

Development of web database application framework based on open source

Bc. Tomáš Tureček

Master thesis
2009

 **Tomas Bata University in Zlín**
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš TUREČEK**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Development of web database application
framework based on open source**

Zásady pro vypracování:

1. Analysis of project targets.
2. Analysis of open source systems and selection of used technologies.
3. Analysis of requirements on web and database applications.
4. Object oriented access to development.
5. Implementation of application server.
6. Implementation of application layer for work with relation database.
7. Sample applications implementation as practical usage of framework.
8. Framework evaluation and possible future evolution.

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. DiBona C., Stone M., Cooper D. **Open Sources 2.0: The continuing evolution.** O'Reilly, London. ISBN 10: 0-596-00802-3
2. Fogel K., **Producing open source software: How to run a successful free software project.** O'Reilly, London. ISBN 10: 0-596-00759-0
3. Lutz M., **Learning Python, Third Edition.** O'Reilly & Associates, 2007. ISBN 10: 0-596-51398-4
4. Lutz M., **Programming Python, Third Edition.** O'Reilly, 2006. ISBN 10: 0-596-00925-9
5. Martelli A., Martelli Ravenscroft A., Ascher D., **Python cookbook, Second Edition.** O'Reilly, Cambridge, 2005. ISBN 10: 0-596-00797-3
6. DuBois P., **MySQL Cookbook, Second Edition.** O'Reilly, Cambridge, 2007. ISBN 10: 0-596-52708-X
7. Bowen R., Coar K., **Apache cookbook, Second Edition,** O'Reilly, Cambridge, 2007. ISBN 10: 0-596-52994-5

Vedoucí diplomové práce:

Ing. Petr Šilhavý

Ústav aplikované informatiky

Datum zadání diplomové práce:

20. února 2009

Termín odevzdání diplomové práce:

27. května 2009

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Práce je zaměřena vývoj frameworku pro tvorbu webových a databázových aplikací. Obsahuje analýzu požadavků a řešení, popis použitých technologií a implementaci v programovacím jazyce Python. Implementace vlastního frameworku je složena z databázové knihovny pro práci s MySQL databází a rozříšení modulu Mod_Python pro webový server Apache. Součástí implementace je ukázková CMS aplikace vytvořena v tomto frameworku.

Klíčová slova: python, apache, mysql, web server, programování, application, framework, CMS

ABSTRACT

The main project target is development of web database application framework. The project contains analysis of requirements and solution, description of used technologies and implementation written in Python programming language. Implementation of the framework consists from database library for working with MySQL database and extension of Mod_Python module for Apache web server. Part of the implementation is sample CMS application using the framework.

Keywords: python, apache, mysql, web server, programming, application, framework, CMS

Thank to supervisor of this master thesis, Ing. Petr Šilhavý, for suggestions and comments before and during the project working process.

licence agreement

CONTENTS

INTRODUCTION	8
I THEORY	8
1 USED TECHNOLOGIES	10
1.1 OPEN SOURCE.....	10
1.2 LINUX	10
1.3 DEBIAN OS	10
1.4 APACHE	10
1.5 PYTHON.....	11
1.6 MOD_PYTHON	11
1.7 SQL.....	11
1.8 MYSQL	11
1.9 POSTGRESQL.....	12
1.10 HTML.....	12
1.11 JAVASCRIPT	12
1.12 CSS	12
1.13 SUBVERSION	12
1.14 LAMP.....	13
2 FRAMEWORK	14
3 PROGRAMMING ARCHITECTURES	15
3.1 OBJECT ORIENTED PROGRAMMING	15
3.2 MODEL - VIEW - CONTROLLER	15
3.3 ORM - OBJECT RELATION MAPING	15
II ANALYSIS	16
4 REQUIREMENTS	18
4.1 WEB APPLICATION AND DATABASE APPLICATION BUILDING SUP- PORTS.....	18
4.2 SECURITY	18
4.3 MULTI-PLATFORM	18
4.4 TEMPLATING SYSTEM SUPPORT	18
4.5 LOCALIZATION	18
4.6 SEO.....	18
4.7 DOCUMENTATION.....	19

5	PROJECT TARGETS	20
6	TECHNOLOGIES SELECTION	21
6.1	LAMP	21
6.2	PROGRAMMING LANGUAGE.....	21
6.3	WEB SERVER	21
6.4	DATABASE SERVER	21
6.5	DATABASE INTERFACE.....	21
6.6	CONNECTION OF USED TECHNOLOGIES.....	22
7	FRAGMENTATION	23
7.1	THE FRAMEWORK.....	23
7.2	SAMPLE USAGE.....	23
7.3	DOCUMENTATION.....	24
III	PRACTICE	24
8	PYROH	26
8.1	INTRODUCTION	26
8.2	CLASS DB.....	26
8.3	CLASS LIST	28
8.4	CLASS RECORD.....	29
8.5	CLASS COLUMN	31
8.6	SAMPLE USAGE OF PYROH LIBRARY	32
9	MANITOU	34
9.1	DESCRIPTION	34
9.2	MOD_PYTHON	35
9.3	CLASS APP.....	35
9.4	CLASS REQUEST	35
9.5	CLASS MYSERVER	36
9.6	CLASS LOCALE.....	36
9.7	MANITOU SAMPLE USAGE.....	36
10	CASCADING STYLESHEETS	37
10.1	DESCRIPTION	37
10.2	LAYOUT STYLES.....	37
10.3	THEMES STYLES.....	37
11	SAMPLE APPLICATION - CMS	38
11.1	DESCRIPTION	38

11.2	SEO	39
11.3	CLASS CMS	39
11.4	FRAMEWORK USAGE DIAGRAMS	42
11.5	TEMPLATES	43
11.6	CONFIGURATION	43
11.7	DATABASE STRUCTURE	46
11.8	TABLES DEFINITION.....	47
CONCLUSION.....		49
SEZNAM POUŽITÉ LITERATURY		49
LIST OF ABBREVIATIONS		52
SEZNAM OBRÁZKŮ		54
SEZNAM TABULEK		55
APPENDICES.....		56

INTRODUCTION

Development of web database application usually consists from implementation of general parts. Each part is designed in steps common for more applications.

These common steps of part's implementation can be included in one software framework.

So the common application core can be used in more applications.

Using this framework or only some of its parts by particular application designing speed up the development and bring readability and transparency into source code writing.

I. THEORY

1 USED TECHNOLOGIES

1.1 Open source

Open source software is kind of software which source code and other rights are provided under a software license which permits users to use and modify the software. Users can redistribute it in modified or unmodified forms. The open source software is developed by organizations or public collaborative manners. [1]

1.2 Linux

Linux is a Unix-like operating system. It is based on the Linux kernel which is core of the system. Development of linux is one of the most significant open source collaboration project. Linux is ported on a wide variety of hardware architecture such as pc, mobile phones, embedded devices and super computers. The core of the system - linux kernel was written by Linus Torvalds in 1991. Other parts of the system come from the GNU operating system announced by Richard Stallman in 1983. [2]

1.3 Debian OS

Debian GNU/Linux is one of the most popular linux operation system. There are two main goals of debian system. It is stable version of used software and very wide software repository. [2]

Current version of Debian (Lenny 5.0) has over twenty five thousand of software packages for each of twelve computer architectures ready for install and use. Each package contains one or more software products. There is Aptitude packaging system as a part of the system which handles packages dependencies. [9]

1.4 Apache

Apache is well-known open source software web server for many operating system platforms. The main functionality of software web server is to serve web contents to clients (web browsers).

Apache supports a lot of features, mainly implemented as compiled modules which extend the basic functionality from Apache. Some common language interfaces support mod_perl, mod_python, Tcl or PHP. The most used authentication modules in Apache web server are mod_access, mod_auth, mod_digest or mod_auth_digest. A sample of other features include SSL and TLS support (mod_ssl), a proxy module, a URL rewriter (known as a rewrite engine implemented under mod_rewrite) or custom log files (mod_log_config) and filtering support (mod_include and mod_ext_filter).

