page metatags. These are editable, when the **Configuration tab** of the editor window is selected.

Now, we will create the first content. In order to do this press the button “Create content” inside the **Functions menu** and the content dialog will appear. (fig. 3.3)

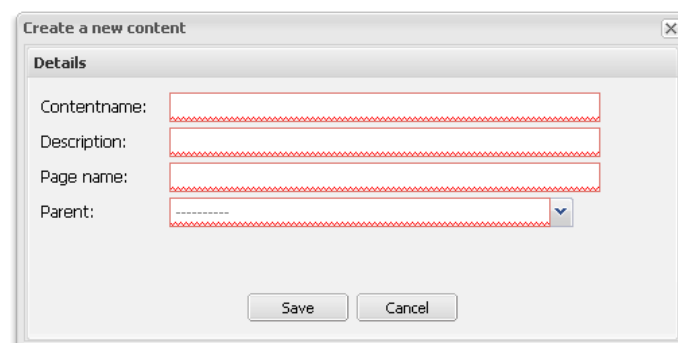


Figure 3.3: Create content

We need to fill all red marked fields (in fact all) to create a content.

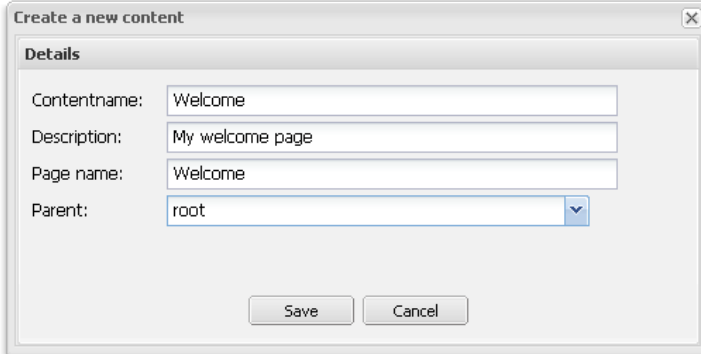
The **contentname** is the name of the content inside the tree. It will also be displayed inside the menu shown on your website. You can change it at any time. Let's name our first page "Welcome".

The **description** field should contain a short description of your page. This information will be indexed by search engines visiting your page. The more detailed a description is, the more it will be useful for search engines. So be careful, with what you write.

The **page name** will be displayed inside the titlebar of the browser and is also useful for setting the title of the bookmark of any page the visitor bookmarks.

The **Parent** field determines on which parent the new page will depend. If you would like to be the new page a root entry on the website, choose "root" in all other cases you may choose any existing node. We will discuss this issue in detail in chapter 5.4.

Now after you have entered the data like in figure 3.4, click the button **Save** of the dialog and the newly created content will open up inside the editor. As you may have noticed, the new content is also visible on the left tree.



The dialog box is titled "Create a new content". It has a "Details" section with the following fields:

- Contentname: Welcome
- Description: My welcome page
- Page name: Welcome
- Parent: root (with a dropdown arrow)

At the bottom are two buttons: "Save" and "Cancel".

Figure 3.4: New content

Your screen should now look like in figure 3.5 Now you may type in some useful welcome text for your visitors, which we will not discuss here.

