

Charles University in Prague  
Faculty of Mathematics and Physics

# **BACHELOR THESIS**



Michal Halaša

## **Extensible Collaborative Development Platform**

Katedra distribuovaných a spolehlivých systémů

RNDr. Petr Hnětynka Ph.D.

Study programme: Informatics

Prague 2011

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague 5.8.2011

Název práce: Rozšiřitelná vývojová platforma s možností spolupráce

Autor: Michal Halaša

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Petr Hnětynka Ph.D., S 212, Malostranské nám. 25, Praha

Abstrakt: V dnešní době existuje více vývojových platforem s důrazem na spolupráci uživatelů. Tyto platformy poskytují vývojářům sadu aplikací umožňujících společný vývoj (VCS nástroje, aplikace na sledování chyb, emailové konference atp.). Na druhou stranu existují samostatné nástroje na obsluhu jednotlivých aplikací. Většinou tyto samostatné nástroje poskytují více funkcionality pro vývojáře, ale musejí být nastaveny a spravovány samostatně.

Cílem této práce je vyvinout vývojovou platformu (tzv. Forge), která je plně rozšiřitelná a poskytuje sjednocený způsob správy všech použitých nástrojů.

Klíčová slova: java, rozšiřitelnost, vývojová platforma

Title: Extensible Collaborative Development Platform

Author: Michal Halaša

Department / Institute: Katedra distribuovaných a spolehlivých systémů

Supervisor of the bachelor thesis: RNDr. Petr Hnětynka Ph.D., S 212, Malostranské nám. 25, Praha

Abstract: Currently, there exist a number of collaborative development platforms (also known as "forges"). These platforms offer to developers a set of tools for collaborative development (e.g. VCS tools, bug-tracking tools, mailing-list managers, etc). On the other hand, there exist many standalone tools for bug-tracking, etc. Usually, these standalone tools offer more features to the developers, however they have to be set up and managed separately.

The goal of this thesis is to develop a collaborative development platform (forge) that is fully extensible, i.e. it will allow developers to easily add existing tools to it and manage them in a unified way.

Keywords: java, forge, extensible, development platform

# Table of Contents

1 Introduction.....	1
2 Codeheap Architecture.....	3
2.1 Presentation Layer.....	4
2.2 Application layer.....	7
2.3 Database layer.....	8
2.4 CodeheapConnector.....	9
3 Built-in Subsystems.....	11
3.1 MantisBT.....	11
3.2 DokuWiki.....	12
3.3 Subversion.....	14
4 Codeheap User View.....	16
4.1 Implementation.....	19
5 Installation.....	22
6 Related work.....	23
6.1 Advantages of Codeheap.....	23
7 Conclusion.....	25
7.1 Difficulties.....	26
7.2 Future improvements.....	26
9 List of Abbreviations.....	28
10 List of Images.....	29
11 Attachment.....	30

# 1 Introduction

Software projects are no longer developed in one place, one office or one classroom. Most of the applications nowadays are developed in decentralized and distributed way. Sometimes the project participants do not even know themselves, they have never met in person, they know each other only by their names or nick names which are shown in documentation, commit descriptions, in code repositories or in bug reports. This leads to a necessity of using several tools that help the team members to communicate, coordinate their work and to manage their responsibilities.

Over the time these tools have evolved to more or less standardized software applications. The basic set of auxiliary applications for every project consists of code repository, wiki platform for software introduction, scheduling and public exposure and the bug reporting tool to help the outside users of the project to report the feature requests or bugs.

Software projects managed by Faculty of Mathematics and Physics are the great example of this kind of environment. Several types of projects (seminar work, bachelor work, year project etc.) are managed by the faculty members. To set up the project, a lot of manual actions are necessary – create the users in the system, create the project in the bug tracker system, create the repository in the Software Configuration Management software, configure the access rights etc.

Managing multiple software projects leads to a necessity of their centralized management to ease the creation and modification of projects and also management of the users and administrators. Another goal is to decrease the complexity and duration of administrative tasks for the administrators of the underlying systems.

Next aim is to create a complex view of the managed projects with the easy to use list with the detailed information about the developers, web pages and other attributes of each project.

Currently on Faculty of Mathematics and Physics all the project, users, repositories, etc. are manually created and modified. That is why a new solution is necessary integrating all the tools that are currently in use – Mantis bug tracking tool, SVN software configuration management and DokuWiki platform for project web pages.

Projects that are designed as development platforms solving these kind of issues (so called Forges) are already existing. Evaluation of several of these platforms has been done but either they are too complex for the faculty needs (SourceForge) or they do not support the already used tools so the migration of the currently existing projects is not feasible. Another condition is that the development

platform must be Open Source to allow easier modification and extension and no license is necessary.

Purpose of this work is to create an application that will simplify project and user creation process in all subsystems necessary for the implementation and management of the project. These are:

- Bug Tracking System (BTS), which is responsible for issue tracking
- Software Configuration Management (SCM) taking care of storing and versioning of all project files in one repository
- Wiki platform for project web pages and project organization - meeting planning and work schedule.

## 2 Codeheap Architecture

Primary goals, when evaluating the architecture for the new development platform, are easy to use centralized management and extensibility. This must be achieved with possibility to develop plugins for the tools in use.

During the architecture evaluation process for Codeheap application, several options were considered. All of them are based on Java language but otherwise different. First choice was a thick desktop client based on Java SE with GUI based on Swing framework. This option was not chosen because of it's complexity to develop (client – server part communication, online/offline operation). Every user would have to install the desktop application to be able to connect to Codeheap. This is against the idea of distributed, nomadic team members that sometimes use several computers to develop the same application.

The other option was a web based application with all user interface implemented as web pages. This fits perfectly on the requirements mentioned earlier. The next question was, what to use for business logic. Several options were again considered – plain Java Servlets, non-standard frameworks like Struts or Spring or the Java EE standardized Enterprise JavaBeans (EJB). Enterprise JavaBeans[7] framework was chosen because it is a Java EE standard, very well integrated with other Java EE standards like JPA and JNDI. From the decision to rely on standards came also the selection of the three tier architecture which clearly separates the presentation, logic of the application and storage of the data.

Three tier architecture introduces three layers – presentation layer, application layer and database layer. Presentation layer is responsible for data presentation to the end user and user interaction. Application layer contains all the business logic of the application and database layer is used for data storage and data model design.

For presentation layer we chose the ZKoss framework because of its great integration with Java, its maturity and the ease of use for smaller and mid sized projects.

Application layer is implemented in Enterprise JavaBeans (EJB) standardized framework in Java EE. Enterprise JavaBeans is a modular component architecture used in most enterprise Java applications that are running inside an Application Server (AS). As the application server we use the GlassFish AS developed by Oracle. It is reference implementation of the enterprise Java standard. Last layer in the design is the database layer based on MySQL relational database because of its simple administration, broad and high quality support.