There is module mod_gzip which provides compression methods on Apache implemented to help with reduction of the size of web pages served over HTTP. Apache logs can be analy-

zed through a web browser using free scripts scanning. Virtual hosting allows to serve many different websites by single Apache instance (installation). [7]

1.5 Python

Python is a high-level programming language. It means that it contains high-level abstraction which is more easier to use and more portable across platforms.

The execution model of python is interpreted. The source code is read by interpreter and directly executed without compilation. [5]

Python uses whitespace indentation as block delimiters which is very unusual in comparison with other popular programming languages. It makes the code more simple to write and easier to read among programmers.

Python has support of multiple programming architectures such as object oriented or functional. The python interpreter provides fully dynamic type system and automatic memory management. [3]

Python is often used as a scripting language like other dynamic languages such as Perl and Ruby. The language has a community-based development model which is managed by the non-profit Python Software Foundation. [4]

1.6 Mod_python

Mod_python is an additive module for Apache web server. It makes possible to embed the Python interpreter within the web server. So the request execution is forwarded from Apache into the Python application which handles the response and returns it back to Apache. [16]

With mod_python you are able to develop web-based applications written in Python language. The applications will run many times faster than traditional CGI. They will have access to Apache internals and will have ability to retain database connections and other data between hits. [4]

1.7 SQL

SQL(Structured Query Language) is a database querying computer language for database querying and managing. It allows selecting, inserting, modifying and deleting data in relation database and managing the database structure. [6]

The SQL was first developed by IBM in 1974. SQL was standardized by the American National Standards Institute(ANSI) in 1986 and later by the International Organization for Standardization(ISO). [13]

1.8 Mysql

MySQL is a relational database management system (RDBMS). MySQL runs as a server providing multi-user access to data stored in databases. The project's source code is available

under terms of the GNU General Public License, as well as under a variety of proprietary agreements. [6]

1.9 PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS). It is released under a BSD-style license and is thus free software.

PostgreSQL is kind of enterprise database system. As with many other open-source programs, PostgreSQL is not controlled by any single company, but has a global community of developers and companies to develop it. [13]

1.10 HTML

HTML (HyperText Markup Language) is a markup language used by web sites creation. It provides a structure description of text-based information in a document by denoting certain text as links, headings, paragraphs, lists and others. It describes interactive forms, embedded images and other objects. It consists from tags surrounded by angle brackets. [10]

1.11 Javascript

Javascript is an interpreted scripting language. It can script HTML web pages. It bring dynamic into static pages and affects behavior of web browsers. JavaScript, despite the name, is essentially unrelated to the Java programming language even though the two do have superficial similarities. Both languages use syntaxes influenced by that of C syntax, and JavaScript copies many Java names and naming conventions. [12]

1.12 CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation (that is, the look and formatting) of a document written in a markup language. Its most common usage is to style web pages written in HTML or XHTML, but the language can be applied to any kind of XML document, including SVG and XUL. CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as colors, fonts, and layout. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting. [11]

1.13 Subversion

Subversion (SVN) is a file versioning system with client to server architecture. SVN is well-known in the open source community and is used as main versioning system on many open source projects. It can handle text and binary files.

Subversion uses the interfile branching model to handle branches and tags. Each version in SVN versioning system has own version number. Branching is the possibility to isolate changes in a separated line of development. Tagging is the possibility to associate additional information with a particular revision. [8]

1.14 LAMP

The acronym LAMP means a software solution stack, usually free and open source software, used to run dynamic web sites or servers. The acronym became by names of technologies connected with the stack. Linux, referring to the operating system. Apache, the web server. MySQL, the database management system (or database server). One of several scripting languages as development language: Perl, PHP or Python (sometimes included Ruby). The combination of these technologies is used primarily to define a web server infrastructure and a programming paradigm of web application development.

Usually the LAMP is distributed as a software package. [17]

2 FRAMEWORK

In general framework is a model on which is based lots of application. It is a common application core used by development.

Development of web database application usually consists from implementation of general parts with common steps for more applications. Each part is designed in steps common .

These common steps can be a part of the framework. So the common application core can be used in more applications.

In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes.

Open source framework can be shared between programmers.

Using a framework speed up application development and bring readability into source code writing. [14]

3 PROGRAMMING ARCHITECTURES

3.1 Object oriented programming

Object oriented programming is programming method for application designing.

An object-oriented program can be viewed as a collection of cooperating objects, opposed to the conventional model. In the conventional model a program is seen as a list of tasks to be processed.

In OOP the functionality of the program is described by objects and their interactions.

Each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can have own functions, procedures and parameters. The actions on these objects are closely associated with the object.

There are lots of features connected with objects such as inheritance, abstraction, encapsulation, polymorphism, decoupling. [4]

3.2 Model - View - Controller

Model view controller (MVC) is an architectural pattern used in software engineering.

Successful use of the pattern isolates business logic from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules. Modification of one of these parts is without affecting the other part.

In MVC, the model represents the information (the data) of the application. The view corresponds to elements of the user interface such as text, checkbox items, and others. And the controller manages the communication of data and the business rules used to manipulate the data to and from the model.

In development of web database applications model can be an object representation of the database schema and business logic. It is used as a layer for working with data. View is used for data output formatting and visualization. Usually the view layer is an templating system. Controller layer handles the interaction of particular parts of application. Controller communicate with both model and view layer.

MVC is one of the most popular program architecture used to web application development. [4]

3.3 ORM - Object relation mapping

Object-relational mapping (ORM) is a programming architecture for converting data between relational databases and object-oriented programming languages.

ORM creates a virtual object database that can be used within the programming language as a library. Usually it is easier and more readable to describe a data selection with objects and their interactions then write queries in SQL language and handle the data fetching.

Object-relational mapping is used to implement the first approach in communication with database. [4]

II. ANALYSIS

4 REQUIREMENTS

4.1 Web application and database application building supports

The framework has to contain a combination of base programming practices used by development of web and database applications. It is request and response handling by method binding on URL address as base part of a web application.

Layer enabling simplification of working and communication with data source is a basic tool used by development of database application. There should be support of using transaction by data storing to the data source.

The framework should be designed as the MVC programming architecture. There should be object oriented access to development.

4.2 Security

Web application has to be able to support secure communication over SSL. It has to prevent foreign source code interpreting. The architecture of database part should avoid SQL injection.

4.3 Multi-platform

Application based on the framework should run on many platforms so the framework should make it possible. On the client's side multiplatform must be provided by code optimization for multiplatform web browsers.

4.4 Templating system support

The framework should support a templating system. It is an important part of web applications. It provides response structure formatting.

4.5 Localization

There should be implemented a possibility of localization of static text used in web application templates.

Localization has to be easily editable so it is recommended to use localization based on keys and translation stored in comma separated values(CSV) file because of its easy editing in some calc manager.

4.6 SEO

The framework should contain support of base SEO practices like URL rewriting. So the framework has to allow development of application which can be search engine friendly.

4.7 Documentation

There should be a documentation of the framework describing program functionality and source code.

5 PROJECT TARGETS

The main target is to develop web database application framework for an easy creation of applications.

The framework should contain all parts described in requirements section.

The framework has to enable to run parts of the application as web server request and as script call.

6 TECHNOLOGIES SELECTION

6.1 LAMP

All technologies selected to use are connected with one solution stack, LAMP. It wasn't one of the targets but the framework can be termed as based on LAMP which may increase its usability and interest of possible users.

6.2 Programming language

From the beginning there was need of using interpreted programming language. Python allows easy writing of object oriented programs because of strong object orientation programming support. Thanks the whitespace usage requirement because of delimiting blocks of code python has its own specific indentation. It improve source code reading between programmers. So it is the next advantage for using python language. Using the exceptions improves debugging possibilities and autocompletion features of enhanced python interpreter ipython speed up the development.

6.3 Web server

As a web server it was choosed Apache version 2 because of its robustness and easy configuration. Communication between the Apache server and the application framework written in python will be provided by implementation of api using the mod_python module. PSP was selected as base templating system because it is part of the mod_python module. The web server request will be forwarded into the application connected on some location of the virtual host. There is requirement to use module mod_rewrite to enable using SEO url addresses.

6.4 Database server

As database storage engine was choosed choosed Mysql server. There is support of Mysql version 5 in the framework because of foreign key constraints. Foreign keys usage is one of the most important feature the database layer will work with. There was requirement to use transactions on database server and Mysql 5 has a support of transactions.