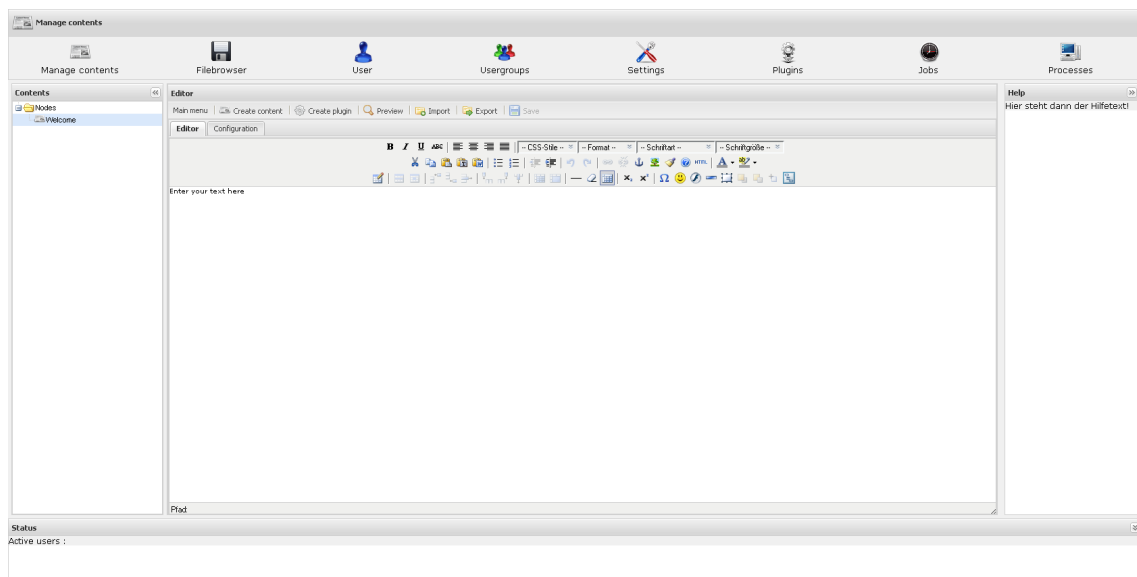


Figure 3.5: Empty content editor

As everybody likes images, we will now insert an image to make the page a little bit more colorful. Press the little button containing the tree



Figure 3.6: Insert an image

(fig 3.6) and the image dialog will pop up.

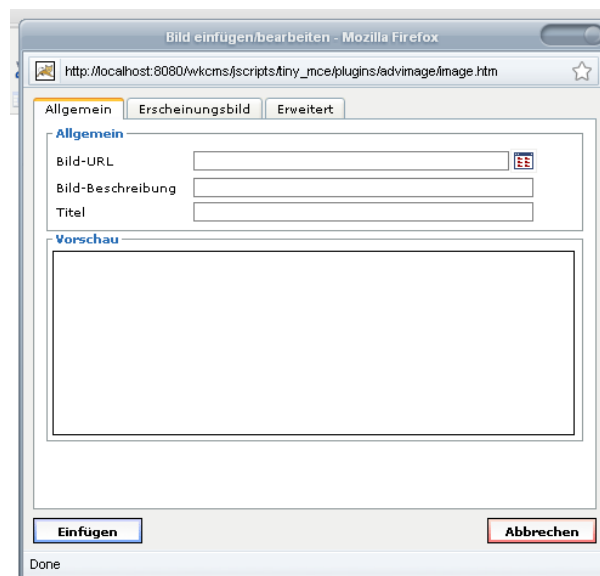


Figure 3.7: Image dialog

Press the small button right of the text field “Image URL” and the file manager will open. Click the “Browse” button on the bottom and choose an image from your harddisk. When your done, press “Upload” and the image will be uploaded to the server. When finished it appears inside the file list on the left side. Activate the checkbox left of the filename and click the button “Choose” below the preview image. (figure 3.8)

You should give the image a description and a title. That makes it a lot easier to be found by search engines. As a benefit, the browser shows a tooltip, while hovering over the image. When you’re fine press “Insert” and the image will be inserted into the content.

Now press “Save” and “Preview” and you will see, how your page will look like. According to the selected template, the page will be rendered very differently. The template mechanism will be discussed in detail in chapter 4.2.

If your page looks somewhat like in figure 3.10 you have successfully created your first page:

Congratulations!

