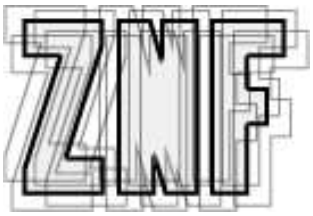


PHP5 MVC framework
for enterprise web applications

http://www.zeronotice.org

version 0.7.10, updated on Wednesday, 14 November 2007, 23:47

ZERONOTICE
INFORMATION TECHNOLOGY SOLUTIONS

Contents

1 Introduction **1**

1.1 Features . . . 1
1.2 Advantages . . . 2
1.3 Philosophy behind . . . 3
1.4 Coding standards . . . 3
1.5 Feedback . . . 4
1.6 Known bugs . . . 4
1.7 Special thanks . . . 4

2 MVC **5**

2.1 MVC for the web-tier . . . 5
2.1.1 Model 1 . . . 6
2.1.2 Model 2 . . . 6
2.1.3 Model 1 or Model 2? . . . 7
2.2 Model-View-Controller and ZNF . . . 7
2.3 Best choice? . . . 8

3 Installation **9**

3.1 Requirements . . . 9
3.2 ZNF installation . . . 9
3.3 Sample application installation . . . 10

4 Usage **11**

4.1 Configuration . . . 11
4.1.1 Application configuration file . . . 11
4.1.2 Module configuration file . . . 13
4.1.3 Database configuration file . . . 17
4.2 Localization . . . 18

4.2.1 Translation file 18
4.3 Presentation 19
4.3.1 Constants 20
4.3.2 Template loading mechanism 20
4.4 Remote actions 21
4.5 File system conventions 23

5 Project **26**
5.1 UML diagrams not up to date 26
5.2 Traceability conventions 27
5.3 Requirements 28
5.4 Use case 28
5.5 Analysis 32
5.6 Realization 33
5.7 Design 35
5.7.1 Class diagram 35
5.7.2 Sequence diagram 37
5.8 Sample application requirements 39

6 Contribute **40**

7 FAQ **42**

8 Credits **43**

A GNU/LGPL License **44**

Bibliography **54**

Chapter 1

Introduction

NOTE: The project is abandoned, read more at http://www.zeronotice.org/.

The goal of this project is to provide an open source framework for building PHP5 enterprise web applications. The core of the ZNF framework is a flexible control layer based on standard technologies like PHP5 and XML. ZNF encourages application architectures based on the Model 2 approach, a variation of the classic Model-View-Controller (MVC) design paradigm.

ZNF provides its own controller component and integrates with other technologies to provide the model and the view. For the model, ZNF can interact with database abstraction layers like PEAR::MDB2 and PDO. For the view, ZNF works well with Smarty template engine and XSL Transformations.

The ZNF framework provides the invisible underpinnings every professional web application needs to survive. ZNF helps you create an extensible development environment for your application, based on published standards and proven design patterns.

The official ZNF web site is http://www.zeronotice.org/.

1.1 Features

- Customizable front controller
- XML configurations to define handling of run-time events
- Authorization

- Input filtering and validation
- Localization
- Themes
- Transparent caching engine
- Remote components
- Model integration with PEAR::MDB2 and PDO
- View integration with Smarty template engine and XSL Transformations
- Capability to run in a E_STRICT environment
- PEAR compatible package

1.2 Advantages

ZNF consists in a robust, modular, secure and well documented implementation of a Model 2 framework. It offers all the advantages coming from proven design patterns on which ZNF is based, especially Model 2 that separates design concerns (data persistence and behaviour, presentation and control), decreasing code duplication, centralizing control and making the application more easily modifiable. Model 2 also helps developers with different skill sets to focus on their core skills and collaborate through clearly defined interfaces.

Some of the basic ideas behind the project are taken from the Apache Struts 1.x framework. In particolar we ported what could have been successful in a PHP5 environment, like the main flow of control and the naming conventions. The rest of the features has been developed keeping in mind the absence of an application server, that means that for each request the data structures are created and destructed.

For example the configurations are is defined in XML files. What happens in Struts is that these files are parsed and, thanks to the application server, kept in memory for all the time of the execution of the application. In PHP this is not possible, but it is also unacceptable that the configuration files are parsed at each request, like happens in many others PHP frameworks. So we made a chaching mechanism that, at the first request, parses and loads the configurations in associative arrays and serializes them to the filesystem. For each successive request, if the XML configuration files are left

untouched, the framework loads and works only with these arrays instead of XML trees, making it extremely fast compared to other solutions.

The implementation take all the advantages of PHP5 new features, especially the new powerful object-oriented Zend Engine 2, and the renewed XML and SOAP handling.

As said ZNF can interact with de-facto standard technologies in the PHP community, like PEAR::MDB2, PDO and Smarty. They are well integrated with the framework but is not mandatory to use them, you can always use your preferred technology for model and view.

Last but not least ZNF has a complete developer's guide and the code is fully commented with PHPDoc syntax, so a full API description is available.

1.3 Philosophy behind

The project has been developed with a lot of criticism. Before it was publicly released, the framework has been rewritten from scratch at least 3 times.

After spending a lot of time studying existing open source frameworks we decided to reimplement a brand new framework from scratch for many reasons. First of all, lot of frameworks are still based on PHP4 and we believe that the Zend Engine 1 is too limited for developing enterprise level web application. At the same time code is often written without a rigorous approach (coding conventions, compliance to W3C standards, output of notice/warnings, lack of documentation/examples). Moreover, lot of projects are shipped with a poor documentation, leaving them very difficult to understand.

The objective of this project is not to provide thousand features, but to provide a discrete set of features well engineered, coded and documented.

1.4 Coding standards

The PHP code of ZNF follows the PEAR coding standards. You can find it in the Chapter 4 of the PEAR Manual, `http://pear.php.net/manual/en/standards.php`. The scripts should not output any notice or warning. Morover, XHTML and CSS code should be fully compliant to W3C XHTML 1.1 and CSS 2.0 standards respectively.

1.5 Feedback

If you found the project interesting and want to contribute do not wait to subscribe and post in the mailing list, you can find all the instructions in the chapter 5.8. The role of the community is very important because is the best way to have feedback with a critical eye. Some of the new features have been added only after some detailed request in our mailing list. Any kind of bug signalling and/or proposals are welcome and encouraged.

1.6 Known bugs

Currently there are no known bugs, but the project is still in beta state so be careful using it in production environments!

1.7 Special thanks

All the Struts development team, for releasing to the community a great framework. Amleto Di Salle and Fabio Mancinelli, for the hours spent with us explaining the Struts structure. Alfonso Pierantonio and Davide Di Ruscio, for giving us ideas and a lot of MVC articles to read. Giuseppe Della Penna, for giving us hints to optimize the code. Henry Muccini, for guiding us in the UML modeling. Sara D'Alia, for giving us the ZNF logo. OpenLUG Linux Users Group of L'Aquila, for supporting our work.