6.5 Database interface

It is used python module MySQLdb as a interface to the MySQL database. It means all queries are executed and data fetched by calling methods of this interface.

6.6 Connection of used technologies

When comes some request some location of the Apache server and the location is set up as the application handler, the request processing will be forwarded into our application. It will be executed method of our applicaiton depended on the request parameters. Instance of database object will be create in each application method and it will be called instance method with data from request appropriate the function of the application method. The response can contain either raw data from the application method or data generated by template assigned top the method or specified in the method.

7 FRAGMENTATION

There is base fragmentation of the project. It is separated into framework as the main part of the project, sample usage as framework demonstration and project documentation. Each fragment is described below.

7.1 The framework

The framework is split into two parts. Each part can be used independently on the other part. The parts are database layer named Pyroh and web application layer named Manitou. Although the target is create web application framework, both of these parts can be used for example in development of the desktop application. Next content of the work is project description and sample usage of the project.

7.1.1 Database layer

Database layer will consist of the sql library communicating with database server. The library will be developed as one way ORM concept and it will use some database interface for selected database engine. It means the library will provide classes appropriate the database structure but it will be able to affect only data, not the structure of the database tables. The layer should be split into class depended on the hierarchical structure of the database. There should be classes describing the database, tables, rows, columns and working with data. Instance of the classes will be in parent - child relation depended on the type of the object or on the database structure relation described by foreign keys.

7.1.2 Web application layer

web application layer will be implemented as framework based on mod_python apache2 web server api. It can be called as part of the web server request or as script. This layer should be logically split into modules described by classes. There should be module for request handling, default application methods and application translation. Web application layer should be able to use some templating system.

7.2 Sample usage

Each part of the framework will contain an example. It is CMS application as sample program using the framework parts as complex. It should basically demonstrate the usage and functionality of the framework.

7.3 Documentation

Documentation should describe program functionality and source code of the framework, its parts and the sample application. Main part of the documentation is the thesis.

III. PRACTICE

8 PYROH

8.1 Introduction

Pyroh (Python Relation Object Handler) is database library for Mysql and PostgreSQL databases which I wrote in python programming language. The main usage of the library is to work with data from database via foreign keys relations without writing sql queries manually.

The working with data consists with using objects which have methods for manipulations. Some methods called on this objects are executed recursively by walking throw parents to childs objects. Sample of this functionality is saving opened objects into database. There is used sql store queries in one database transaction.

8.2 Class DB

8.2.1 Description

Instance of DB class handles database connection and operation over connected database. If instance is created, autoload methods for all database tables are prepared. We work with the instance through the member methods described below.

8.2.2 Method `__init__`

DB class object constructor. This method assign constructor arguments into object parameters. Parameter `app` stores parent object reference. We can work with parent object throw this reference. Parameter `db` is name of the database we want connect and work with. Parameter `user` is user name used by connection to database. Parameter `passwd` is password of the user used by connection to database. Parameter `character_set` is is value of character encoding we assign to the communication with database after successfull connection. Parameter `prefix` is table name prefix used in database. Variable `c` returns reference to sql database cursor.

8.2.3 Method `construct`

For each row of rows set returned by the `fetch_tables` method we call two times method `declare`. First call creates new class according to table name inherited from class `List`. Second call creates new class according to table name inherited from class `Record`.

8.2.4 Method `fetch_tables`

It returns rows set with informations of all tables in the database. It is used mysql special query `'SHOW TABLES'`.

8.2.5 Method `fetch_columns`

It returns rows set with informations of all columns from selected table in the database. It is used mysql special query 'SHOW FULL COLUMNS FROM TABLE'.

8.2.6 Method `fetch_autoloaders`

The argument table of this function is table name. This method returns informations about tables which are in related by foreign keys with our table. It returns tables which either depends on our table or our table depends on it. Information about database table keys are stored in table 'key_column_usage' of internal mysql database 'information_schema'. We return intersection between query we select all rows where table column 'table_name' match our table and query we select all rows where table column 'referenced_table_name' match our table.

8.2.7 Method `declare`

Creates virtual class for selected database table. It declares new class which inherit from extend class. It sets attribute root with pointer to root class and attribute table with table name from database.

8.2.8 Method `is_simple_key`

This method checks if the selected column of our table is the primary key of database table and if it is a simple key (it means if the key is only on one column). Information about database table keys are stored in table 'key_column_usage' of internal mysql database 'information_schema'. We select all rows from the table where column 'table_name' matches our table we explore and column 'constraint_name' matches value 'PRIMARY'. If our column has simple key returned rows count has to be equal one, we return boolean value.

8.2.9 Method `classnamefromtable`

Method `classnamefromtable` returns classname for created class according to table name whereupon the new class is build. The PYROH library expects database table names in plural form (f.e. 'items'). If the new class extends class 'Record' we reduce the last char of te table name, so we get the name like 'item'. If parameter prefix was passed over constructor, we add the prefix to the classname. We return new name of the class to be created.

8.2.10 Method `version`

Return version of database server.

8.3 Class List

8.3.1 Description

This class makes list operation of selected records from one table.

8.3.2 Method `__init__`

Class List object constructor. It has two arguments. Argument parent is reference to the parent object. Argument references is array of referenced column names from parent object. We assign parameter parent and parameter references with constructor argument, we create parameter list as empty array and we call member method load.

8.3.3 Method load

First we get the clause of the query by calling member method `get_clause`. Clause consists array of columns. If the array is empty, we generate full select query without restrictive condition in where clause. If the array is not empty, we generate select query with restrictive condition in where clause for each column in clause array. Then we execute generated query and for each returned row we add new instance into array list by calling method `add`.

8.3.4 Method `get_clause`

We return array of all column instances from our table which are included in array references passed in constructor method.

8.3.5 Method add

First we create instance of class based on class Record. Then we assign into columns of the instance values from data from argument of the method. We append the instance into member array list and return the array.

8.3.6 Method save

For each instance in member array list we call instance member method `save`. It means we save each instances of list as records of database table.

8.3.7 Method `tohash`

It returns array of associative arrays returned by calling method `tohash` on each instance array list.

8.3.8 Method search

Method search returns instances of array list depended on search argument. We specify the search argument as associative array with columns and values. We walk throw all instances and if an instance don't correspond to all search condition, we don't return it.

8.3.9 Method find

We call method search. If returned array instances count is less then one, we return new instance. Otherwise we return first instance of returned array.

8.3.10 Method assing_args

Method assign_args provides explicit assignment of data from method argument. It creates object model according to data structure and assign the data values into created objects.

8.4 Class Record

8.4.1 Description

This class makes row operation of one selected record from one table. This class is mapped on one table row.

8.4.2 Method __init__

It is class constructor. Argument keys is associative array of the primary key column names and their values to be used to build load query. Argument references is associative array of referenced column names from parent object. Argument references is associative array of referenced column names from parent object. Argument sql is sql query to be execute to get data for the new instance. Parameter columns is associative array of column informations and values. Parameter autoloaders is associative array of autoloaders. Parameter load indicate if the instance values are loaded from database table record. If argument keys is specified, we call method rulekey to assign values from keys into columns. If argument sql is specified, we call method load_sql, otherwise we call method load.

8.4.3 Method get_columns

Method returns all column instances of the columns property.

8.4.4 Method get_key

This method returns columns from method get_columns which are marked as part of primary key.

8.4.5 Method get_autoloaders

Returns autoloaders property of the instance.

8.4.6 Method autoload

Returns selected autoloader by name.

8.4.7 Method create_autoloaders

Method creates all possible autoloaders. First are fetched tables informations important for autoloaders creation by calling method `fetch_autoloaders` of DB class instance. For each table in fetched tables informationis we create autoload method as instance property named by table name.

8.4.8 Method create_columns

If the base class we inherit from doesn't have attribute columns defined, it is created by calling DB class instance method `fetch_columns`. For each columns informations in base class attribute columns is created new column instance.

8.4.9 Method reference_values

For each refence in instance references, value of referenced instance column is assigned into instance column.

8.4.10 Method rulekey

This method assign values from data returned by calling method `get_key` into instance columns values.