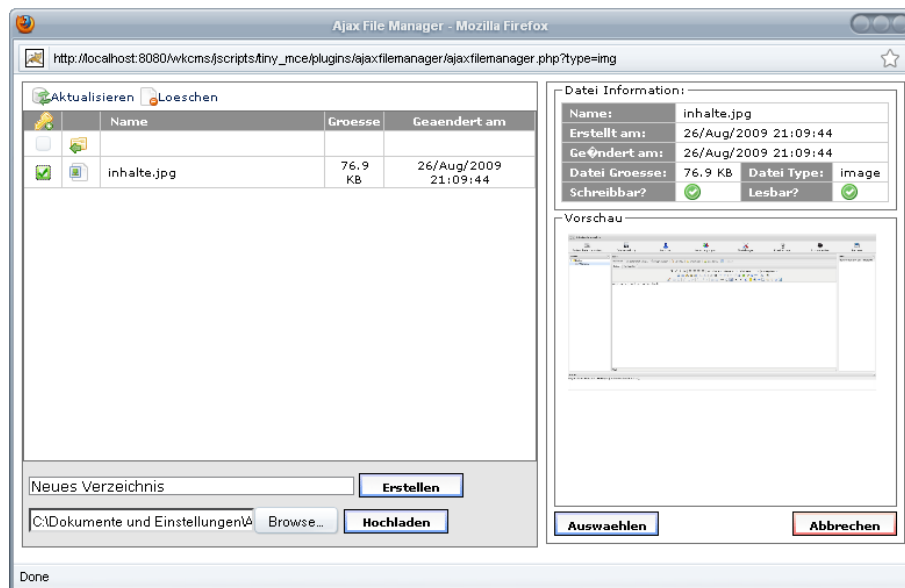


Figure 3.8: File manager

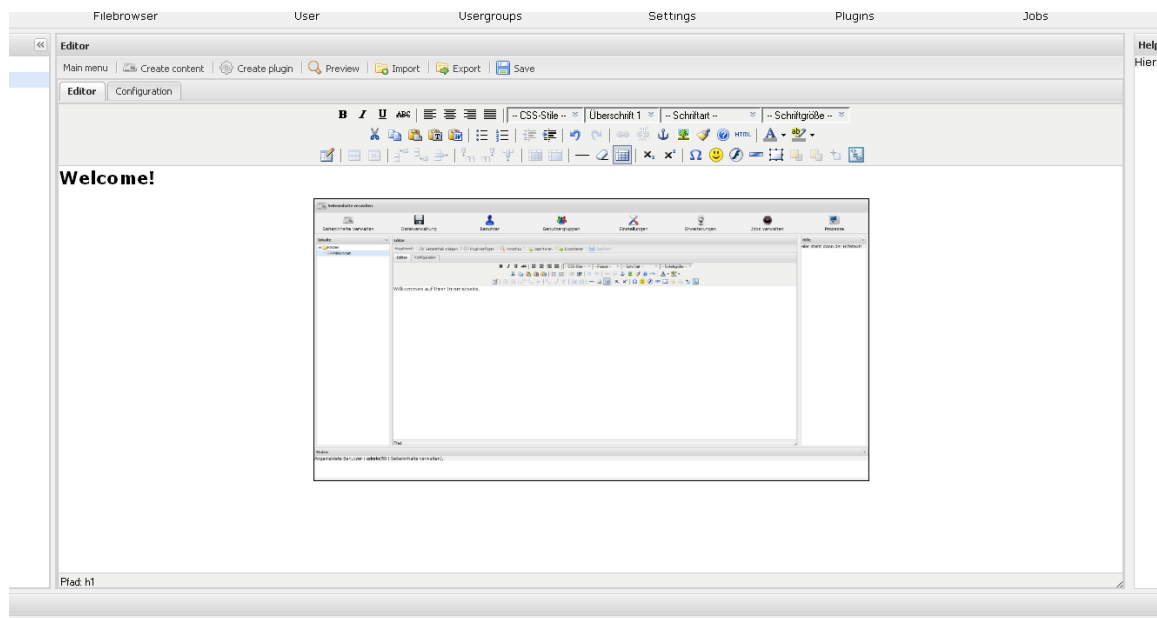


Figure 3.9: After inserting image

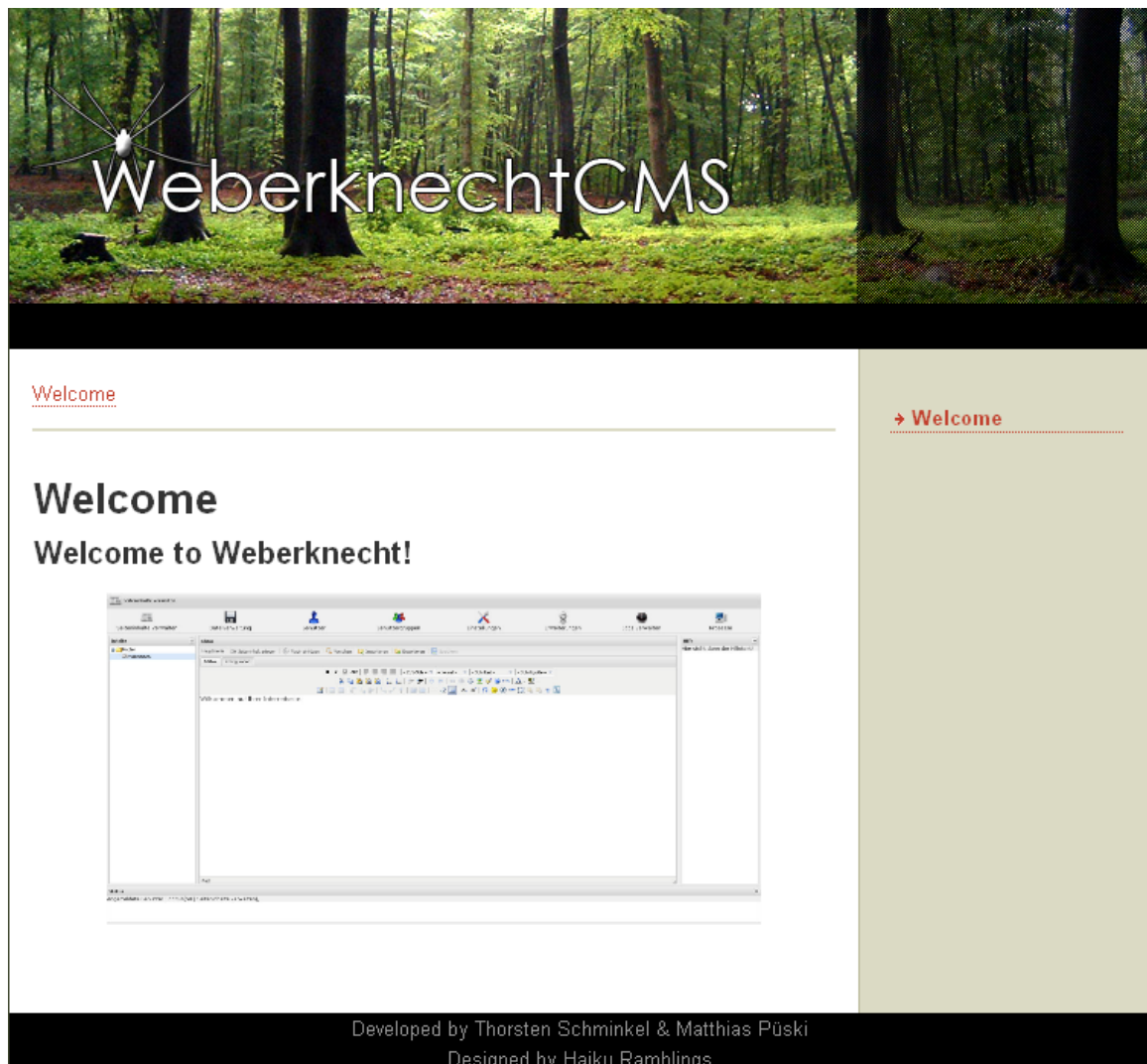


Figure 3.10: Preview

Chapter 4

Using plugins

4.1 Introduction

WeberknechtCMS provides a flexible plugin interface, which allows the user as well as the developer to install and to create new plugins in an easy way, without hassling with the system internals.

Plugins are used to provide additional modules for page rendering. A plugin may display an image gallery, provide a user registration form or give the user the ability to vote for something. For better understanding, what a plugin can do for you I'll introduce the file list plugin in the next section.

4.2 Using your first plugin

Consider the case that you want to provide a file list of a specific directory inside the content management directory. For this purpose you could have created a directory called **downloads** and put some files in it. So, how to provide these files to your visitors?

Let's create a new node with a plugin in it. As you've learned in the last chapter, all functions reside inside the functions menu. Click on the menu and choose **Create plugin**. The plugin dialog will appear as in figure 4.1

First of all, you need to choose the right plugin for this task. Open the selection box **Plugin** and a list with all available plugins will show up. (fig. 4.2) As you've might have guessed, it's the **FileListPlugin** which fits for our needs.

When you select the plugin the dialog will show the parameters which can (or must in most cases) be passed to the plugin. For the filelist plugin there are three parameters.