Chapter 2

MVC

Model-View-Controller (MVC) is the recommended architectural design pattern for interactive applications. It organizes them into three separate modules:

- the first for the application model with its data representation and business logic;
- the second for views that provide data presentation and user input;
- the third for a controller to dispatch requests and control flow.

Most web-tier application frameworks use some variation of the MVC design pattern.

The MVC design pattern provides a host of design benefits. MVC separates design concerns (data persistence and behaviour, presentation and control), decreasing code duplication, centralizing control and making the application more easily modifiable. MVC also helps developers with different skill sets to focus on their core skills and collaborate through clearly defined interfaces. For example, a project may include developers of views, application logic and database functionality. An MVC design can centralize the control of such application facilities as security, logging and screen flow. New data sources are easy to add to an MVC application by creating code that adapts the new data source to the existing code. Similarly, new client types are easy to add by adapting the new client type to operate as an MVC view. MVC clearly defines the responsibilities of participating classes, making bugs easier to track down and eliminate.

2.1 MVC for the web-tier

The original version of MVC born in the Xerox laboratories and was meant for the development of GUI applications in Smalltalk environment. The literature on web-tier

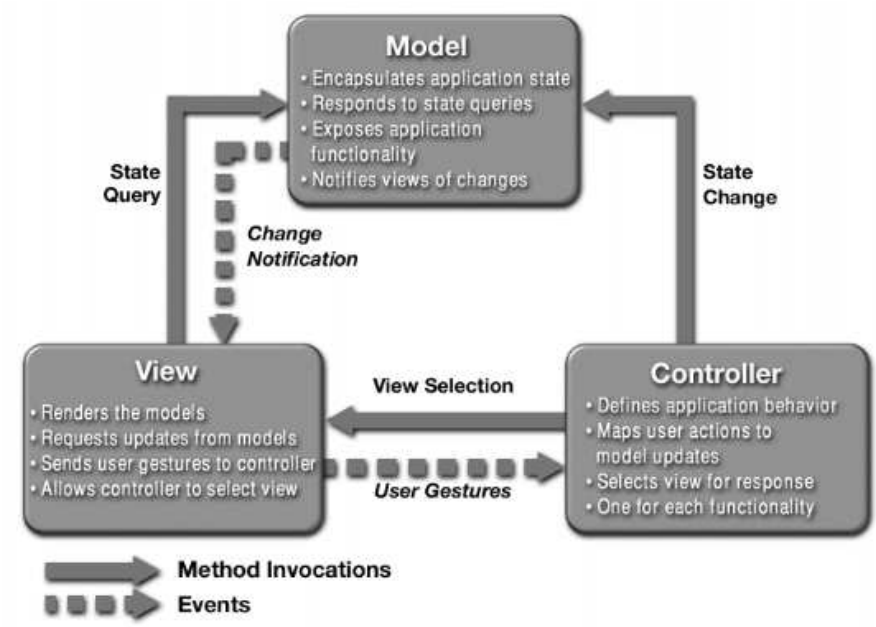


Figure 2.1: The Model View Controller pattern

technology, especially the one of J2EE, frequently uses the terms Model 1 and Model 2 without explanation.

2.1.1 Model 1

Model 1 is the architecture on which, unfortunately according to us, most developers base the design of web applications. Consists of a browser directly accessing web-tier pages, like PHP scripts. The page access data that represent the application model and the next view to display is determined either by hyperlinks selected in the source document or by request parameters. A Model 1 application control is decentralized, because the current page being displayed determines the next page to display. In addition, each page or class processes its own inputs (parameters from GET or POST).

2.1.2 Model 2

Model 2 is the web variant of the MVC pattern, it differs from the original because the view has not the capability to “hear” the model in order to receive notifications of changes in the state. In web applications in fact, views are generated on demand and not in real-time as happens in GUI applications.

A Model 2 architecture differs from Model 1 because introduces a controller class between the browser and the content being delivered. The controller centralizes the logic for dispatching requests to the next view based on the request URI, input parameters and application state. The controller also handles view selection, which decouples pages and classes from one another. Model 2 applications are easier to maintain and extend, because views do not refer to each other directly. The Model 2 controller class provides a single point of control for security and logging and often encapsulates incoming data into a form usable by the back-end MVC model. For these reasons, the Model 2 architecture is recommended for most interactive applications.

An MVC application framework can greatly simplify implementing a Model 2 application. Projects such as ZNF include a configurable front controller class and provide abstract classes that can be extended to handle request dispatches. Some frameworks include macro languages or other tools that simplify application construction.

2.1.3 Model 1 or Model 2?

The Model 1 architecture can provide a more lightweight design for small, static applications. Model 1 architecture is suitable for applications that have very simple page flow, have little need for centralized security control or logging, and change little over time. Model 1 applications can often be refactored to Model 2 when application requirements change.

2.2 Model-View-Controller and ZNF

The ZNF contribution to the controller component of the MVC design pattern is as follows: the main ZNF class handles run-time events in accordance with a set of rules that are provided at deployment time. Those rules are contained in some ZNF configuration files and specify how the system responds to every outcome received from the business logic. Changes to the flow of control require changes only to the configuration files. ZNF also provides a `ZNF_Action_Action` class, which a PHP developer subclasses to create an concrete action. At run-time, the ZNF class is said to "execute actions", which means that the class invokes the execute method of each of the instantiated action classes. The object returned from the execute method directs the ZNF class to what action or PHP page to access next. We recommend that you promote reuse by invoking business logic from the action class rather than including business logic in that class.

The ZNF contribution to the view component of the MVC design pattern is twofold: it provides the `ZNF_Action_ActionForm` class, which a PHP developer subclasses to create a form bean. At run-time, the bean is used in two ways:

- when a PHP view prepares the related HTML form to display, it accesses the bean, which holds values to be placed into the form. Those values are provided from business logic or from previous user input.
- when user input is returned from a browser, the bean validates and holds that input either for use by business logic or, if validation failed, for subsequent re-display.

ZNF also provides the `ZNF_Presentation_Smarty` class, based on Smarty Template engine, and `ZNF_Presentation_XSLT` class, which handles XSL Transformations, that a PHP developer may use for presentation of contents.

The ZNF contribution to the model component of the MVC design pattern is minimal: it only provides the `ZNF_Business_MDB2` and `ZNF_Business_PDO` classes, based on the PEAR::MDB2 and PDO packages respectively, that a PHP developer may use to interact with relational databases.

2.3 Best choice?

Is ZNF the best choice for every project? No. If you need to write a very simple application, with a handful of pages, then you might consider a Model 1 solution. But if you are writing a more complicated application with dozens of pages that need to be maintained over time, then ZNF can help.

Chapter 3

Installation

3.1 Requirements

ZNF framework is operating system independent. It requires an Apache Web Server version 2.0.30 or better with PHP module version 5.1 or better (5.2 recommended). The Apache Web Server installation must have support for `.htaccess` files in order to use additional configuration directives like access restrictions for almost all the subdirectory of the framework. The PHP module has to be compiled with `dom` extension.