8.4.11 Method assing_args

Method `assign_args` creates object model based on the data structure in the argument. There are assigned values from data structure for each new created instance.

8.4.12 Method load

Method `load` get data from database into Record class instance by executing sql query constructed by instance primary key columns values returned by instance method `get_clause`. Clause consists of array of column instances.

8.4.13 Method load_sql

Argument sql contains sql query to be execute. This method load data from database into Record class instance by executing sql query from method argument.

8.4.14 Method get_clause

Returns hash of columns and names returned by instance method get_key.

8.4.15 Method check

Method check calls instance method check on each column object of the instance.

8.4.16 Method save

Save instance of class Record as table record. First is called method reference_values to get values of inherited foreign key columns from columns of parent instance. Than it check the _loaded instance param. It means whether the record being save is loaded from database or not. If it is loaded we use update sql query. Otherwise we use insert sql query to save the data into database. Than is called method save on instances of all opened autoloaders. So the method save recursive all opened autoloaders instances. All sql queries are executed in one sql transaction.

8.4.17 Method tohash

Method tohash returns hash (dictionary) of instance columns and their values.

8.5 Class Column

8.5.1 Description

Class Column represents table column of one row and provides operations with it's value by member methods.

8.5.2 Method __init__

Argument parent is reference to the parent instance. Argument structure is associative array of informations about column such as data name, type, value or primary key mark. The method provides data assignment into new created instance.

8.5.3 Method get_value

Returns value of the parent instance property with the same name as column has.

8.5.4 Method `get_db_value`

Returns modified return of method `get_value` in database friendly format.

8.5.5 Method `set_value`

Set the value of the parent instance property with the same name as column has.

8.5.6 Method `get_format`

Returns formatting string used by building queries with parameters.

8.5.7 Method `get_update`

Return sql update string of the column. Default value of the mark argument is character equal.

8.5.8 Method `get_where`

Returns the sql string of the column used in where clause of the query.

8.6 Sample usage of PYROH library

```
Python 2.5.2 (r252:60911, Jan  4 2009, 21:59:32)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys                                     #importing sys module
>>> sys.path.append('/usr/share/web')              #importing pyroh library
>>> import pyroh                                   #importing pyroh library
>>> db = pyroh.DB(db='cms')                         #creating database instance
{'collation': 'utf8 general ci', 'comment': 'UTF-8 Unicode',
'mbminlen': 1, 'name': 'utf8', 'mbmaxlen': 3}
generating classes
... defining class Articles
... defining class Article
... defining class Articletexts
... defining class Articletext
... defining class Languages
... defining class Language
... defining class Pages
... defining class Page
... defining class Pagetexts
... defining class Pagetext
... defining class Users
... defining class User
>>> page = db.Page({'idpage':1})                   #creating record intance of table page
creating record object <class 'pyroh.Page'>
... creating column: idpage
... creating column: idparent
... creating column: iduser
... defining autoloader: page
... defining autoloader: user
... defining autoloader: articles
... defining autoloader: pages
... defining autoloader: pagetexts
loaded
{'iduser': 1L, 'idpage': 1L, 'idparent': None}
>>> page.pagetexts().find({'idlanguage':2}).tohash() #finding page pagetext by selected value
{'content': '<p>linuxova dostribuce</p><h3>xbxdfgsdfgsdfg <br /></h3>',
'url': 'web', 'idlanguage': 2L, 'idpage': 1L, 'title': 'web'}
>>> page.pagetexts().list[0].language().tohash()   #returning language of first pagetext from page
{'symbol': 'en', 'idlanguage': 1L, 'name': 'english'}
```

Fig. 1. Sample usage of PYROH library

It is a sample usage of PYROH library. It shows working with sample database (database structure described in Sample applicaiton section). The code above (fig. 1) is output from python interpreter running in the interactive mode. First we import the library and create instance of selected database. Information about database encoding and collation are printed into standard output. For each table in the database are created two classes, class according to table name inherited from class List and class according to table name inherited from class Record.

9 MANITOU

9.1 Description

Manitou is web application framework based on mod_python apache2 web server api. It provides application call as part of the web server request (fig. 1) and application call as script. Call as part of the server request is mapped on the Apache request by implementation of mod_python api. Call as script can be used for administrator's work with the application or for scheduled automatic run of a part of the application.

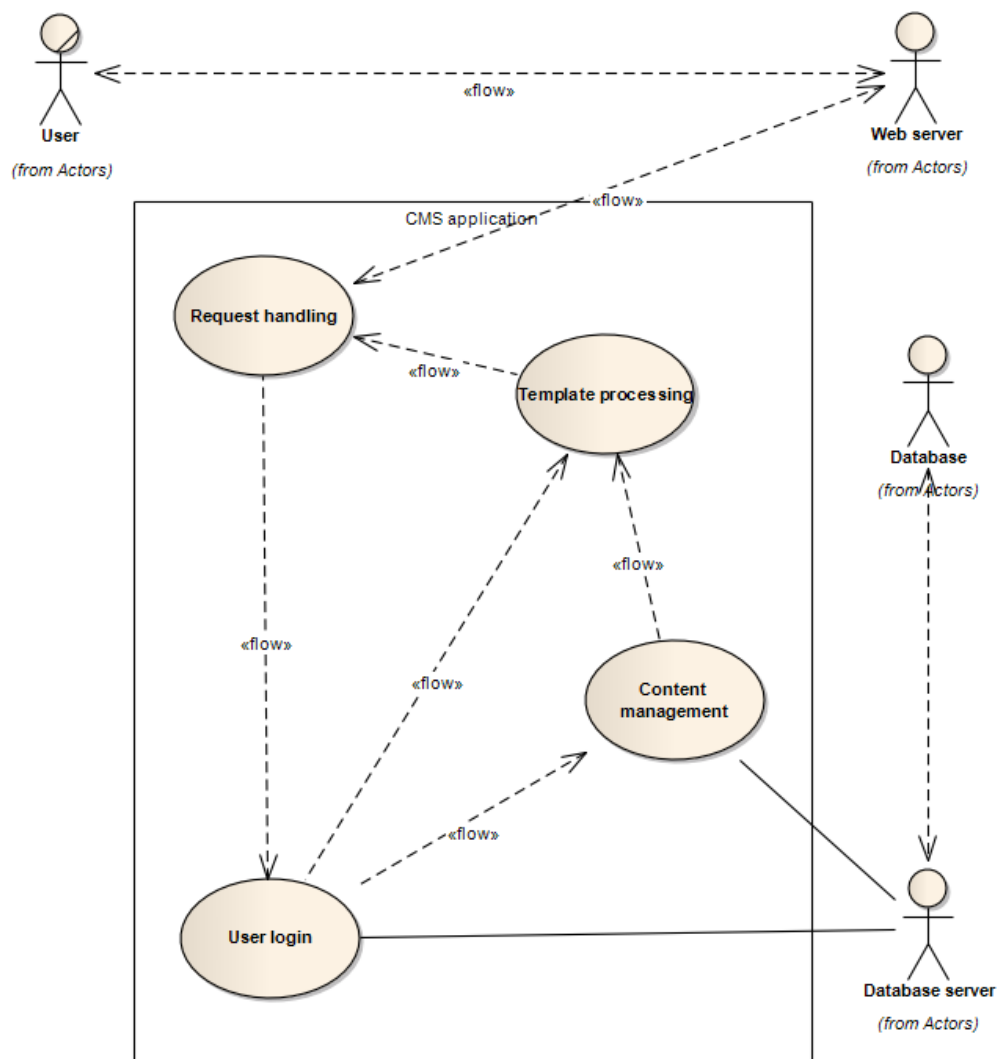


Fig. 1. Manitou sample request processing - use case diagram

9.2 Mod_python

9.3 Class App

9.3.1 Description

Class App is the main application handler class. It inherits from class Request python internal class object. If you create new application based on Manitou framework, you just create new class which inherits from class App.

9.3.2 Method __init__

This is the instance constructor method. It has only one argument default_method with defined default value. It is method name to execute in main application class when method not specified in the request or as script parameter.

9.3.3 Method view

it is general view method on some instance of database class. If parameter class is specified in request it returns new created instance of this class.