All of these applications are Open Source Software.

Complementary layer is the CodeheapConnector - Application Programming Interface (API) designed for subsystem connection and management, based on EJB and packaged with the application. Every subsystem in use (bug tracker, wiki, software configuration management tool) is implemented with CodeheapConnector. That is how pluggable architecture is achieved. For some subsystems the plugin is necessary in their implementation language to enhance the provided API for Codeheap intergration.

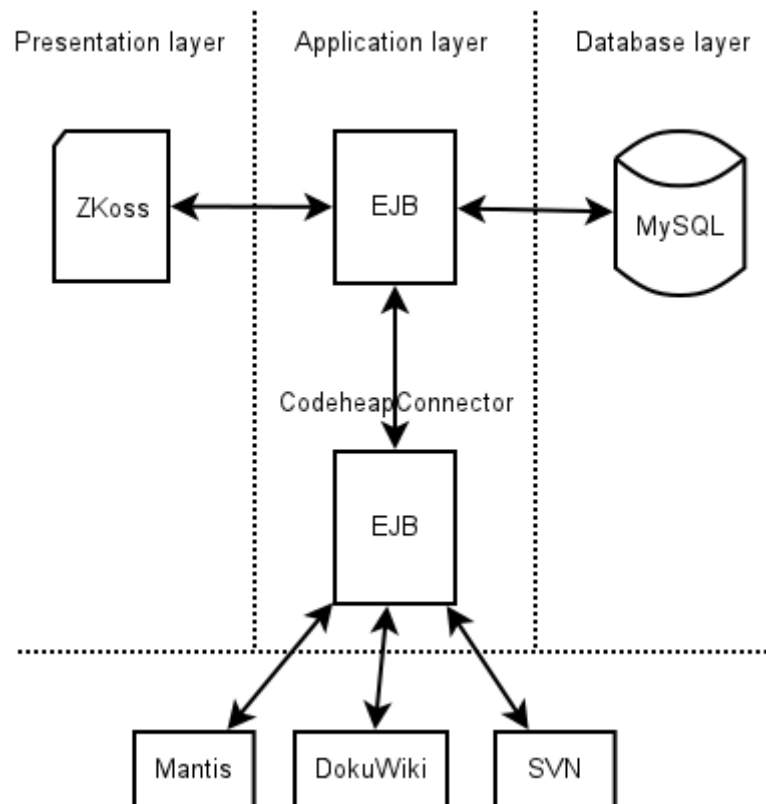


Image 1: Codeheap Architecture

## 2.1 Presentation Layer

ZK [1] is an open-source Asynchronous JavaScript and XML (Ajax) web application framework written in Java, that enables creation of rich graphical user interfaces for Web applications with no JavaScript and little programming knowledge.



The core of ZK consists of an Ajax-based event-driven mechanism, over 123 XUL and 83 XHTML-based components, and a markup language for designing user interfaces. Programmers design their application pages in feature-rich XUL/XHTML components, and manipulate them upon events triggered by end user's activity. It is similar to the programming model found in desktop GUI-based applications.

ZK is server side framework with very low memory and CPU footprint on client machine. All computation is done on server and only the results are send back to the client browser. This is very useful when the network throughput is sufficient because of the instant feedback on the client side of the user actions. This adds flexibility and removes some security issues because cross site scripting and SQL injection attacks are taken care of by the framework itself.

The ZK User Interface Markup Language (ZUML) is based on XML. The XML document models a tree of ZK components. XML element attributes contain/specify initial values for the component. XML processing instructions carry information related to the whole page, such as page character set and title.

Different sets of components are distinguished by XML namespaces. For example, the components from the XUL namespace and those from XHTML namespace can be used simultaneously.

In ZUML pages the Java code (interpreted with BeanShell [2]), XUL and other scripting languages can be mixed together but the preferred way is to extract the logic of the page to separate Java class called Composer and leave the ZUML page only for the layout.

## Composer

CommonComposer binds the page data and actions between ZUML definition and Java implementation. It is extension of the ZK implementation of this pattern in GenericForwardComposer class. Throughout the whole ZK framework, naming convention is preferred over the configuration, e.g. here in Composer class the binding between ZUML page and backing Composer class is done via specifically named methods, e.g. method with the specific signature

```
public void onClick$submitButton(Event event)
```

is automatically invoked when user clicks on the submitButton defined in ZUML page like this

```
<button id="submitButton" label="Click me!" />
```

Convention over Configuration [3] paradigm increases the development speed and reduces the necessary configuration overhaul. Readability to the code is also increased. Drawback is a flexibility loss, but this is acceptable for smaller and mid-sized projects. It can be avoided in ZKoss by not using the ZUML-Composer binding and using custom solution.

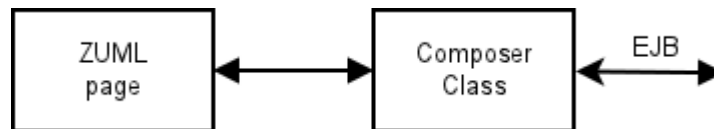


Image 2: ZK Architecture

## ZUML

ZK User Interface Markup Language (ZUML) is used to define the visual part of a page. Very similar to the HTML but much richer. Pages are stored with *zul* extension and ZK request handling servlets are registered in standard *web.xml* configuration file. ZK specific configuration (session parameters, stylesheet definition etc) is placed in *zk.xml* file.

ZUML page is rendered to HTML and JavaScript and sent to the client browser by ZK engine. Due to its server side architecture only thin JavaScript layer is sent to the client. This small JavaScript code catches the user interaction with the page (enter text, press key, move mouse) and sends the events to the server. On the server, the event is caught by internal ZK servlet, parsed and dispatched for execution or thrown away if no action is registered for that type of event.

ZK library implemented in *WEB-INF/tld/codeheap.tld* provides customised Resource Bundle and the security methods to check directly in ZUML page whether the current user has the role to access/invoke the operation. If not, the desired HTML item can be disabled. This is achieved by injecting the library code into a ZUML page in itsheader using the following declaration

```
<?taglib uri="/WEB-INF/tld/codeheap.tld" prefix="b"?>
```

and following use

```
<tab label="Users" visible="{b:h('admin')}" />
```

## 2.2 Application layer

Application layer is responsible for the whole application logic and data manipulation that is not necessary only for the presentation layer. For example, data loading from the database and merging is done in application layer but the data alteration so it's better shown or rendered in the client is the job for the presentation layer and so for the ZKoss framework.