The default testing platform is GNU/Linux, so bugs signalling for other platform is encouraged.

3.2 ZNF installation

ZNF is distributed using a PEAR channel, so you need to be familiar with the PEAR package manager (refer to http://pear.php.net/manual/en/installation.cli.php).

To add our channel to your PEAR install, use:

```
pear channel-discover pear.zeronotice.org
```

Then you will be able to install our framework by using:

```
pear install zeronotice/ZNF
```

Note that the `docs` directory of your PEAR install will contain the ZNF documentation and the skeleton to use as a base for your next application.

3.3 Sample application installation

To have a better understanding of the framework, after reading the chapter 3.3 you can install our sample application.

- download news manager package from

  http://prdownloads.sourceforge.net/znf/znf-news-0.7.6.tar.bz2?download

- unpack the package in a directory of the document root of your web server
- initialize a new database (replace `<username>`, `<password>`, `<host>` and `<directory>` with your values.

  – create the database

  ```
  #> mysqladmin create znf
  ```

  – grant privileges to the user

  ```
  mysql> GRANT ALL PRIVILEGES ON znf.* TO
  <username>@localhost IDENTIFIED BY '<password>';
  ```

  – load tables from the DDL file in the `sql` directory of the News and User packages

  ```
  $> mysql -u<username> -p<password> znf <
  <path>/News/sql/News.ddl.sql
  <path>/News/sql/News.dml.sql
  $> mysql -u<username> -p<password> znf <
  <path>/User/sql/User.ddl.sql
  <path>/User/sql/User.dml.sql
  ```

  – put authentication information into the database configuration file, `znf-db-config.xml`, refer to chapter 4.1.3 for details.

  – open the URI `http://<host>/<directory>/` in your browser, ZNF should display the news manager system. To login in the administration area use the `admin/admin` username/password.

Chapter 4

Usage

In this chapter we describe how to use the key features of the framework.

4.1 Configuration

ZNF uses at least two XML configuration files located in the `config` directory: an application configuration file, named `znf-app-config.xml`, and one or more module configuration files, typically named with the syntax `znf-<modulename>-config.xml`.

The application configuration file used to define the application-wise configurations, in particular the default values that the application have to use and the modules that are available. The module configuration file used to initialize the framework resources, like `ZNF_Action_ActionForm` to collect input from the client, `ZNF_Action_ActionMapping` to direct input to server-side actions, and `ZNF_Action_ActionForward` to select output pages. With these configuration files it is possible manage the application without modifying the code of the classes.

If you want to use the model layer classes, ZNF uses another configuration file for the database, named `znf-db-config.xml`, used to connect to the specified relational database.

4.1.1 Application configuration file

The root element of the application configuration file should be `znf-app-config`, possible child elements are `app-settings` and `modules`. For `modules` possible child elements are `module`.

`app-settings` element attributes

- `lang`. String (2-digits), used to specify the default language. The specified language must be configured for all the packages and themes used in the application.
- `langAutodetect`. Boolean value, used to request the language autodetection from the user's browser settings.
- `locale`. String (2-digits + _ + 2-digits), used to specify the default locale.
- `theme`. String, used to specify the default theme. The specified theme must be present in the theme directory.
- `sessionId`. String, used to specify the custom session identifier.

`module` element attributes

- `default`. Boolean value, used to select this module as the default one. Actions belonging to that module can be selected without specifying the module in the URI. The application can have only one default module. The attribute is optional and defaults to `false`.
- `name`. String (without _), used to specify a unique name for the module. The specified name can be used in the URI as a value of the `znfModule` GET variable to do a per-action module selection, or alternatively as a value of the `znfChangeModule` GET variable to do a persistent module selection.
- `processor`. String, used to specify the fully qualified class name of a customized request processor. The specified class must extend `ZNF_Action_RequestProcessor` and will be loaded in place of the default request processor class. It is useful when a customized processing logic flow is needed.

`auth-config` element attributes

- `path`. String, used to specify an action name. The specified action will be invoked in case authentication or authorization restriction are encountered. It is useful to automatcally redirect to a login form.

Application configuration file sample

Here is a sample application configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<znf-app-config xmlns="http://www.zeronotice.org/ZNF/znf-app-config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.zeronotice.org/ZNF/xsd/znf-app-config.xsd">

  <app-settings
    lang="en"
    locale="en_UK"
    langAutodetect="false"
    theme="default"
    sessionId="ZNFID" />

  <modules>

    <module
      default="true"
      name="news"
      config="znf-news-config.xml" />

    <module
      name="user"
      config="znf-user-config.xml" />

    <module
      name="general"
      config="znf-general-config.xml" />

    <module
           name="contents"
      config="znf-contents-config.xml" />

  </modules>

  <wrong-auth-config
    path="user//wrongAuth" />

  <auth-config
    path="user//login" />

</znf-app-config>
```

4.1.2 Module configuration file

The root element of the module configuration file should be `znf-module-config`, `global-forwards`, `action-mappings` and `form-beans`. For `global-forwards` possible child elements are `forwards`, for `action-mappings` are

`action` and for `form-beans` are `form-bean`.

`action` element attributes

- `default`. Boolean value, used to select this action as the default one for the module. The default action will be invoked when no other actions are specified in the URI. The module can have only one default action. The attribute is optional and defaults to `false`.
- `forward`. String, used to specify an application-relative path to a view script or alternatively an action name. In the first case the control-flow will be forwarded to the specified view script instead of loading an action class, in the second case the control-flow will be redirected to the specified action, so the action itself is only used as an alias for the specified action. Input data will be filtered and and validated before the forward. The attributes `forward` and `type` are mutually exclusive and exactly one must be defined.
- `input`. String, used to specify an action name. The control-flow will be returned to the specified action if a validation error is encountered. It is possible to refer to actions of different modules using the sintax `<modulename>//<actionname>`.
- `name`. String, used to specify a form-bean name. The specified form-bean will be used to process and eventually validate input data. The specified name has to match the name of a defined form-bean element. The attribute is optional.
- `nextPath`. String, used to specify an action name. The specified name will be used to automatically update references in the view scripts, so the control-flow is fully configurable in just the configuraion file.
- `path`. String (without _), used to specify a unique name for the action. The specified name has to be used in the URI as a value of the `znfAction` GET variable to do an action selection.
- `parameter`. String, used as a general purpose configuration parameter. The specified parameter can be used to pass extra information to the action class loaded. The value will be ignored by the request processor and simply stored in the `$_REQUEST` superglobal.

- `roles`. String (`,` separated), used to specify a list of security role names. The specified roles will be allowed to invoke this action. The request processor verifies that the user has at least one of the specified roles before to load an action.
- `scope`. String (`request` or `session`), used to specify in which of the `request` or `session` scopes the form-bean has to be placed. The attribute can be defined only if the `name` attribute is present.
- `type`. String, used to specify the fully qualified class name of the action. The specified class must extend `ZNF_Action_Action`. The attributes `forward` and `type` are mutually exclusive and exactly one must be defined.
- `validate`. Boolean value, used to request the call of the `validate()` method of the form-bean specified in the the `name` attribute. The attribute is optional and defaults to `false`.

The forward element maps a logical name to an URI. The application can then perform a forward or redirect, using the logical name rather than the literal URI. This helps to decouple the controller and model logic from the view. The forward element can be defined in action elements.

`forward` element attributes

- `name`. String, used to specify a unique name for the forward. The attribute is required.
- `path`. String, used to specify an URI or action name. In the first case the control-flow will be forwarded or redirected to the specified URI, in the second case the specified action will be invoked. It is possible to refer to actions of different modules using the sintax `<modulename>//<actionname>`. The attribute is required.
- `redirect`. Boolean value, used to request that the request processor performs a redirect instead of a forward. The attribute is optional and defaults to `false`.

`form-bean` element attributes

- `name`. String (without _), used to specify a unique name for the form-bean. This attribute is required.

- `type`. String, used to specify a fully qualified class name. The specified class must extend `ZNF_Action_ActionForm`. The attribute is required.

Module configuration file sample

Here is a sample module configuration for a login work flow:

```
<?xml version="1.0" encoding="utf-8"?>
<znf-module-config xmlns="http://www.zeronotice.org/ZNF/znf-module-config" xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.
    zeronotice.org/ZNF/xsd/znf-module-config.xsd">

  <action-mappings>

    <action
      path="login"
      forward="User/Presentation/Login.php"
      nextPath="loginSubmit"/>
    <action
      path="loginSubmit"
      type="User_Action_Login"
      name="loginForm"
      input="login"
      validate="true">
      <forward
        name="success"
        path="welcome"/>
      <forward
        name="failure"
        path="login"/>
    </action>

  </action-mappings>

  <form-beans>

    <form-bean
      name="loginForm"
      type="User_Action_LoginForm"/>

  </form-beans>

</znf-module-config>
```

In this example there are defined two actions:

- `login` is the first action that simply forwards to the `Login.php` view that displays the login form;

- `loginSubmit` is the second action that make use of `LoginForm` (extends `ZNF_Action_ActionForm`) to check input data, `Login` action (extends

`ZNF_Action_Action`) to control the application flow and interact with the model and finally redirect the flow to success or failure view.

4.1.3 Database configuration file

The root element of the database configuration file should be `znf-db-config`, possible child element is `db`.

`db` element attributes

- `default`. Boolean value, used to select this database as the one used by the application. The application can only have one default database. The attribute is optional and defaults to `false`.
- `dbms`. String, used to select the DBMS type. The attribute is optional and defaults to `mysql`.
- `username`. String, used to specify the database username. The attribute is required.
- `password`. String, used to specify the database password. The attribute is required.
- `hostname`. String, used to specify the database hostname. The attribute is optional and defaults to `localhost`.
- `dbname`. String, used to specify the database name. The attribute is required.

Database configuration file sample

Here is a sample database configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<znf-db-config xmlns="http://www.zeronotice.org/ZNF/znf-db-config" xmlns:xsi="http://
    www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.zeronotice.org
    /ZNF/xsd/znf-db-config.xsd">

  <db
    default="true"
    dbms="mysql"
    username="znf"
    password="znf"
    hostname="localhost"
    dbname="znf"
```

```
11      tableprefix="znf_"/>
12
13  </znf-db-config>
```

4.2 Localization

ZNF has built-in support for localization, it uses XML files to load translated strings. The behaviour is very simple: in the root of the application and in every package you have to create a directory called `lang`, in which you have to create at least one file for the default language, and one file for every other language you want for your application.

The `lang` directory placed in the root should contain translations of applications-wise text, for example the English word "welcome". Contrary, the `lang` directories placed in every package should contain translation of package-specific text, for example the English word "login" in the `User` package.

The language file should be named with the 2 digits abbreviation of the language, for example `en` for English and `it` for Italian.

4.2.1 Translation file

The root element of the XML document should be `znf-translation`, possible child element is `item`.

`item` element attributes

- `code`. Unique alphanumeric string, used to reference this translation item in the application. The attribute is required.
- `text`. Text translated in the language specified.

Note that you can add language files with not all the items translated, the framework will automatically load untranslated text from the default language.

By default the framework is set to `en` for the language and `en_US` for the locale. If you want to change the default value you have to change the values of the `app-settings` attributes in the configuration file.

You can specify if the `ZNF_Presentation_Translation` should load the translations of a specific package or the global ones, typically used in the whole application. The first parameter of the constructor of `ZNF_Presentation_Translation`

let you to specify the package to use. If you pass a string to this parameter the `ZNF_Presentation_Translation` object will load translations files from the `lang` directory of the specified package. If no string is passed to this parameter the `ZNF_Presentation_Translation` object will load translations files from the `lang` directory of the root of the application.

Translation file samples

Here is a sample translation, `en.xml`, for the default English language:

```
<?xml version="1.0" encoding="utf-8"?>
<znf-translation lang="en" xmlns="http://www.zeronotice.org/ZNF/znf-translation"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
    www.zeronotice.org/ZNF/xsd/znf-translation.xsd">

  <item code="welcome" text="Welcome to ZNF"/>
  <item code="contacts" text="Contacts"/>

</znf-translation>
```

Here is the Italian translation, `it.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<znf-translation lang="it" xmlns="http://www.zeronotice.org/ZNF/znf-translation"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
    www.zeronotice.org/ZNF/xsd/znf-translation.xsd">

  <item code="welcome" text="Benvenuto su ZNF"/>
  <item code="contacts" text="Contatti"/>