9.4 Class Request

9.4.1 Description

Class Request implements Apache request by using mod_python api. It contains methods for request parsing and handling and response generating.

9.4.2 Method request

This method handles the request. It call application method depended on sended parameters and returns response. It creates session class instance which is used to store temporary user values on the server. If application contains requested method, the method is executed. Returned instances are forwarded into template. So it is used MVC architecture.

9.4.3 Method parse_args

This method parse arguments from request structure into hash (dictionary) structure.

9.4.4 Method test

It is only test of functionality of method parse_args. It is used for debugging the method.

9.4.5 Method get_accept_languages

This method returns tuple of browser variable HTTP_ACCEPT_LANGUAGE send in header or the request.

9.4.6 Method replace_diacritics

Method replace_diacritics remove diacritic characters from string from method argument.

9.5 Class MyServer

9.5.1 Description

Class MyServer provides the application based on manitou as stand alone server.

9.6 Class Locale

9.6.1 Description

Class Locale contains methods for application localization. The localization consists of hash functionality with localized strings.

9.6.2 Method read_messages

This method reads csv file separated by semicolon or line character without text delimiter. For each row of the file is created record in the localization hash.

9.6.3 Method get_string

Method get_string returns localized string of hash for selected key and user language.

9.7 Manitou sample usage

i

9.7.1 Source code description

10 CASCADING STYLESHEETS

10.1 Description

Project subdirectory css contains cascading style sheets files. They are split into layout and themes styles.

10.2 Layout styles

There is couple of styles for different page layouts. The main usage of layout styles is logical block positioning on the html pages. There are lot of logical blocks on the main page of the CMS application such as navigation, page text and article text. The position of each logical block of the hml page is described in the layout style. Switching the layout style file causes another position of logical blocks.

10.3 Themes styles

Themes styles affects the graphical view of html pages.

11 SAMPLE APPLICATION - CMS

11.1 Description

CMS sample application is lightweight web application for web pages creation. It contains pages display and pages administration sections. The display section contains two content sections, categorized pages and articles assigned to pages (fig. 1). It has support for content localization of both pages and articles. The administration section provides control of administrators which can enter this section and control of pages and articles contents.

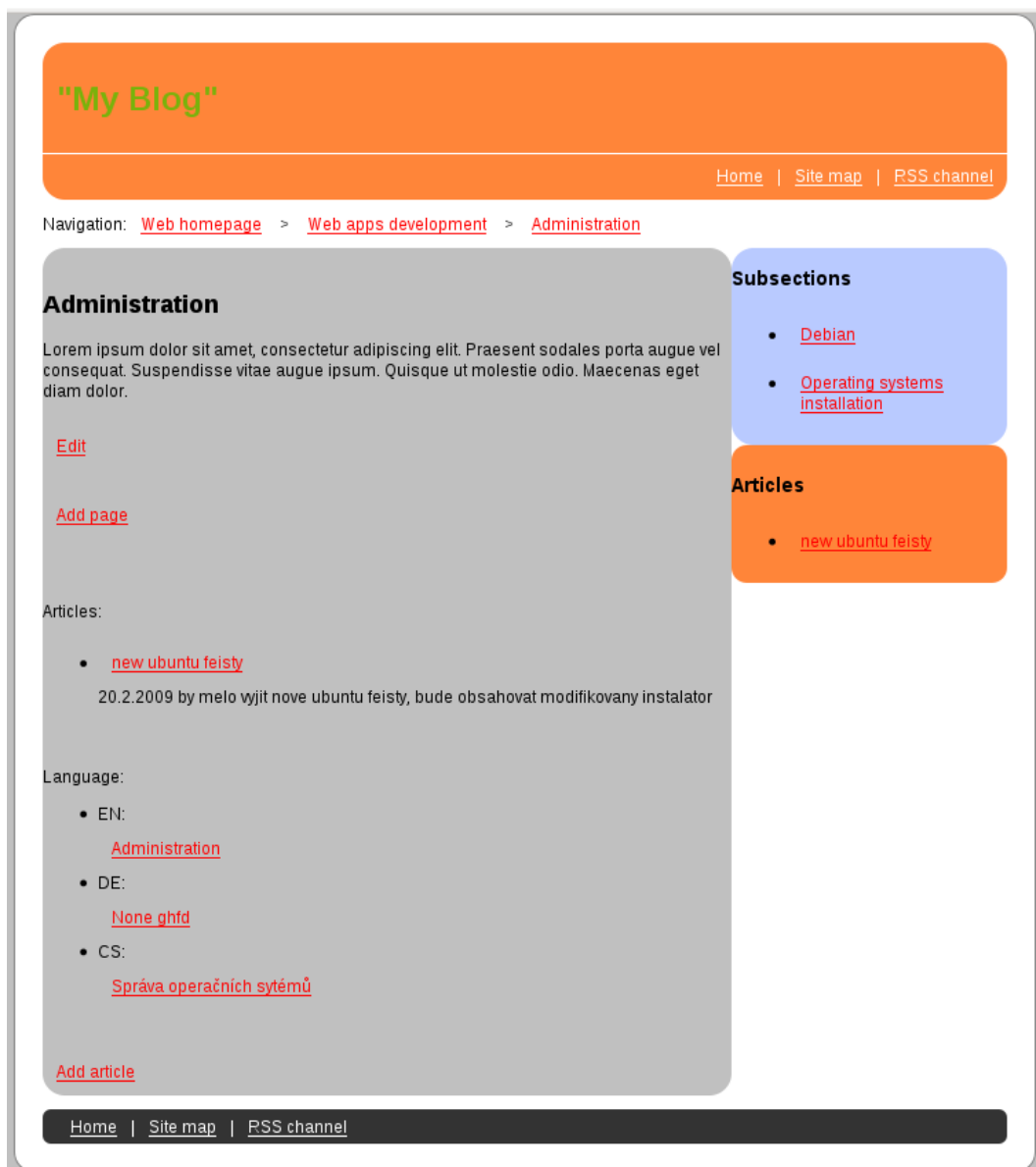


Fig. 1. Layout of the visitor section of the CMS application

11.2 SEO

There is support of Search Engines Optimization(SEO) in the CMS application. The SEO is based on the optimization of request addresses(URL) which are visible to the visitors.

The URL of each page consists from titles of all pages parent to the selected page in the directory tree.

The URL of articles consists from title of the article and titles of all pages in the directory tree parent to the page where the article is assigned to.

11.3 Class Cms

11.3.1 Description

Class Cms is core of the application. It inherits from class App from Manitou part of the framework.

11.3.2 Method `__init__`

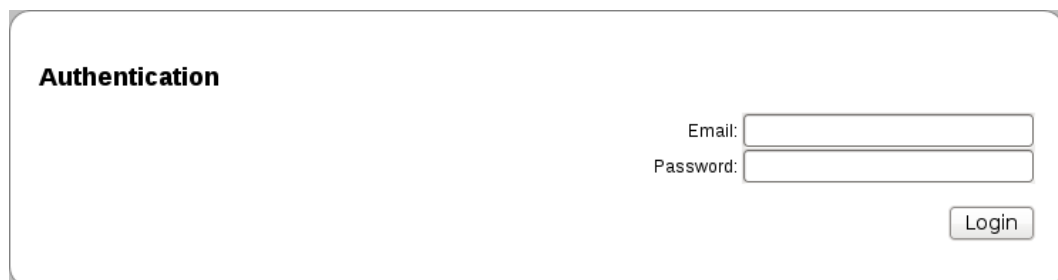
This method is constructor of the class. It calls inherited class constructor. It creates Pyroh library class DB instance.

11.3.3 Method `homepage`

Method `homepage` handles initial page behavior. First we create instances of `pagetext` class for a page selected as homepage. Then we try to recognize language from locale of client browser. We compare recognized language with available `pagetext` and their languages. If match we redirect the request to the selected `pagetext` with appropriate language.

11.3.4 Method `authenticate`

Method `authenticate` handles user login and usage of the non public methods of cms application. It returns True or False depended on success of login or existing session. If exists user session, method returns True. Or if not we check if exists user with the same login and password as entered in the input form (fig. 2). If exists we store the identifier of the selected user into session and return True.