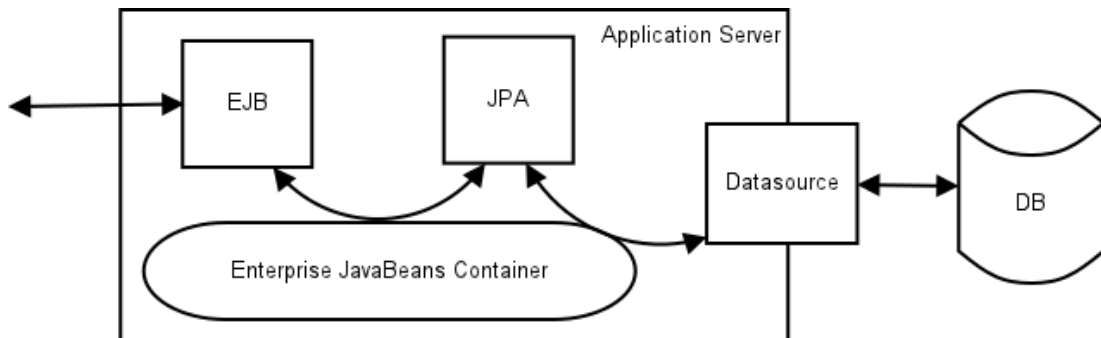


Image 3: Application Layer

Data is loaded by standard JEE Java Persistence API (JPA) implemented by the EclipseLink library, which is included in the GlassFish application server. The data is then provided to the presentation layer via public API of Enterprise JavaBeans objects. The EclipseLink also takes care of persisting of the data using entity to database mapping. This mapping is defined in two places. First one is *persistence.xml* configuration file defining the list of classes that are mapped on database tables and the datasource to be used for accessing the database. Second place where the mapping is defined is the JPA entities. There are specified all available queries, mapping between SQL and Java datatypes, way the binding data from other tables is loaded etc.

Application layer is logically divided into two modules *codeheap-api* and *codeheap-ejb*. In *codeheap-api* there are the public EJB interfaces and JPA Entities including their database mapping configuration. Interfaces are separated by concern, so there is the *IProject* interface providing all *Project* entity related data, *IUser* interface with methods to load users in the system, user types or login method. Other two interfaces *ISettings* and *IConfig* are used to manipulate configuration and management data.

In addition to data manipulation, application layer is also responsible for subsystem calls via CodeheapConnector which is described later on.

The modules of the application layer are deployed to the Application Server. They run inside of the EJB container, which manages their entire life cycle. The application server provides transaction handling, object pooling, database

connection pools, directory services so the application doesn't have to implement all these features itself.

## 2.3 Database layer

For database layer we use the MySQL database. There are two main tables in the design - USER and PROJECT, which are bound together via project-developer and project-assigner associations. Besides these two main tables, there are other tables, directly or indirectly associated to the main ones. For example for a user table there is the user type attribute (root, administrator, project leader, user) which defines set of operations that can be used by the user. User has Many-To-Many relation with the Project table defined by binding table project\_2\_user. Users in the root or administrator role can also be defined as sponsors of the project. Sponsor is the creator or initiator of the project who is responsible for the scope and setting up the project.

Projects have their basic attributes like name, start and end date, website and also associated objects like license, project type and keywords. To keep the design simple, no other database features (triggers, views, stored procedures) except tables and relations are used. Application server must have database connection pool configured to be able to connect to a database. This is done by GlassFish CodeheapPool Connection Pool and CodeheapPool JDBC Ressource. These are used in the application layer to map the JPA Entities to the live data in the DB.

Database model is on Image 4.

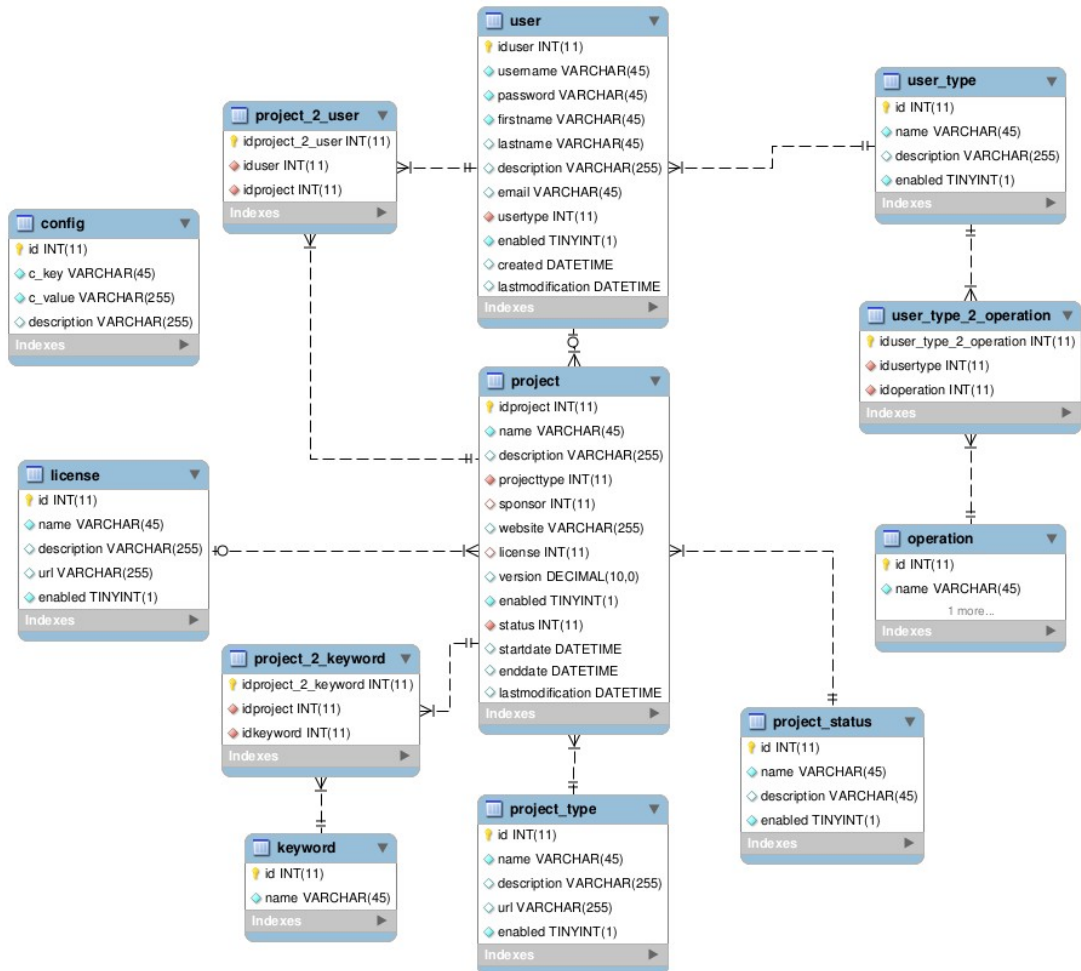


Image 4: Database model

## 2.4 CodeheapConnector

CodeheapConnector is used to access the subsystems that are installed in Codeheap. It is designed to be extensible and pluggable. CodeheapConnector API consists of common methods for all the subsystems (create project, create user, assign user to a project, etc.) and some specific methods only for a specific subsystem (for bug tracking software it is the number of project issues or list of pages for Wiki platform).