</znf-translation>
```

4.3 Presentation

ZNF provides the `ZNF_Presentation_Smarty` class, based on Smarty Template engine, and `ZNF_Presentation_XSLT` class, which handles XSL Transformations, that a PHP developer may use for the presentation of contents. Note that all classes which role is to render pages to display to the user must extend the `ZNF_Presentation_Render` abstract class.

You can find the Smarty Template engine documentation at http://smarty.php.net/docs.php and the XSL Transformation documentation at http://www.w3.org/TR/xslt

4.3.1 Constants

Note that the `$znf` variable is reserved in both Smarty and XSL templates. This variable contains the following attributes, that you can use to simplify and speed up the creation of a view:

- `znf.baseHref`. Concatenation of hostname and relative path of the public script directory, used as prefix in hyperlinks in order to refer to correct URIs.
- `znf.actionIndex`. name of the GET attribute used to select the action to execute.
- `znf.moduleIndex`. name of the GET attribute used to select the module configuration.
- `znf.themePath`. Relative path of the current theme, used as prefix in hyperlinks (css, javascripts, images...) in order to refer to correct paths.
- `znf.packagePath`. Relative path of the package in use, used as prefix in hyperlinks (css, javascripts, images...) in order to refer to correct paths.
- `znf.path`. Current action path.
- `znf.nextPath`. Next action path to request.
- `znf.credits`. ZNF credits.

4.3.2 Template loading mechanism

ZNF supports two different kinds of templates:

- package template: associated to the package, shipped with the package to provide a working component ready to use, generally contains markup for the handling and presentation of data;
- theme template: associated to the whole application, used to build the structure and the look of the application.

If you want to change the layout of a package template you can put the new one, with the same name of the original one, in the `themes/packages` directory specifying the name of the package and the language. For example if you want to customize

the `catList.tpl` english template of the News package, you have to put the new `catList.tpl` in the `themes/packages/News/en` directory.

You can specify if `ZNF_Presentation_Render` should load the template of a specific package or the theme ones. The first parameter of the constructor of `ZNF_Presentation_Render` let you to specify the package to use. If you pass a string to this parameter the `ZNF_Presentation_Render` object will load template files from the `template` directory of the specified package. If no string is passed to this parameter the `ZNF_Presentation_Render` object will load template files from the theme of the application.

4.4 Remote actions

ZNF has built-in support for remote actions. This support is still very experimental. Remote action execution using the framework facilities makes the processing flow divided in 2 sub-flows, one for the remote client (ie. the application with which the user interacts using a browser) and one for the remote server (ie. the ZNF application running on a remote host that executes actions as web services).

The framework mask to the developer the complexity of the remote communication, leaving the configuration of the remote action execution really simple. The remote client sends the path corresponding to the action to be executed and the serialized state of the application to a remote server running ZNF and receives a string used to process the forward. The remote server receives the path corresponding to the action to be executed and the serialized state of the application of the remote client running ZNF and sends a string used to process the forward. The communication is made with SOAP protocol.

If you want to try the remote support you have to use the `Remote` package, creating a new module configuration both in client side and server side.

The client have to configure the actions to be executed remotely with the `path` attribute set to the path used in the remote server for the wanted action and the `parameter` attribute set to the URI of the remote host, like `http://<remotehost>/<directory>`.

The server have to configure one action with the `path` set to `webService` and `default` set to `true`. This action used by the customized request processor `Request_Action_RequestProcessorServer` to intercept all the SOAP requests coming from the client.

Remote module configuration file samples

Here is a sample client module configuration for an action that check updates for the framework:

```
<?xml version="1.0" encoding="utf-8"?>
<znf-module-config xmlns="http://www.zeronotice.org/ZNF/znf-module-config" xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.
    zeronotice.org/ZNF/xsd/znf-module-config.xsd">

  <action-mappings>

    <!-- Updates action begin -->
    <action
      default="true"
      path="checkUpdates"
      parameter="http://www.zeronotice.org">
      <forward
        name="available"
        path="Updates/Presentation/UpdatesAvailable.php"/>
      <forward
        name="unavailable"
        path="Updates/Presentation/UpdatesUnavailable.php"/>
    </action>
    <!-- Updates action end -->

  </action-mappings>

</znf-module-config>
```

Here is a sample server module configuration for an action that check updates for the framework:

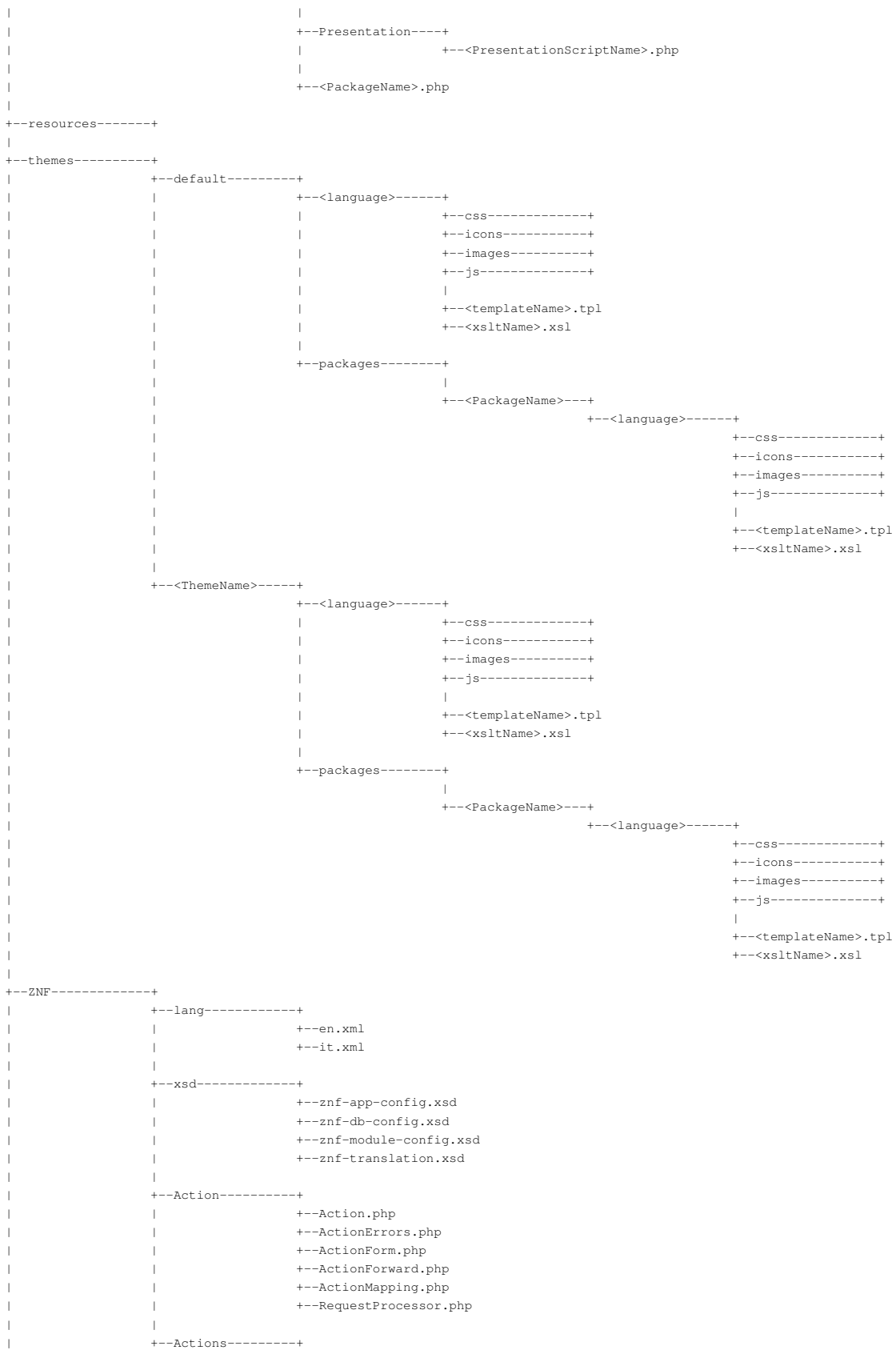
```
<?xml version="1.0" encoding="utf-8"?>
<znf-module-config xmlns="http://www.zeronotice.org/ZNF/znf-module-config" xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.
    zeronotice.org/ZNF/xsd/znf-module-config.xsd">

  <action-mappings>

    <!-- Updates action begin -->
    <action
      default="true"
      path="checkUpdates"
      parameter="http://www.zeronotice.org">
      <forward
        name="available"
        path="Updates/Presentation/UpdatesAvailable.php"/>
      <forward
        name="unavailable"
        path="Updates/Presentation/UpdatesUnavailable.php"/>
    </action>
    <!-- Updates action end -->
```

```
19
20   </action-mappings>
21
22 </znf-module-config>
```

4.5 File system conventions

Last but not least, this a little scheme of the filesystem conventions. Try to adhere always to this conventions if you do not want to have surprises.