The image shows a web form for user authentication. The form is enclosed in a rounded rectangular border. At the top left, the word "Authentication" is written in bold. On the right side, there are two input fields. The first is labeled "Email:" and the second is labeled "Password:". Below the "Password:" field, there is a button labeled "Login".

Fig. 2. Login form for user authentication in the administration section

11.3.5 Method login

Sense of this method is returning of user login form from appropriate template.

11.3.6 Method logout

This method provides user logout. We destruct identifier of user class in session and we save the session. On response is returned user login form.

11.3.7 Method user

This method returns user instance loaded by user identifier from session.

11.3.8 Method user_search

Method user_search returns class list instance of users if user is authenticated.

11.3.9 Method user_view

If user is authenticated, method returns class user instance loaded by parameters from request.

11.3.10 Method user_edit

Method user_edit returns class user instance loaded by parameters from request and class languages instance if user is authenticated.

11.3.11 Method user_save

Method user_save first creates class user instance loaded by parameters from request. Then all parameters from request are assigned into the instance and its possible autoloader classes instances. Finally the instance is saved.

11.3.12 Method pagetext_view

Method pagetext_view returns class Pagetext instance loaded by request url address. Language of the Pagetext object is assigned into member variable language for next usage in the request.

11.3.13 Method page_view

Method page_view returns class Page instance loaded by parameters from request.

11.3.14 Method page_edit

Method page_view returns class Page instance loaded by parameters from request if user is authenticated.

11.3.15 Method page_save

Method page_save first creates class page instance loaded by parameters from request. Then all parameters from request are assigned into the instance and its possible autoloader classes instances. Finally the instance is saved.

11.3.16 Method article_view

Method article_view returns class Article instance loaded by parameters from request.

11.3.17 Method article_edit

Method article_edit returns class Article instance loaded by parameters from request if user is authenticated.

11.3.18 Method article_save

Method page_save first creates class Article instance loaded by parameters from request. Then all parameters from request are assigned into the instance and its possible autoloader classes instances. Finally the instance is saved.