There are three types of available backend systems.

- Software Configuration Management (SCM) takes care of the storage, versioning and merging of the project files

- Bug Tracking System (BTS) is used to keep track of the issues of the project. These can be either design or functional bugs and feature requests
- Wiki is used as a first hand collaboration system for the project team. It's used as a project homepage, to schedule the work and meetings and for the project documentation

CodeheapConnector API is defined as a set of Interfaces, one for each subsystem. It is defined in the *codeheap-connector-api* module. There are four interfaces:

- IGenericConnector is an abstract interface with definitions of common methods for all other three interfaces which extend this one
- IBTSCConnector for bug tracker integration
- ISCMConnector for software configuration management integration
- IWikiConnector for wiki platform integration

Each of the three non generic interfaces is implemented as Enterprise JavaBean component responsible for subsystem communication. Implementations for the default subsystems (SVN, MantisBT and DokuWiki) are already provided in *codeheap-connector-ejb* module as SVNSCMConnectorBean, MantisBTSCConnectorBean and DokuWikiConnectorBean classes.

All the communication between the core application and the subsystems is by invoking methods on these interfaces so there is clear logic separation between the core Codeheap application and CodeheapConnector subsystems.

# 3 Built-in Subsystems

Default subsystems which are part of the Codeheap implementation, because they are currently used in Faculty of Mathematics and Physics, are MantisBT, SVN and DokuWiki.

## 3.1 MantisBT

Mantis Bug Tracker is a free open source web-based bug tracking system released under the terms of the GNU General Public License version 2. The most common use of MantisBT is to track software defects. However, MantisBT is often configured by users to serve as a more generic issue tracking system and project management tool.

### Codeheap customization

For communication with external systems MantisBT contains Mantisconnect API build on obsolete NuSOAP library providing Web Service capabilities for PHP. To be able to connect to and use the Mantisconnect API from Codeheap, outdated Web Service library Apache Axis v1 has to be used. This puts some constraints on the extensibility of the API and flexibility of its usage.

CodeheapConnector API for communicating with MantisBT is defined in the IBTSCconnector interface. Its implementation - MantisBTSCconnectorBean class is part of the *codeheap-connector-ejb* module. This bean uses the Mantisconnect client generated by Axis and placed in *biz.futureware.mantisconnect* package. This client is called from EJB and remotely invokes the Mantisconnect API with web service calls.

The client is generated by the WSDL2Java class from the Axis library.

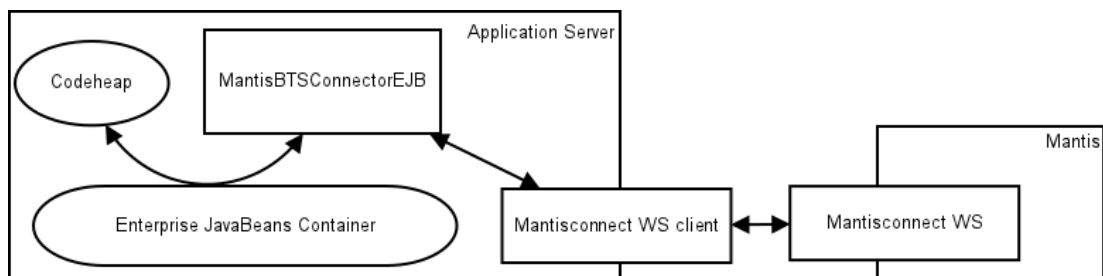


Image 5: MantisBT Connector

To be able to control MantisBT from the Codeheap, it was necessary to extend the Mantisconnect API by the following web service methods:

- mc\_user\_create
- mc\_user\_update
- mc\_user\_set\_password
- mc\_user\_delete
- mc\_user\_set\_default\_project
- mc\_project\_add\_user
- mc\_project\_remove\_user
- mc\_get\_user\_created\_issues
- mc\_get\_count\_user\_created\_issues
- mc\_get\_user\_solved\_issues
- mc\_get\_count\_user\_solved\_issues
- mc\_get\_project\_created\_issues
- mc\_get\_count\_project\_created\_issues
- mc\_get\_project\_solved\_issues
- mc\_get\_count\_project\_solved\_issues

Implementation of these methods is in *mc\_codeheap\_api.php* file that is part of the Codeheap plugin for MantisBT.

The following constants must be defined to enable MantisBT integration:

- WS\_TARGET - target URL for WS invocation
- TECHNICAL\_USER - user used for WS authentication
- TECHNICAL\_PASSWORD - password of the WS user

## 3.2 DokuWiki

DokuWiki is a wiki software aimed at small companies' documentation needs. DokuWiki is licensed under GPL 2 and written in the programming language PHP. It works on plain text files and thus needs no database. Its syntax is similar to the one used by MediaWiki and makes sure the data files remain readable outside the wiki.

DokuWiki is a standards compliant, simple to use Wiki, mainly aimed at creating documentation of any kind. It is targeted at developer teams, workgroups and small companies. It has a simple but powerful syntax which makes sure the datafiles remain readable outside the Wiki and eases the creation of structured texts. All data is stored in plain text files – no database is required.



## Codeheap customization

For each project the so called DokuWiki Namespace is created. User ACL rights can be defined separately for each Namespace which is very suitable for user access separation of different projects. Namespace is very similar to a folder paradigm in file system structure and pages could be considered as files. So for each project we create the folder with one simple file generated from template and representing the home page of the project. For each project/namespace we also create the group for which the access rights are set for this namespace. Only authenticated members of this group can access and modify the Namespace pages.

So in the DokuWiki we have to work with slightly different naming for the same entities. Project is equal to Namespace, project users are equal to the members of the group with the same name as the project.

DokuWiki provides API on XML-RPC basis which is the protocol for remote procedure invocation over the HTTP protocol with XML based data representation. For interoperability with Java we use the Apache XML-RPC framework.

Codeheap implementaion is in the DokuWikiConnectorBean in the *codeheap-connector-ejb* module.