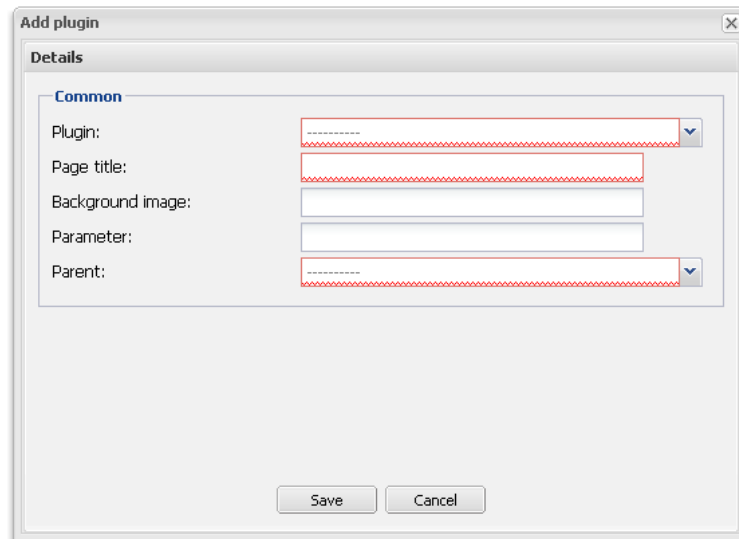


Figure 4.1: Create Plugin

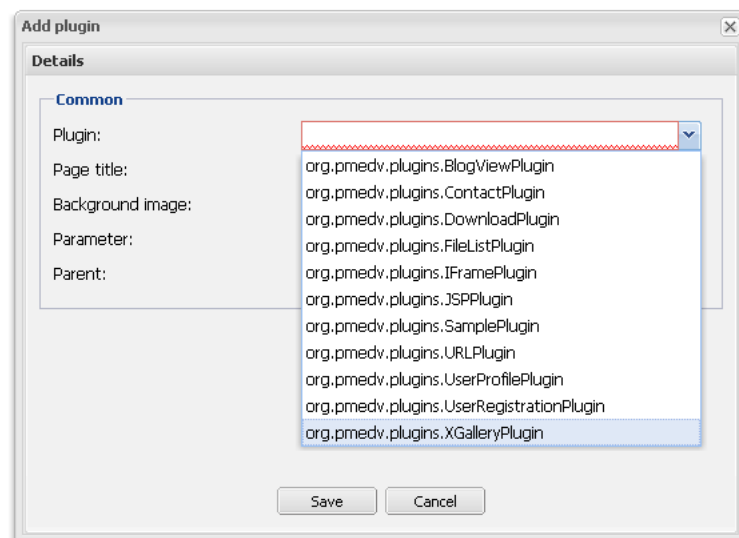


Figure 4.2: List of Plugins

Description

This text will be displayed on top of the filelist. Put a little description for your downloads in it or just leave it empty, if you don't want any text to be displayed.

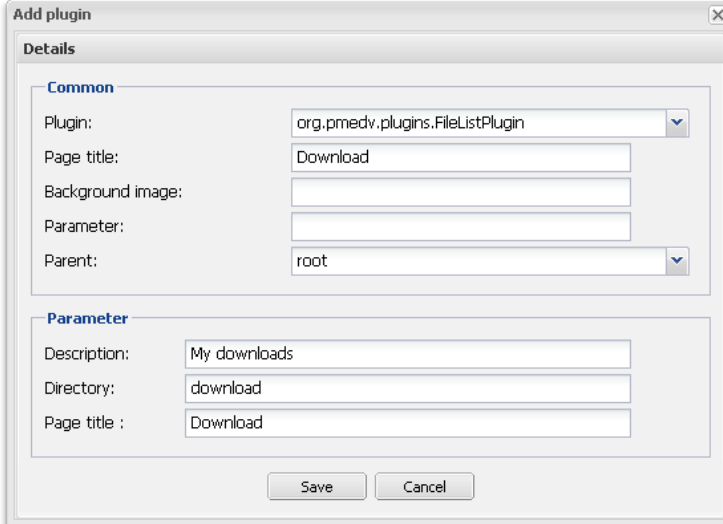
Directory

This is the most important parameter, it determines which directory will be chosen for display. Assuming you have a directory downloads just put the name **downloads** in there. In fact the field **Directory** denotes the path of the directory relative to the cms root.

Pagetitle

This is the title of the page which is displayed inside the browsers titlebar. If you leave it blank the default page name will be displayed.

After filling out the form it should look like figure 4.3.



The screenshot shows a window titled "Add plugin" with a close button in the top right corner. Inside the window is a "Details" section. Under the "Common" sub-section, there are five fields: "Plugin:" with a dropdown menu showing "org.pmedv.plugins.FileListPlugin", "Page title:" with a text box containing "Download", "Background image:" with an empty text box, "Parameter:" with an empty text box, and "Parent:" with a dropdown menu showing "root". Under the "Parameter" sub-section, there are three fields: "Description:" with a text box containing "My downloads", "Directory:" with a text box containing "download", and "Page title :" with a text box containing "Download". At the bottom of the window are two buttons: "Save" and "Cancel".