11.4 Framework usage diagrams

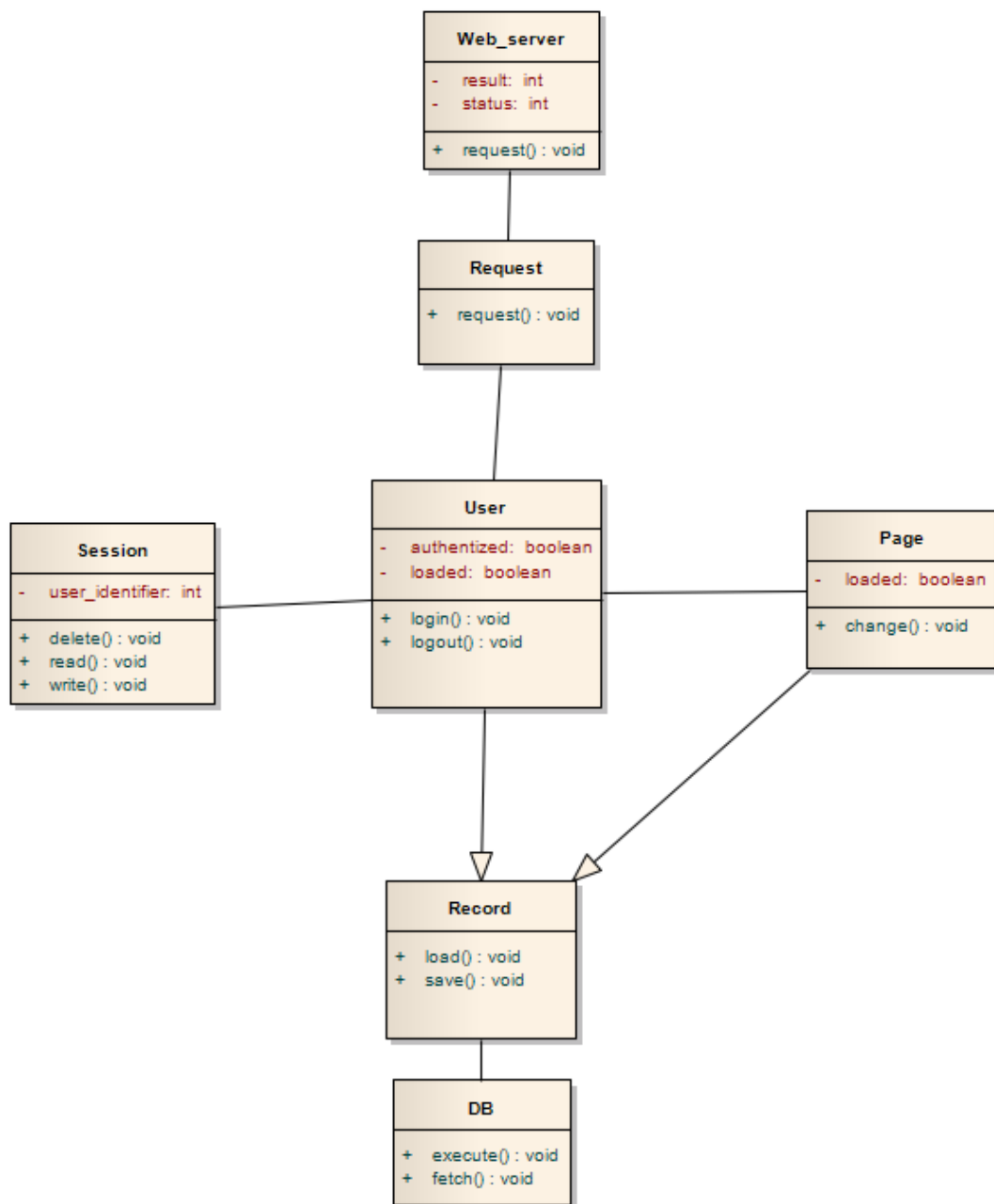


Fig. 3. Class scheme diagram

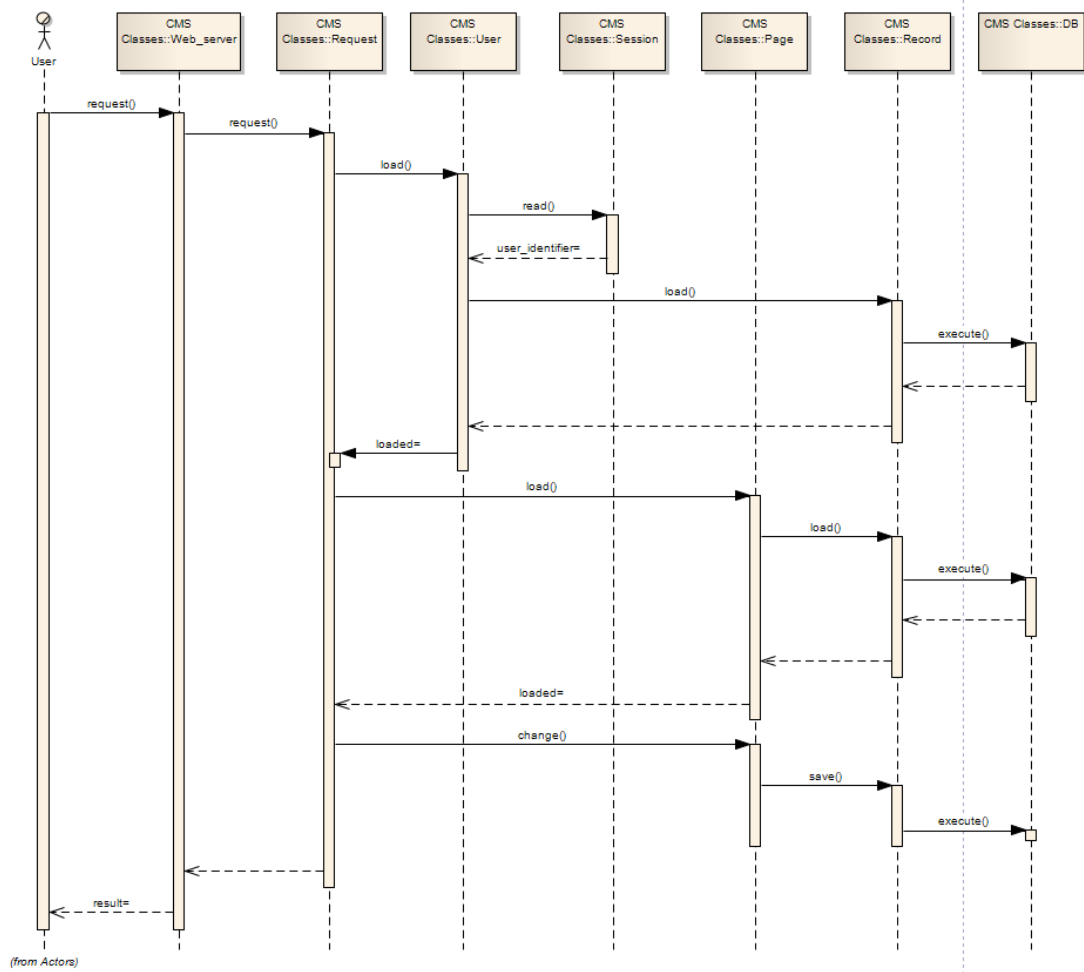


Fig. 4. Page content changing - sequence diagram

11.5 Templates

Templates of application CMS use PSP template engine which is part of the mod_python. There are template files which are processed dependently on the requested application method specified in web request parameters. Appropriate template is processed in the request with data returned by requested method of the application. The data consists from one or more class instances. The application methods are separately described above.

11.6 Configuration

There is configuration section of application running on Apache server. It can be part of server configuration or part of .htaccess file in document root directory. There is sample virtualhost configuration (fig. 5) containing application mapping on location and rewrite rules for SEO url.

Tab. 1. List of required software packages to run CMS application on Debian OS

Package name	Package description
apache2	Apache HTTP Server
apache2-mod-python	Python-embedding module for Apache 2
mysql-server	MySQL database server
python	An interactive high-level object-oriented language
python-mysqldb	A Python interface to MySQL

```
<VirtualHost *>
  ServerName cms.domain.org
  ServerAdmin webmaster@localhost

  DocumentRoot /var/www/kosta/
  Alias /css/ /usr/share/web/css/
  Alias /tinymce2/ /usr/share/tinymce2/www/

  ErrorLog /var/log/apache2/error.log
  CustomLog /var/log/apache2/access.log combined
  LogLevel warn

  ServerSignature On

  <Directory /var/www/kosta/>
    RewriteEngine On

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^{.+[^/]}/$ /$1 [R=301]

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^{.+[^/]}$ /?method=pagetext_view&seo=$1 [QSA]
  </Directory>

  <Location />
    SetHandler python-program
    PythonAutoReload On
    PythonDebug On
    PythonPath "sys.path+['/usr/share/web/']"
    PythonHandler manitou/handler
    PythonOption application "cms.cms.Cms"
    PythonOption database "cms"
  </Location>
  <Location ~ "/(css|tinymce2/)">
    SetHandler none
  </Location>
</VirtualHost>
```

Fig. 5. Apache2 virtualhost configuration

11.7 Database structure

There is text file data.sql containing sql code to be executed to create the CMS database.

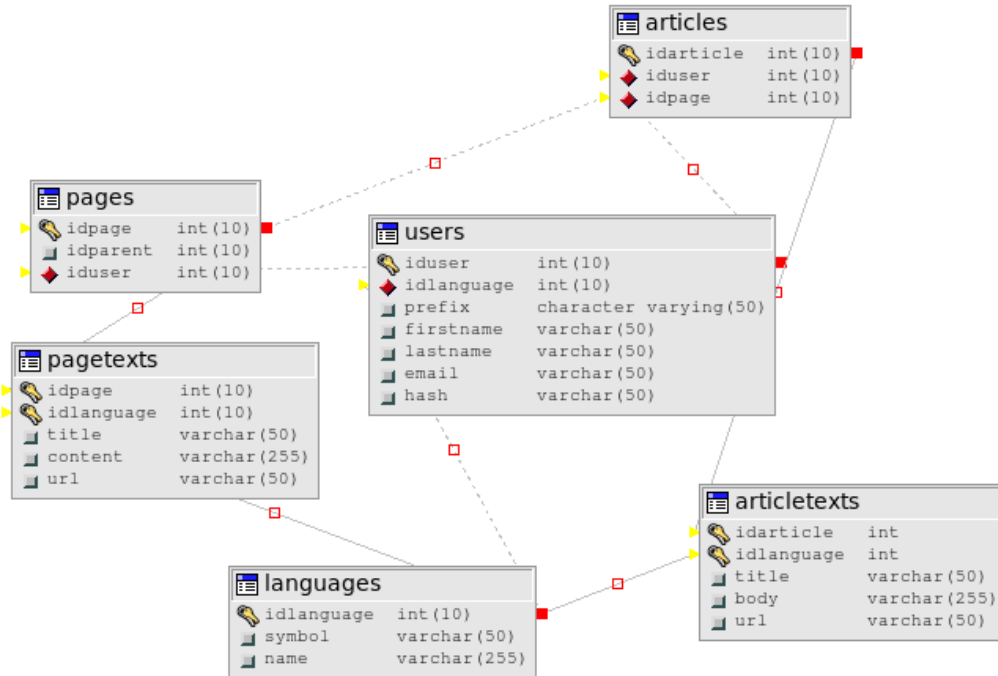


Fig. 6. Database scheme

Tab. 2. Foreign keys definition

TABLE	COLUMN	REFERENCED_TABLE	REFERENCED_COLUMN
articles	idpage	pages	idpage
articles	iduser	users	iduser
articletexts	idarticle	articles	idarticle
articletexts	idlanguage	languages	idlanguage
pages	idparent	pages	idpage
pages	iduser	users	iduser
pagetexts	idpage	pages	idpage
pagetexts	idlanguage	languages	idlanguage
users	idlanguage	languages	idlanguage

11.8 Tables definition

Tab. 3. Table articles definition

Field	Type	Null	Key	Default	Extra
idarticle	int(10) unsigned	NO	PRI	NULL	auto_increment
idpage	int(10) unsigned	NO	MUL	NULL	
iduser	int(10) unsigned	NO	MUL	NULL	

Tab. 4. Table articletexts definition

Field	Type	Null	Key	Default	Extra
idarticle	int(10) unsigned	NO	PRI	NULL	
idlanguage	int(10) unsigned	NO	PRI	NULL	
title	varchar(50)	NO		NULL	
body	varchar(255)	NO		NULL	

Tab. 5. Table languages definition

Field	Type	Null	Key	Default	Extra
idlanguage	int(10) unsigned	NO	PRI	NULL	auto_increment
symbol	varchar(50)	NO		NULL	
name	varchar(255)	NO		NULL	

Tab. 6. Table pages definition

Field	Type	Null	Key	Default	Extra
idpage	int(10) unsigned	NO	PRI	NULL	auto_increment
idparent	int(10) unsigned	YES	MUL	NULL	
iduser	int(10) unsigned	NO	MUL	NULL	

Tab. 7. Table pagetexts definition

Field	Type	Null	Key	Default	Extra
idpage	int(10) unsigned	NO	PRI	NULL	
idlanguage	int(10) unsigned	NO	PRI	NULL	
title	varchar(50)	NO		NULL	
content	varchar(255)	NO		NULL	
url	varchar(50)	YES	UNI	NULL	

Tab. 8. Table users definition

Field	Type	Null	Key	Default	Extra
iduser	int(10) unsigned	NO	PRI	NULL	auto_increment
prefix	varchar(50)	YES		NULL	
firstname	varchar(50)	NO		NULL	
lastname	varchar(50)	NO		NULL	
email	varchar(50)	NO	UNI	NULL	
hash	varchar(50)	NO		NULL	
idlanguage	int(10) unsigned	NO	MUL	NULL	

CONCLUSION

It was developed light-weight framework for web database application creation which consists from database library and web server extension. As a demonstration of its usability there was implemented sample CMS application.

Using this framework by development of application save time. The framework is easy to modify and extend.

There are some suggestions for next development of the framework. It can be added PostgreSQL support into pyroh library to bring multi database support. So class DB should be changed into abstract class implemented by new two classes separated for MySQL and PostgreSQL.