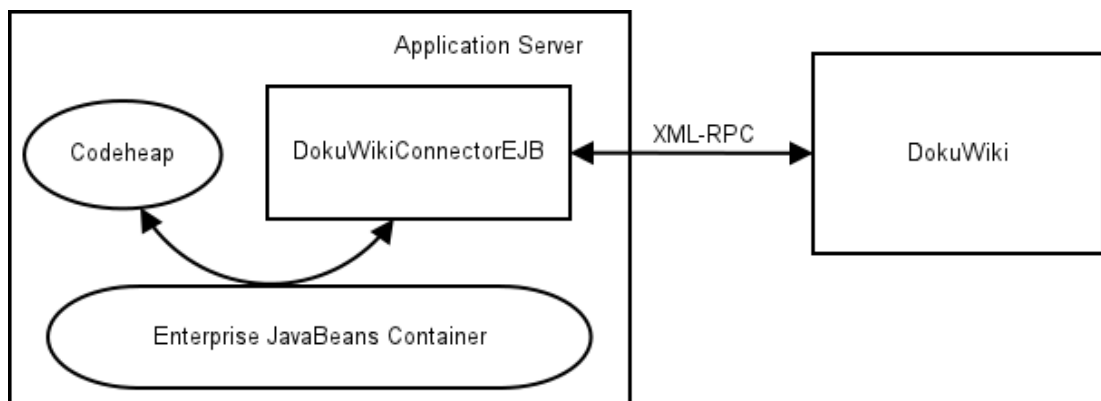


Image 6: DokuWiki Connector

Provided API must have been extended with several methods by modifying the `xmlrpc.php` file to allow Namespace and user manipulation and definition.

Following methods and their implementation is provided:

- `codeheap.createUser`
- `codeheap.modifyUser`
- `codeheap.deleteUser`
- `codeheap.createProject`
- `codeheap.modifyProject`

- codeheap.deleteProject
- codeheap.assignUserProject
- codeheap.removeUserProject

RPC\_TARGET constant (target URL for XML-RPC calls) must be defined to enable DokuWiki integration into Codeheap.

### 3.3 Subversion

Apache Subversion (often abbreviated as SVN, after the command name svn) is a software versioning and a revision control system founded and sponsored in 2000 by CollabNet Inc. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly-compatible successor to the widely used Concurrent Versions System (CVS).

SVN is most widely used Open Source Software CVS system in the world as of today.

#### Codeheap customization

Shell scripts manipulating with the SVN repositories implemented as Linux shell scripts are provided for Codeheap integration. These scripts are invoked via standard Java `java.lang.Runtime.exec(String[] commands)` method for OS native programs execution from within JVM. Implementation of the CodeheapConnector SCMConnectorInterface is provided in SVMSCMConnectorBean. Some methods like editUser and editProject are not supported by SVN Connector because there are no attributes to alter in SVN for given entity.

Constants SVN\_SCRIPT\_PATH and SVN\_REPOSITORY\_PATH pointing to the folder with the shell scripts and SVN repository must be defined in the database CONFIG table.

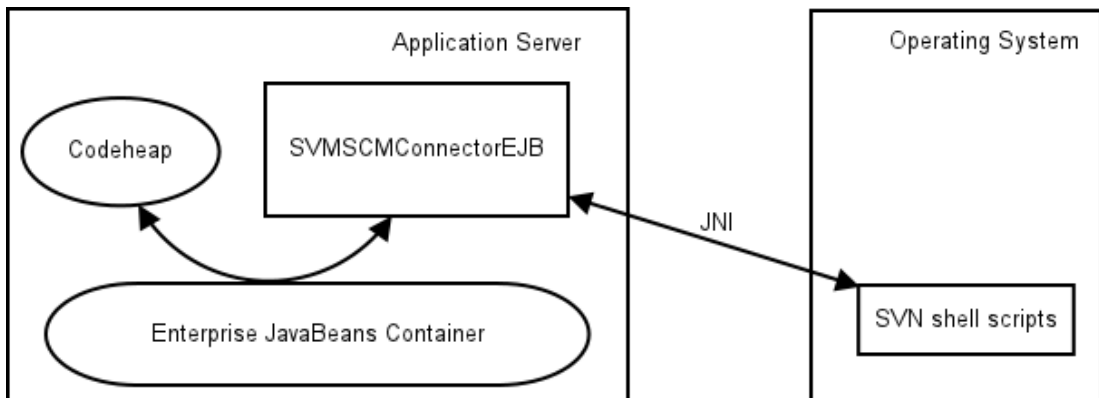


Image 7: Subversion Connector

Operating system user running the GlassFish AS must have the rights to access and manipulate the SVN repositories. This can be achieved with proper operating system rights configuration.

Following scripts for SVN repository manipulation are provided:

- create\_project.sh
- delete\_project.sh
- assign\_user\_project.sh
- remove\_user\_project.sh

## 3.4 Implementing New Subsystem

To implement different subsystems than the ones provided following steps must be taken:

1. Implement CodeheapConnector interface of choice as an Enterprise JavaBean
2. Package the implementation as a JAR file and include it in Codeheap application (as a new module in Codeheap Enterprise Archive) or deploy to Glassfish separately
3. Set the corresponding configuration parameter (WIKI\_CONNECTOR, BTS\_CONNECTOR or SCM\_CONNECTOR) so it contains the JNDI name of the newly implemented Enterprise JavaBean

If all interface methods are implemented, new subsystem is ready to be used.

One of the areas for future enhancements is to add new subsystems like Mailing List or Maven Repository directly to the CodeheapConnector API.

# 4 Codeheap User View

As described in the introduction Codeheap is designed to be a single point of administration for the project, so called Forge [4]. It has two main goals - to provide a way to administer the subsystems and to display the overview of the managed projects and users. First goal is achieved by CodeheapConnector module. This module provides interfaces to communicate with the each type of subsystem - Software Configuration management (SCM), Bug Tracking Software (BTS) and Wiki platform. How this module works is described in previous chapters. The second goal of the Codeheap application is to provide the overview of the managed projects and users. This is achieved in administration web interface where projects and users can be also managed and in public web pages that display the read only information about the managed entities.

User interface is divided into two separate parts - the public web that can be accessed by anyone and the administration part accessible only for registered and logged in users. In the public web pages the list of managed users and projects is shown with the ability to filter and sort lists by desired attributes and also the detail of the user or project can be shown with more versatile info about it. For the user his or her name, email address, description, user type and of course the list of projects he or she is participating on. Number of issues created or solved is shown also.

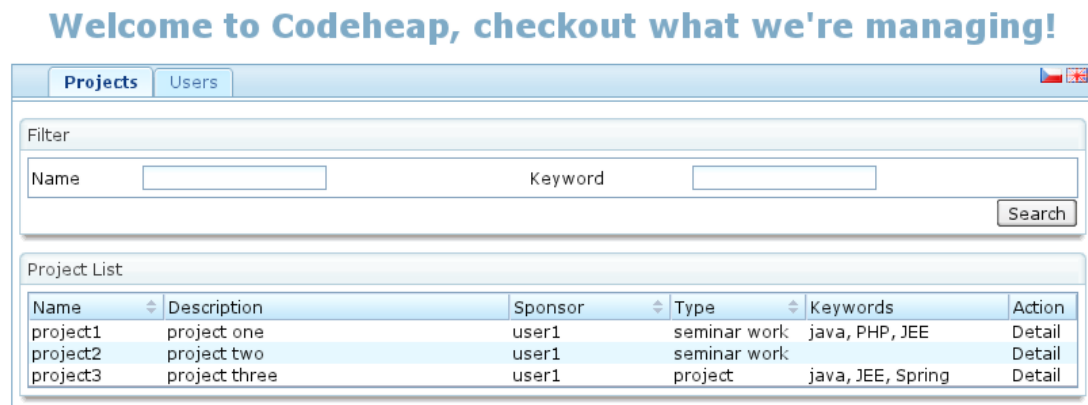


Image 8: Welcome Page

For the project its name, description, type, website URL, sponsor, license, status, start and end date, keywords and the list of committers are shown in the detailed view. Also the statistical information like number of issues so far is present.