```
+--cache-----------+
|
+--config----------+
|                  +--znf-app-config.xml
|                  +--znf-db-config.xml
|                  +--znf-<module>-config.xml
|
+--docs------------+
|                  +--api-------------+
|                  |
|                  +--src-------------+
|                  |                  +--latex-----------+
|                  |                  |
|                  |                  +--uml-------------+
|                  |
|                  +--AUTHORS
|                  +--ChangeLog
|                  +--COPYING
|                  +--README
|                  +--TODO
|                  +--znf-x.x.x.pdf
|                  +--Makefile
|
+--lang------------+
|                  +--<language>.xml
|
+--lib-------------+
|
+--modules---------+
|                  +--<PackageName>---+
|                                     +--docs------------+
|                                     |
|                                     +--lang------------+
|                                     |                  +--<language>.xml
|                                     |
|                                     +--sql-------------+
|                                     |                  +--<PackageName>.ddl.sql
|                                     |                  +--<PackageName>.dml.sql
|                                     |                  +--<PackageName>.drl.sql
|                                     |
|                                     +--templates-------+
|                                     |                  +--<language>------+
|                                     |                                     +--css-------------+
|                                     |                                     +--icons-----------+
|                                     |                                     +--images----------+
|                                     |                                     +--js--------------+
|                                     |                                     |
|                                     |                                     +--<templateName>.tpl
|                                     |                                     +--<xsltName>.xsl
|                                     |
|                                     +--Action----------+
|                                     |                  +--<ActionClassName>.php
|                                     |
|                                     +--Business--------+
|                                     |                  +--<BusinessClassName>.php
```


|
|
+--Presentation----+
|
+--<PresentationScriptName>.php
|
+--<PackageName>.php
|
+--resources-------+
|
+--themes----------+
|
+--default---------+
|
+--<language>------+
|
+--css-------------+
|
+--icons-----------+
|
+--images----------+
|
+--js--------------+
|
|
+--<templateName>.tpl
|
+--<xsltName>.xsl
|
|
+--packages--------+
|
|
+--<PackageName>---+
|
+--<language>------+
|
+--css-------------+
|
+--icons-----------+
|
+--images----------+
|
+--js--------------+
|
|
+--<templateName>.tpl
|
+--<xsltName>.xsl
|
|
+--<ThemeName>-----+
|
+--<language>------+
|
+--css-------------+
|
+--icons-----------+
|
+--images----------+
|
+--js--------------+
|
|
+--<templateName>.tpl
|
+--<xsltName>.xsl
|
|
+--packages--------+
|
|
+--<PackageName>---+
|
+--<language>------+
|
+--css-------------+
|
+--icons-----------+
|
+--images----------+
|
+--js--------------+
|
|
+--<templateName>.tpl
|
+--<xsltName>.xsl
|
+--ZNF-------------+
|
+--lang------------+
|
+--en.xml
|
+--it.xml
|
|
+--xsd-------------+
|
+--znf-app-config.xsd
|
+--znf-db-config.xsd
|
+--znf-module-config.xsd
|
+--znf-translation.xsd
|
|
+--Action----------+
|
+--Action.php
|
+--ActionErrors.php
|
+--ActionForm.php
|
+--ActionForward.php
|
+--ActionMapping.php
|
+--RequestProcessor.php
|
|
+--Actions---------+