It can be added json-rpc and xml-rpc web services support into Manitou extension. Next feature to be implemented is a support for other templating systems than PSP which is part of Mod_Python Apache module.

BIBLIOGRAPHY

- [1] DiBona C., Stone M., Cooper D. *Open Sources 2.0: The continuing evolution*. O'Reilly, London, 2005. [ISBN 10: 0-596-00802-3].
- [2] Fogel K., *Producing open source software: How to run a successful free software project*. O'Reilly, London, 2005. [ISBN 10: 0-596-00759-0].
- [3] Lutz M., *Learning Python, Third Edition*. O'Reilly & Associates, 2007. [ISBN 10: 0-596-51398-4].
- [4] Lutz M., *Programming Python, Third Edition*. O'Reilly, 2006. [ISBN 10: 0-596-00925-9].
- [5] Martelli A., Martelli Ravenscroft A., Ascher D., *Python cookbook, Second Edition*. O'Reilly, Cambridge, 2005. [ISBN 10: 0-596-00797-3].
- [6] DuBois P., *MySQL Cookbook, Second Edition*. O'Reilly, Cambridge, 2007. [ISBN 10: 0-596-52708-X].
- [7] Bowen R., Coar K., *Apache cookbook, Second Edition*. O'Reilly, Cambridge, 2007. [ISBN 10: 0-596-52994-5].
- [8] Collins-Sussman B., B. w. Fitzpatrick, Pilato C. M. *Version Control with Subversion* O'Reilly, Cambridge, 2004. [ISBN 10: 0-596-00448-6].
- [9] Harrisi D. B., Vyas J. *Debian GNU/Linux 3.1 Bible* Paperback, 2005. [ISBN 13: 978-0-7645-7644-7].
- [10] Musciano C., Kennedy B. *HTML: The Definitive Guide, Third Edition* O'Reilly, London, 2006. [ISBN 10: 1-56592-492-4].
- [11] Meyer E. A. *CSS Pocket Reference, Third Edition* O'Reilly, London, 2007. [ISBN 10: 0-596-51505-7].
- [12] Flanagan D. *JavaScript: The Definitive Guide, Fifth Edition* O'Reilly, London, 2006. [ISBN 10: 0-596-10199-6].
- [13] Worsley J. C., Drake J. D. *Practical PostgreSQL* O'Reilly, London, 2002. [ISBN 10: 1-56592-846-6].
- [14] Daly L. *Next-Generation Web Frameworks in Python* O'Reilly, Cambridge, 2007. [ISBN 10: 0-596-51371-2].
- [15] *The GNU Operating System* [online]. [cit. 2009-04-15]. Available from: <http://www.gnu.org>.

- [16] *Mod_python - Apache/Python integration* [online]. [cit. 2008-01-25]. Available from: <http://www.modpython.org>.
- [17] Dougherty D. *LAMP: The Open Source Web Platform* [online]. [cit. 2001-01-26]. Available from: <http://www.onlamp.com/pub/a/onlamp/2001/01/25/lamp.html>.

LIST OF ABBREVIATIONS

API	Application programming interface
OOP	Object oriented programming
MVC	Model - View - Controller
ORM	Object relation mapping
CMS	Content management system
SEO	Search engine optimization
URL	Uniform Resource Locator
CLI	Command line environment
GUI	Graphical user interface
PSP	Python server pages
PDF	Portable document format
CSS	Cascading style sheets
LAMP	Linux + Apache + MySQL + Perl, PHP or Python
CGI	Common Gateway Interface
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
WWW	World Wide Web
GNU	GNU's Not Unix
GPL	General Public License
PHP	Professional Home Page
SSL	Secure Sockets Layer
TLS	Transport Layer Security
CTL	Tool Command Language
DB	DataBase
DBMS	DataBase Mmanagement System
RDBMS	Relational DataBase Mmanagement System
ORDBMS	Object-Relational DataBase Mmanagement System
BSD	Berkeley Software Distribution
HTML	HyperText Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XUL	XML User interface Language
XML-RPC	remote procedure call protocol which uses XML
JSON	JavaScript Object Notation
JSON-RPC	remote procedure call protocol which uses JSON

SVG	Scalable Vector Graphics
PYROH	Python Relation Object Handler
RSS	Really Simple Syndication (standardized XML file format)
ANSI	American National Standards Institute
ISO	International Organization for Standardization
CSV	Comma separated values

LIST OF FIGURES

Obr. 1. Sample usage of PYROH library	32
Obr. 1. Manitou sample request processing - use case diagram	34
Obr. 1. Layout of the visitor section of the CMS application	38
Obr. 2. Login form for user authentication in the administration section	39
Obr. 3. Class scheme diagram	42
Obr. 4. Page content changing - sequence diagram	43
Obr. 5. Apache2 virtualhost configuration	45
Obr. 6. Database scheme	46

LIST OF TABLES

Tab. 1.	List of required software packages to run CMS application on Debian OS	44
Tab. 2.	Foreign keys definition	46
Tab. 3.	Table articles definition	47
Tab. 4.	Table articletexts definition	47
Tab. 5.	Table languages definition	47
Tab. 6.	Table pages definition	48
Tab. 7.	Table pagetexts definition	48
Tab. 8.	Table users definition	48

APPENDICES

- P I. CMS sample application data structure sql source
- P II. CD Disk with framework, sample application and master thesis

APPENDIX P I. CMS SAMPLE APPLICATION DATA STRUCTURE SQL SOURCE

```
SET STORAGE_ENGINE=INNODB;
DROP database cms;
CREATE database cms DEFAULT CHARACTER SET = utf8;
GRANT ALL ON cms.* TO 'www-data';
USE cms;
CREATE TABLE languages (
  idlanguage INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  symbol VARCHAR(50) NOT NULL,
  name VARCHAR(255) NOT NULL,
  /*state INTEGER(10) UNSIGNED NULL DEFAULT NULL,*/
  PRIMARY KEY(idlanguage)
);
CREATE TABLE users (
  iduser INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  prefix VARCHAR(50),
  firstname VARCHAR(50) NOT NULL,
  lastname VARCHAR(50) NOT NULL,
  email VARCHAR(50) NOT NULL,
  hash VARCHAR(50) NOT NULL,
  idlanguage INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(iduser),
  UNIQUE(email),
  FOREIGN KEY(idlanguage)
  REFERENCES languages(idlanguage)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
);
CREATE TABLE pages (
  idpage INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  idparent INTEGER UNSIGNED NULL DEFAULT NULL,
  iduser INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idpage),
  FOREIGN KEY(idparent)
  REFERENCES pages(idpage)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  FOREIGN KEY(iduser)
  REFERENCES users(iduser)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
);
CREATE TABLE pagetexts (
  idpage INTEGER UNSIGNED NOT NULL,
  idlanguage INTEGER UNSIGNED NOT NULL,
  title VARCHAR(50) NOT NULL,
  content VARCHAR(255) NOT NULL,
  url VARCHAR(50) NOT NULL UNIQUE,
  PRIMARY KEY(idpage, idlanguage),
  FOREIGN KEY(idpage)
  REFERENCES pages(idpage)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  FOREIGN KEY(idlanguage)
  REFERENCES languages(idlanguage)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
);
CREATE TABLE articles (
  idarticle INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  idpage INTEGER UNSIGNED NOT NULL,
  iduser INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idarticle),
  FOREIGN KEY(idpage)
  REFERENCES pages(idpage)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  FOREIGN KEY(iduser)
  REFERENCES users(iduser)
```

```
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
    );
CREATE TABLE articletexts (
    idarticle INTEGER UNSIGNED NOT NULL,
    idlanguage INTEGER UNSIGNED NOT NULL,
    title VARCHAR(50) NOT NULL,
    body VARCHAR(255) NOT NULL,
    url VARCHAR(50) NOT NULL UNIQUE,
    PRIMARY KEY(idarticle,idlanguage),
    FOREIGN KEY(idarticle)
        REFERENCES articles(idarticle)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    FOREIGN KEY(idlanguage)
        REFERENCES languages(idlanguage)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
    );
```

**APPENDIX P II. CD DISK WITH FRAMEWORK, SAMPLE APPLICATION AND
MASTER THESIS**