Project Details	
<b>project3 project details</b>	
Description	project three
Sponsor	user1
Description	project three
Website	<a href="http://www.mff.cuni.cz">http://www.mff.cuni.cz</a>
License	GPLv2
Version	1
Enabled	<input checked="" type="checkbox"/>
Project type	project
Keywords	java, JEE, Spring
Issues Created	0
Issues Fixed	0
Assigned users	user1, user3

*Image 9: Project Detail*

The second part of the Codeheap web interface, the private zone, is far richer in content. Every user registered in the application can log in but not all actions are available for all users. This depends on the user type assigned to the user. User types and their roles are statically defined in the application. There are four types of users: root, administrator, project leader and regular user. Root has all the options available for him or her as this account is created as a part of the installation process. No other user can be created with this role.

The role with the most rights is the administrator which is almost identical to the root user but users in this role cannot administer the configuration parameters in Config tab.

Project leader can only add/remove members and change the project attributes. User can only login and manage his or her attributes like email, description etc.

Mapping of the roles between the user types and their roles is shown in the next table.

Private interface is divided into six tabs with the each one dedicated to manage logically different type of data. First one is just informational with the options to manage the current user properties like first and last name, description or email.

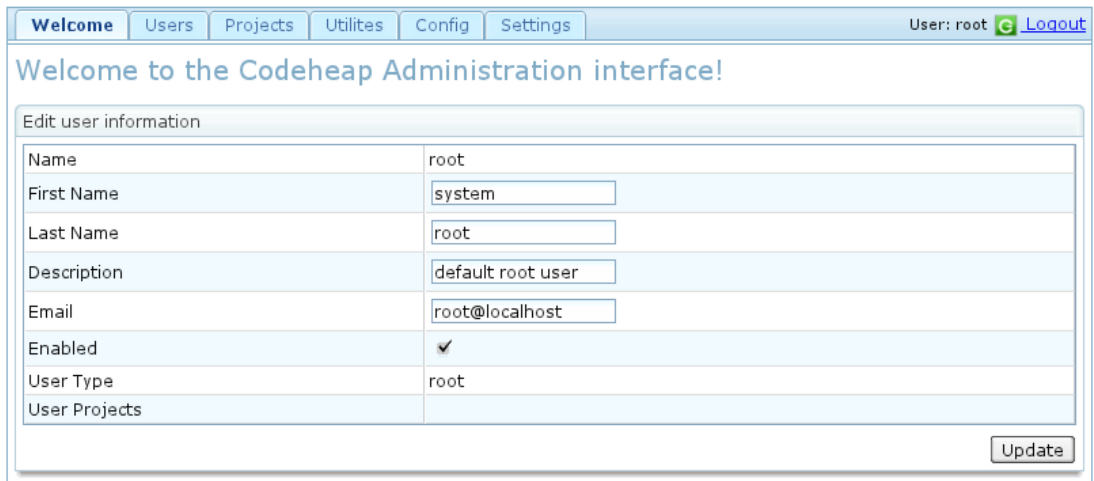


Image 10: Welcome Page of Private Web

The second and third tab (if accessible depending on the current user type) are used for user and project manipulation with typical CRUD<sup>1</sup> operations. Users can be created and modified. Projects can be created, modified, keywords assigned. Very important is option to add and remove participating users.

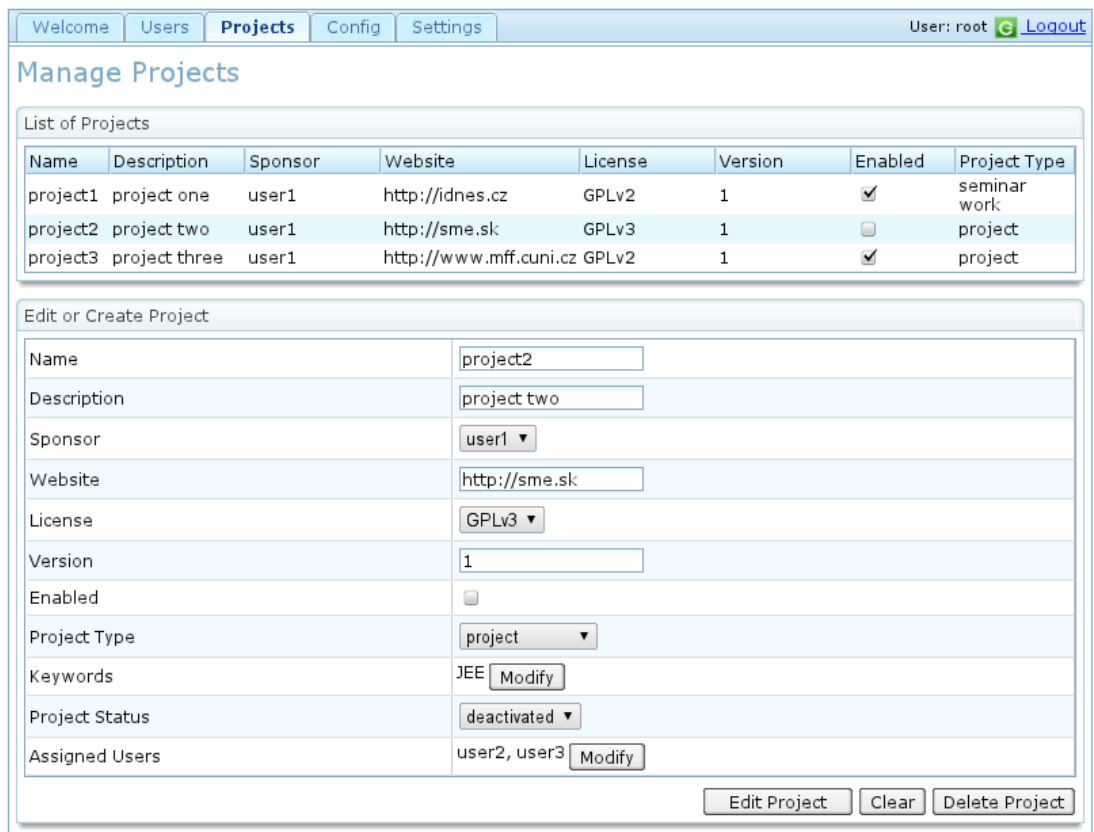


Image 11: Projects Management

<sup>1</sup> Create, Read, Update and Delete

The next tab Config is only accessible for the root user type where all system properties can be set. This includes the JNDI name of the Enterprise JavaBeans to use when accessing the subsystems and properties of the subsystems (like URL of the web service interface, username and password to be used when invoking web service, etc).