```
|                  |                  +--DispatchAction.php
|                  |
|                  +--Business--------+
|                  |                  +--DB.php
|                  |
|                  +--Config----------+
|                  |                  +--AppConfig.php
|                  |                  +--DBConfig.php
|                  |                  +--ModulesConfig.php
|                  |
|                  +--Presentation----+
|                  |                  +--Render.php
|                  |                  +--Smarty.php
|                  |                  +--Template.php
|                  |                  +--Translation.php
|                  |                  +--XSLT.php
|                  |
|                  +--Util------------+
|                  |                  +--Utility.php
|                  |
|                  +--ZNF.php
|
+--autopackage.php
+--index.php
+--Makefile
```

Chapter 5

Project

This chapter is meant for those developers who want to join the development team or simply have a deeper understanding of the framework.

Current generation web applications are becoming very complex, developing this kind of software without a very robust and well-understood process would be foolish. We realized ZNF following some of the best practices described in the RUP methodology, with the only exception that the development team is made by only three people, when in RUP the team is described as a sum of analysts, designers, coders and so on... :)

5.1 UML diagrams not up to date

Before to start describing the project, we need to explain the reasons why in this release we cannot provide an updated version for our UML diagrams.

One of the strongest point of ZNF, compared to other free software projects, was exactly that it has been following the principles of the software engineering. In particular we always delivered lot of UML diagrams attached to the project. For the engineering we used the Community Edition of Poseidon for UML, a CASE tool developed by Gentleware, that started as a fork of the open source ArgoUML project. The Community Edition offered a limited set of functionalities but sufficient to develop our project. For that edition of Poseidon for UML it was possible to automatically retrieve a free licence with a simple registration at the first program start-up, and no fee was required.

Unfortunately, starting from the version 5.0, released in late 2006 the licence model of Gentleware changed radically. Now the Community Edition can be used just paying a "rent subscription" and unfortunately is not possible to retrieve anymore the free

licence with older versions. After some month after the new release (and after some request of clarification from our development team) the company launched an Open Source Sponsorship campaign: they provide free licenses of Poseidon for UML Community Edition for non-commercial open source projects. At first glance it seems a very good idea, but to get the licence you have to agree to some terms. We do not want to absolutely judge these terms, but considering the ZNF purpose the development team do not want to accept them and become a sponsor of Gentleware.

We tried to switch to other CASE tools but unfortunately XMI is still the "standard" less suitable for the interoperability between the tools. XMI v1.0, v 1.1 and v1.2 are native file formats for model storage, but these formats do not specify how to store diagrams. So if you use a tool to open the XMI file, exported with another tool, only the model will be loaded (diagrams and views not).