Figure 4.3: The filelist plugin

After hitting the save button, the plugin should appear inside the tree on the left side. Now open the browser and point it to your page. Assuming your download directory contains some files it should look somewhat like in figure 4.4

This is just an example for plugin usage, the complete list of plugins and a description of the parameters can be found in the appendix.

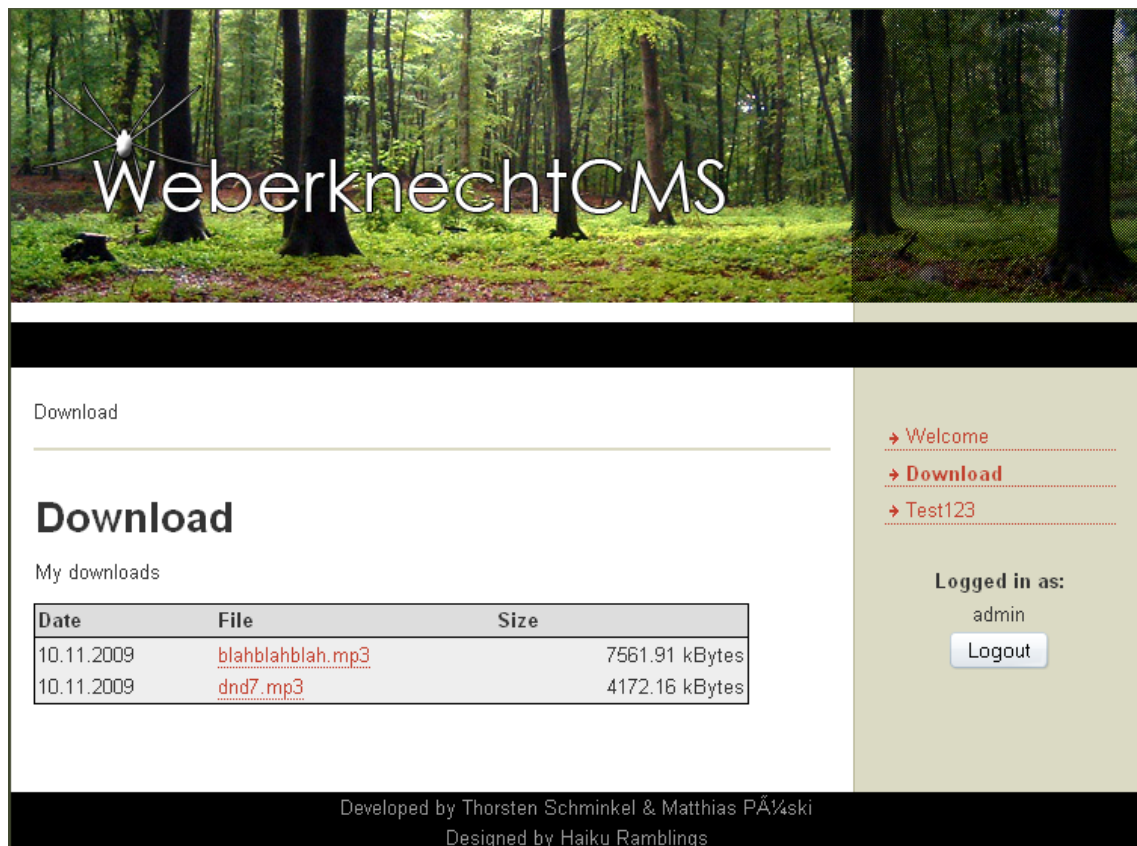


Figure 4.4: A filelist example

Chapter 5

Using templates

5.1 Introduction

So, what is a template all about. What does it and what is it needed for? These questions will be answered in the next few sections. As you begin using WeberknechtCMS you should now about how the content inside the system is organized and how it gets displayed inside your browser.

WeberknechtCMS stores all contents (the so called nodes) inside a tree structure which is persisted to the database. As you have already noticed, the content is organized in a tree structure. If a visitor opens your page and clicks on a node inside the menu, the content is fetched from the database, the selected template gets filled with the content and the page is finally displayed.

Figure 5.1 should give you an idea of how a page is rendered.

To refine the idea of the page we'll build a simple page example from scratch. All template files are located inside the template folder. To create a new one open the file manager and navigate to the templates folder. There should already be a few templates, at least a folder named **skeleton** should exist.

Imagine a simple internet page, where the menu is located on the left side, the logo on the top and the main content on the right side of the screen. This is a very typical pattern for internet pages and might look like in figure 5.2.