The last Settings tab can be used for the several codebook definitions (project type, licenses etc.) and subsystem diagnostics.

Whole user interface (public and private part) is fully localised and the session language can be chosen by clicking on the flag icon:

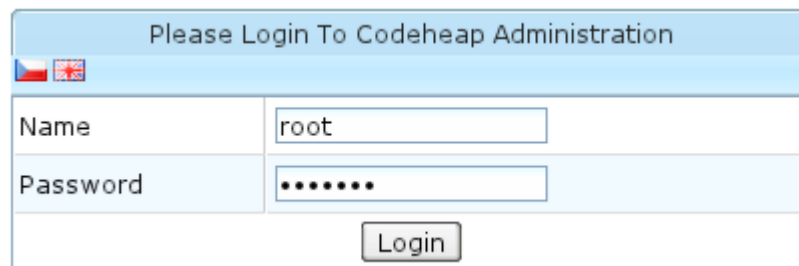


Image 12: Login Page

Localisation is done with the standard Java Resource Bundle files stored in *i3-label.properties* file for the default English. Also the *i3-label\_cs.properties* file is provided for the Czech translation. Each localisation key is constructed as *pageId\_key* pair for better organization.

For page translation ZK built-in method loaded with

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c" ?>
```

definition in the page header. To translate the value of the Label component:

```
<label value="{c:l('login_username')}" />
```

Chosen language is used during whole web session.

## 4.1 Implementation

For Codeheap implementation Netbeans IDE is used in version 6.9.1 with build in *Java Web and EE* and *PHP* plugins. To enable ZK integration REM plugin<sup>2</sup>

<sup>2</sup> <http://sourceforge.net/projects/rem1>

was used. GlassFish v3 is used as an Application Server. As a building tool we use the Apache Maven in its previous stable version 2.2.1 for better integration with used frameworks.

Codeheap is implemented in Netbeans IDE but the project structure is defined by Maven. Maven is a software tool for project management and build automation. It uses a construct known as a Project Object Model (POM) to describe the software project being built, its dependencies on other external modules and components. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging. Maven ability to dynamically download the dependencies and automatically defining the build order of the modules makes it very successful ancestor of the previously much used Apache Ant build tool.

Because of the Maven implied structure, there is one root Project Object Model *pom.xml* file in *codeheap* folder defining all the modules, dependencies and Maven plugin repositories used in all child modules. This allows centralized configuration for the dependencies versioning and maven plugin behaviour for all modules, e.g. defining final name of the packages, version of Java and EJB specification etc.

There are six modules in Codeheap:

1. codeheap-api
2. codeheap-ejb
3. codeheap-web
4. codeheap-ear
5. codeheap-connector-api
6. codeheap-connector-ejb

### **codeheap-api**

This module contains the public API in Java interfaces as defined for the application layer and accessible for the web module. There are also all the JPA entities defined there.

### **codeheap-ejb**

In this module the codeheap-api implementation in EJB objects is placed and also the *persistence.xml* file with the definition of the DB Connection Pool used in Application Server for the database connection for the underlying JPA framework EclipseLink.

### **codeheap-web**

Module of the presentation layer with all the ZUML pages and their backend Composers classes plus some ZKoss plugin classes, LoginFilter class that filters the unauthorized requests and standard web project configuration and servlet definition



in *web.xml*, *glassfish-web.xml* and *zk.xml* files. Localization files are placed in *src/main/webapp/WEB-INF* subdirectory.

### **codeheap-ear**

Packaging module taking care of the creation of the distribution EAR file and also the deploying functionality to the GlassFish via the Maven plugin is included for the development purposes. Installation folder of the GlassFish, domain name and other attributes must be defined in the modules pom.xml file to allow automatized deployment.

### **codeheap-connector-api**

Public CodeheapConnector API described in three interfaces, one for each subsystem. All of them are ancestors of the interface IGenericConnector defining the common methods that must be implemented by all subsystems. Some supplementary methods specific for each subsystem are placed into their interfaces, e.g. method to get the number of issues created for the project are only meaningful for the Bug Tracking System.

### **codeheap-connector-ejb**

Referential implementation of the CodeheapConnector API for the default subsystems SVN, MantisBT and DokuWiki as Enterprise Java Beans. Also the generated client classes for the Mantisconnect web service is placed here.

# 5 Installation

At first, all the subsystems must be installed, then customized with Codeheap plugins and set up with initial settings. Then the MySQL and GlassFish application server must be installed and configured as underlying layer for Codeheap. At last, Codeheap enterprise application archive must be deployed to the GlassFish. Afterwards, the Codeheap web interface is accessible via [http://<HOST\\_NAME>:8080/codeheap](http://<HOST_NAME>:8080/codeheap) for administrative section only for authorized users and [http://<HOST\\_NAME>:8080/codeheap/web](http://<HOST_NAME>:8080/codeheap/web) for the public part of the web pages.

Detailed installation manual is included on the attached CD.

## 6 Related work

The preferred way for the development of the free and open source software is taking places only on the internet. This is allowed with the good internet connectivity from the whole world. It implies more collaborative and less hierarchical structures of a team and the necessity of the tools to allow team communication. In last ten years many projects have evolved to provide the means for this. Some of them are intended to be used with specific development language (JavaForge, RubyForge). The others are designed to be used only for subprojects of a platform (Launchpad for Ubuntu, OpenMoko for open source mobile phones or Freedesktop.org for projects build on top of the Freedesktop.org).

The term forge refers to a common prefix or suffix adopted by the various software development management systems created after the example of the best known collaboration platform - SourceForge. Over two million users are registered in SourceForge and more than three hundred thousand projects as of today.

Software forge is a collaboration platform allowing collaborative software development over the Internet. A forge [4] platform aggregates a set of applications with integrated Web interfaces, and generally hosts multiple independent projects. Software developers who are registered as contributors to the hosted projects can then use the various project management tools, and software development tools.

Software forges have become popular and have proved successful in allowing development of a large number of free software projects in recent years.

Forge is by definition a web based application and only web browser is necessary for managing user account or project. On the other hand, to access the subsystem, regular clients are used, e.g. TortoiseSVN for SVN on Windows or Eclipse IDE plugin for MantisBT. This is very useful because developers do not have to learn how to use new tools but use the ones they are already familiar with.

### 6.1 Advantages of Codeheap

Codeheap is lightweight and extensible. Easy to use and modify collaboration platform designed for maximalised simplicity of use. All the subsystems are removable and can be replaced by other subsystem of the same type - e.g. SVN can be replaced by Git or other Software Configuration Management software. This is achieved by pluggable CodeheapConnector architecture.

Codeheap is built on latest but mature Java technologies. This has big benefits especially on user interface which is very user friendly and responsive because of the Ajax use in the ZK framework.

Another advantage is use of standard three tier architecture with application layer built on top of the most used middleware framework in the enterprise Java world - Enterprise JavaBeans. This has the advantage of the easily readable and extensible code structure implied from the Java standards and the design patterns [6] common in the Java world - Facade pattern for public API of application layer, Abstract Factory pattern for calls to the subsystems and Model View Controller (MVC) design pattern used typically for the three tier architecture. Model is represented by database and JPA entities, View is represented by ZUML pages and Composer classes. For Controller Enterprise JavaBeans are used.

Advantage of using standard architecture and design patterns is in a very steep learning curve for a potential new developer.

Codeheap is implemented primarily for environments where some project management tools are already in use. In this case, migrating to a completely new platform like SourceForge is too difficult. Codeheap modular architecture makes the transition smoother.

# 7 Conclusion

Codeheap is fully operational collaboration platform based on proven industry standards. It has a mature base architecture with the high emphasis on clear separation of concerns provided by Model View Controller (MVC) design pattern use in its three tier architecture. It is easily extensible to be used with other subsystems via its CodeheapConnector application programming interface (API). For Model, represented by database layer, standard Java Persistence API (JPA) and MySQL is used. JPA provides the mapping capabilities between JPA entities and database tables and also for the Many-To-Many, Many-To-One and One-To-One relations between the entities.

Enterprise JavaBeans (EJB) are used as Controller or in other words, Application layer. EJB is mature server side component framework used to encapsulate the business logic of the application. Enterprise JavaBeans are deployed to EJB Container in Application Server that takes care of their whole lifecycle. It also provides the naming service (JNDI), transactions and object pooling mechanism so a developer can only focus on implementing core business logic. Presentation view, or View in MVC pattern, is implemented in ZK framework. ZK framework is widely used Java based presentation framework for Rich Internet Applications (RIA) with strong emphasis on Ajax use with the rich set of components. Big advantage over other presentation frameworks is the ability to code almost everything in Java without knowledge of HTML or JavaScript. It is very well designed for fast prototyping so quick integration with other layers was possible. This speeds up the development and debugging of the entire application.

Java platform has very good interoperability frameworks available which make the integration of subsystems written in other languages feasible. This allows integrating several independent applications into one whole. In case of default built-in subsystems (MantisBT and DokuWiki), it is the integration with PHP scripting language. DokuWiki integration is done with Apache XML-RPC library that makes it possible to call the remote PHP procedures over the HTTP protocol. To call the Mantisconnect API published from MantisBT as web services, Apache Axis v1 library is used.

DokuWiki and MantisBT API was not suitable enough for all features necessary for Codeheap. That's why plugins for these applications had to be developed in PHP. This was particularly difficult because of the lack of coding standard for PHP. Both subsystems have completely different architecture and code structure. To be able to develop plugins it was necessary to understand the philosophy of the code of both subsystems

## 7.1 Difficulties

Biggest obstacle during Codeheap development was a necessity to develop the plugins for the MantisBT and DokuWiki because unfamiliarity with the PHP language. Getting to know the way these systems are implemented also took some time. Unfortunately, this was unforeseen at the beginning of the project because the provided API was expected to be sufficient.

Another problem was the obsolete architecture of the Mantisconnect web services. That's why the outdated Apache Axis library in version 1 had to be used. Overall, the interoperability tasks between the Codeheap in Java language and MantisBT and DokuWiki in PHP were not easy to overcome in transparent, clear way.

## 7.2 Future improvements

For the future, several aspects of the application can be improved.

- more subsystem to be integrated - mailing lists, forums
- Maven repository to enable project collaboration
- better integration with the subsystems providing more information about the managed projects and users
  - statistics and graphs for the public web pages showing the activity on the projects
  - ranking system for the projects and users to be able to show most active projects and or users
- utilites to import or export data from subsystems for easier system restoration

## 8 References

1. ZK framework, <http://www.zkoss.org>
2. Beanshell, <http://en.wikipedia.org/wiki/Beanshell>
3. Convention over Configuration,  
[http://en.wikipedia.org/wiki/Convention\\_over\\_configuration](http://en.wikipedia.org/wiki/Convention_over_configuration)
4. Forge - [http://en.wikipedia.org/wiki/Forge\\_\(software\)](http://en.wikipedia.org/wiki/Forge_(software))
5. Java EE, <http://www.oracle.com/technetwork/java/javaee>
6. Design Patterns,  
[http://en.wikipedia.org/wiki/Design\\_pattern\\_\(computer\\_science\)](http://en.wikipedia.org/wiki/Design_pattern_(computer_science))
7. Enterprise JavaBeans, <http://www.oracle.com/technetwork/java/javaee/ejb>
8. Eckel, B.: Thinking in Java, <http://www.mindview.net/Books/TIJ/>

# 9 List of Abbreviations

- API - Application Programming Interface
- JVM - Java Virtual Machine
- JEE - Java Enterprise Edition
- EJB - Enterprise JavaBeans
- JPA - Java Persistence API
- JDBC - Java Database Connectivity
- XML - Extensible Markup Language
- JAX-RPC - Java API for XML-based RPC
- JAX-WS - Java API for XML Web Services
- JNDI - Java Naming and Directory Interface
- JNI - Java Native Interface
- PHP - PHP: Hypertext Preprocessor
- HTML - Hypertext Markup Language
- XUL - XML User Interface
- AS - Application Server
- ACL - Access Control List
- EAR - Enterprise Archive
- JAR - Java Archive
- IDE - Integrated Development Environment
- GUI - Graphical User Interface



# 10 List of Images

Image 1: Codeheap Architecture.....	4
Image 2: ZK Architecture.....	6
Image 3: Application Layer.....	7
Image 4: Database model.....	9
Image 5: MantisBT Connector.....	11
Image 6: DokuWiki Connector.....	13
Image 7: Subversion Connector.....	14
Image 8: Welcome Page.....	16
Image 9: Project Detail.....	17
Image 10: Welcome Page of Private Web.....	18
Image 11: Projects Management.....	18
Image 12: Login Page.....	19

# 11 Attachment

Attached CD has the following contents:

- Installation manual
- Installation packages for MantisBT and DokuWiki
- Codeheap plugins for MantisBT and DokuWiki
- SVN manipulation scripts
- MySQL and GlassFish packages
- SQL scripts to create the database schema
- Insert scripts to populate the schema with default values
- Codeheap archive file to be deployed to GlassFish
- Codeheap source files
- User and Programming documentation
- This thesis in PDF file