The system provides several tags for page construction. A full list of page tags with the according explanation can be found in the appendix. For now let's concentrate on the basic page elements used in the example picture.

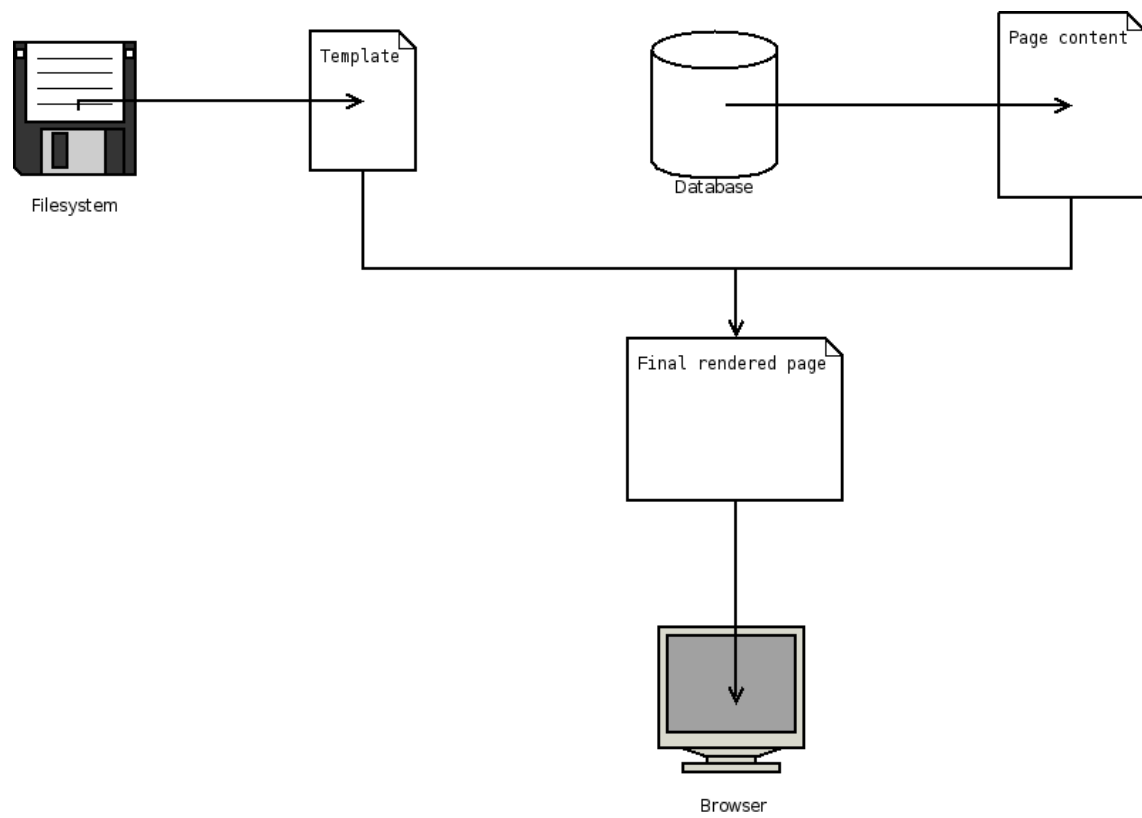


Figure 5.1: The template mechanism

Pagetitle		Suchbegriff <input type="text"/> <input type="button" value="Search"/>
Welcome News Imprint	Imprint <hr/> 1. Content <p>The author reserves the right not to be responsible for the topicality, correctness, completeness or quality of the information provided. Liability claims regarding damage caused by the use of any information provided, including any kind of information which is incomplete or incorrect, will therefore be rejected. All offers are not-binding and without obligation. Parts of the pages or the complete publication including all offers and information might be extended, changed or partly or completely deleted by the author without separate announcement.</p> 2. Referrals and links <p>The author is not responsible for any contents linked or referred to from his pages - unless he has full knowledge of illegal contents and would be able to prevent the visitors of his site from viewing those pages. If any damage occurs by the use of information presented there, only the author of the respective pages might be liable, not the one who has linked to these pages. Furthermore the author is not liable for any postings or messages published by users of discussion boards, guestbooks or mailinglists provided on his page.</p> 3. Copyright <p>The author intended not to use any copyrighted material for the publication or, if not possible, to indicate the copyright of the respective object. The copyright for any material created by the author is reserved. Any duplication or use of objects such as images, diagrams, sounds or texts in other electronic or printed</p>	Logged in as: admin <input type="button" value="Logout"/>

Figure 5.2: A template example

5.2 Creating your first template

The first thing you'll obviously need, is an empty template, for this make a copy of the skeleton folder, which contains everything needed for a basic page. Give the copy a name of your choice. The copied folder will be the new template.

Open the file main JSP with your favorite text editor. Don't be scared about all the things you find there, the first step is to delete everything in it, except the first line since we'll build a template from scratch. The first line is needed to tell the browser which type of document we have. So just leave it as it is for now.

Every template needs the basic header to be included. The header provides basic page includes as well as the tags we need to build up the template. The header is included as follows:

```
<%@include file="header.jsp" %>
```

Listing 5.1: Include the header

5.3 Defining the page header

Next we'll create the header which is also needed for every template:

```
<html:html>
  <head>
    <title>
      <page:pagetitle/>
    </title>
    <bean:write name="MainpageForm" property="content.metatags" filter="false"/>
  </head>
```

Listing 5.2: Simple HTML header

Wait, that scares you and does not look like ordinary HTML at all? You're right and this is the explanation:

There are two types of tags available to build a template.

- General page tags
- Tags that contain the content and its metainformation

General page tags are tags that produce for example the page title, they contain configuration information from the server, such as the hostname the site is running on and so on.

The tag **<page:pagetitle/>** for example just renders the pagetitle of the current selected node. The second tag

<bean:write name="MainpageForm" property="content.metatags" filter="false"/>

renders the metatags you've defined for the content.

5.4 Adding the body part

The next essential part is the body part of the page. It simply start with a standard HTML body tag Unfortunately this manual can not give you an introduction to HTML because this is far beyond the scope of this book. For further reading I suggest you to read of the many online HTML books and tutorials.

The body part of your template might now look like the following:

```

<body>
  <table border="1" align="center">
    <tr>
      <td colspan="2" class="header">
        <h1>
          <page:config property="sitename"/>
        </h1>
      </td>
    </tr>
    <tr>
      <td valign="top" width="15%">
        <page:xmenu mode="website"/>
      </td>
      <td valign="top" width="70%">
        <p>
          <h2><page:pathway separator="/" /></h2>
        </p>
        <hr/>
        <p class="content">
          <bean:write
            name="MainpageForm"
            property="content.contentstring"
            filter="false"/>
        </p>
      </td>
    </tr>
  </table>
</body>

```

Listing 5.3: footer of the page

Now for the hardest part - there is a lot of information inside this small piece of code. As you remember, there are two kinds of tags, the first one here

<page:config property="sitename"/>

displays the name of the site defined in the configuration module. The next one

<page:xmenu mode="website"/> renders a basic menu into the page. The only required attribute here is the attribute **mode** which can be either **website** or **blog**. For a simple page menu we choose the mode **website**.

The next and most important tag renders the main html content into the page:

<bean:write name="MainpageForm" property="content.contentstring" filter="false"/>

The **MainpageForm** object contains all available page objects, such as the currently selected

node and the according content as well. If you want to address some page object and one of its properties, you'll have to write

<bean:write name="MainpageForm" property="object.propertyname"/>

The full list of available objects and properties can be found in the appendix of this manual.

After the body part has been defined, the document has to be closed with:

```
</html:html>
```

Listing 5.4: body of the page

With this step you have finished building your first template. You may now upload it to the server. After uploading the template is available for selection in the configuration module.

Chapter 6

Managing page menus

Chapter 7

Import and export of data

List of Figures

1.1	Weberknecht architecture	4
2.1	Weberknecht installation	7
2.2	Weberknecht login	8
2.3	Weberknecht mainscreen	8
2.4	CVS Perspective	12
2.5	Create a CVS Repository	12
2.6	CVS branches	13
2.7	Checkout	14
2.8	Project	14
2.9	Validation	15
2.10	Build Path	15
2.11	Ant	16
3.1	Content manager	18
3.2	Content menubar	18
3.3	Create content	19
3.4	New content	21
3.5	Empty content editor	21
3.6	Insert an image	22

3.7	Image dialog	22
3.8	File manager	23
3.9	After inserting image	23
3.10	Preview	24
4.1	Create Plugin	26
4.2	List of Plugins	26
4.3	The filelist plugin	27
4.4	A filelist example	28
5.1	The template mechanism	30
5.2	A template example	31

Bibliography