We did not find yet a CASE tool able to fully import Poseidon for UML project files, so at the time of writing more that one year of design activity can be considered trashed. That is the reason why we will not ship updated diagrams with the v0.7.9 release. The ones shown in this page are still the v0.7.6, and it is likely that they will be left untouched for a long period, considering the big efforts necessary to start up a new design from scratch.

5.2 Traceability conventions

Note that we used a notational convention for the traceability of the elements in the various UML diagrams:

- every requirement is numbered with a number of the form 0.x;
- every use case is numbered with a number of the form 1.x and is followed by a number enclosed in ( ) that represents the requirement to which the use case refer;
- every method of the analysis class diagram is followed by a number enclosed in ( ) that represents the use case to which the method refer;
- traceability from analysis to design model is expressed in a realization diagram.

5.3 Requirements

We started collecting requirements for the whole framework, in order to capture all the constraints that the system must observe.

0.1. The framework is based on the MVC design pattern

- 0.1.1. Application flow is mediated by a central Controller
- 0.1.2. Framework behaviour can be modified by application components

0.2. The Controller portion is focused on receiving requests from the client, typically a user running a web browser

- 0.2.1. Parses a configuration file, which contains a set of mappings that define path match against the request URI
- 0.2.2. Ensures the requesting user has authorization
- 0.2.3. Delegates input data filtering and validation to an input filter
- 0.2.4. Delegates control flow and error handling to a flow controller
- 0.2.5. Delegates presentation of contents to the appropriate View

0.3. The Model represents or encapsulates an application's business logic or state

- 0.3.1. Can be divided into two major subsystems: the internal state of the system and the function that can be executed to change that state
- 0.3.2. The flow controller acts as an adapter between the request and the model
- 0.3.3. Framework architecture provides support for any approach to accessing the model

0.4. The view portion is responsible of producing the next phase of the user interface

- 0.4.1. Provides functionalities to present static or dynamic contents
- 0.4.2. Provides support for internationalization
- 0.4.3. Presents possible errors

5.4 Use case

We captured and expressed detailed system behaviour in 3 use case models, presented in figures 5.1, 5.2 and 5.3.

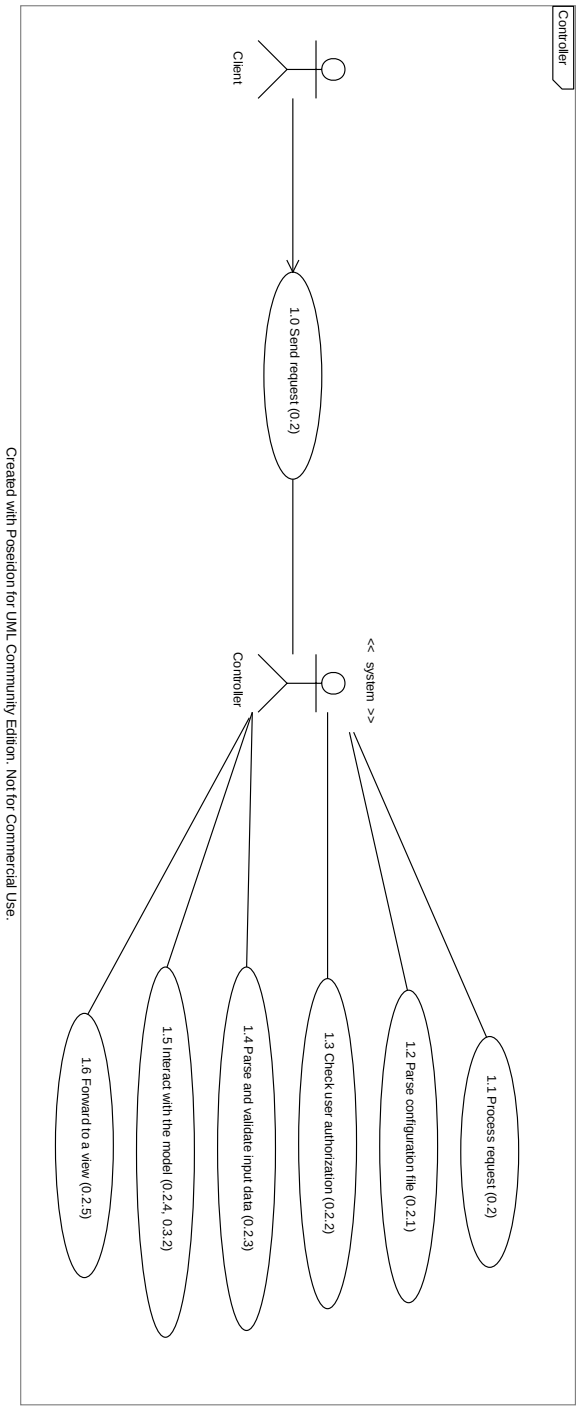


Figure 5.1: UseCase: Controller layer

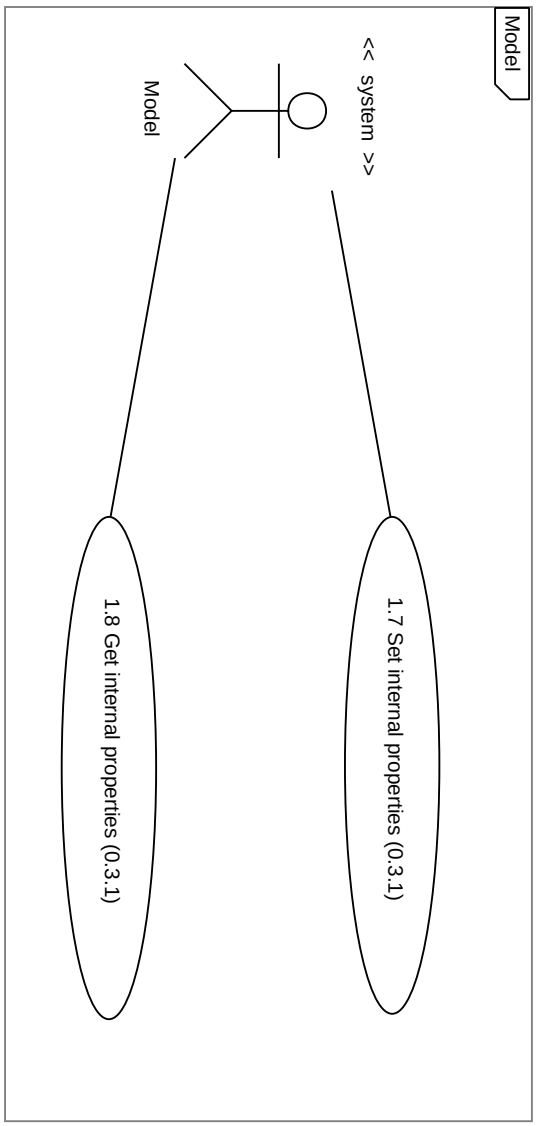


Figure 5.2: UseCase: Model layer

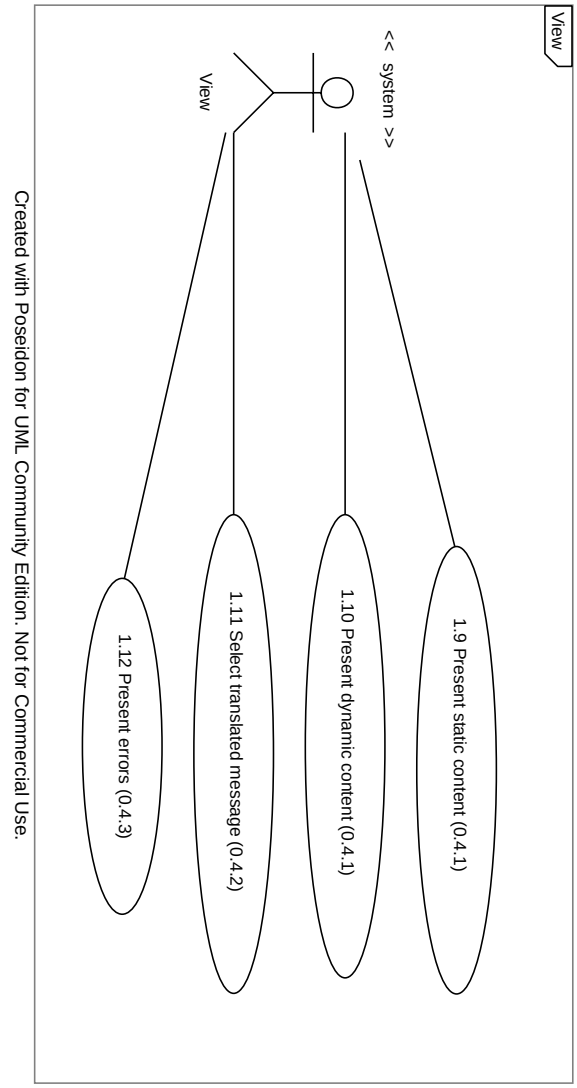


Figure 5.3: UseCase: View layer

5.5 Analysis

Focusing only on the functional requirements and ignoring the architectural constraints of the system we realized a class diagram for the analysis. The class diagram is presented in figure 5.4.

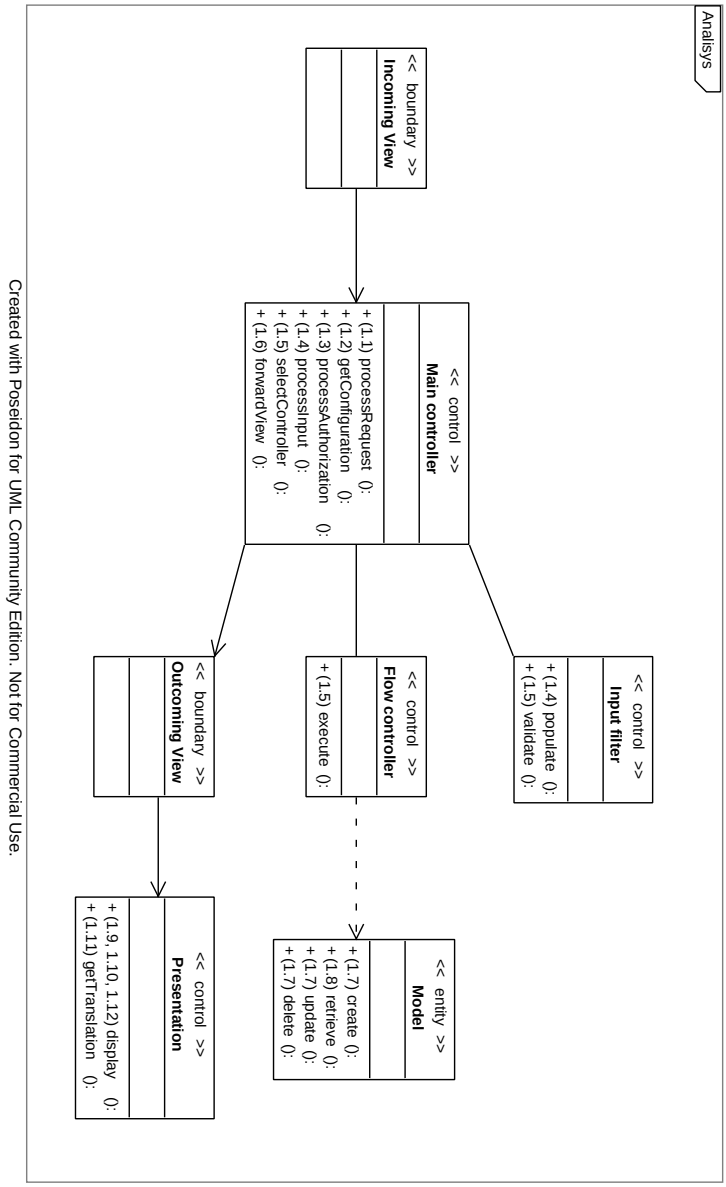


Figure 5.4: Analysis: Class diagram

5.6 Realization

This section outline how we passed from the analysis model to the design one.

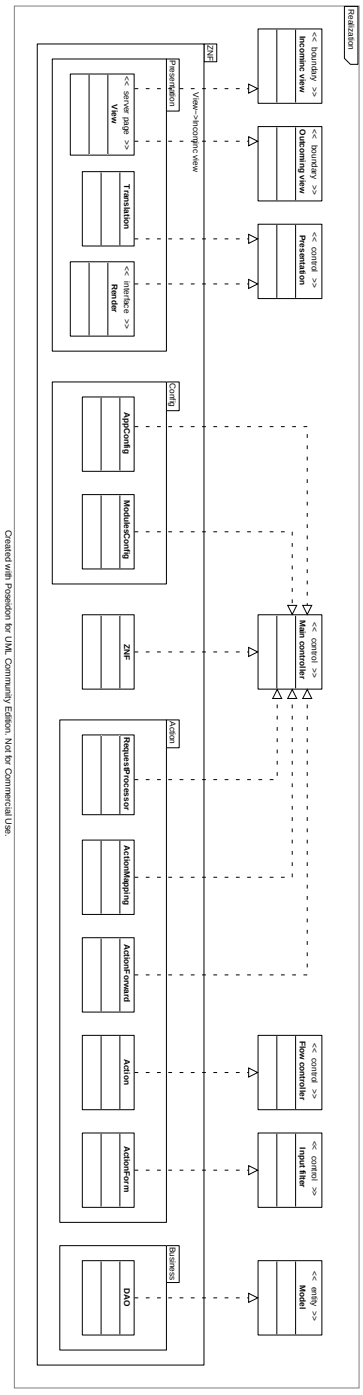


Figure 5.5: From analysis to design: Realization diagram

5.7 Design

Refining the analysis model, such that it can be implemented with components that obey the rules of the architecture, we created a class diagram and a sequence diagram.

5.7.1 Class diagram

The class diagram is presented in figure 5.6.

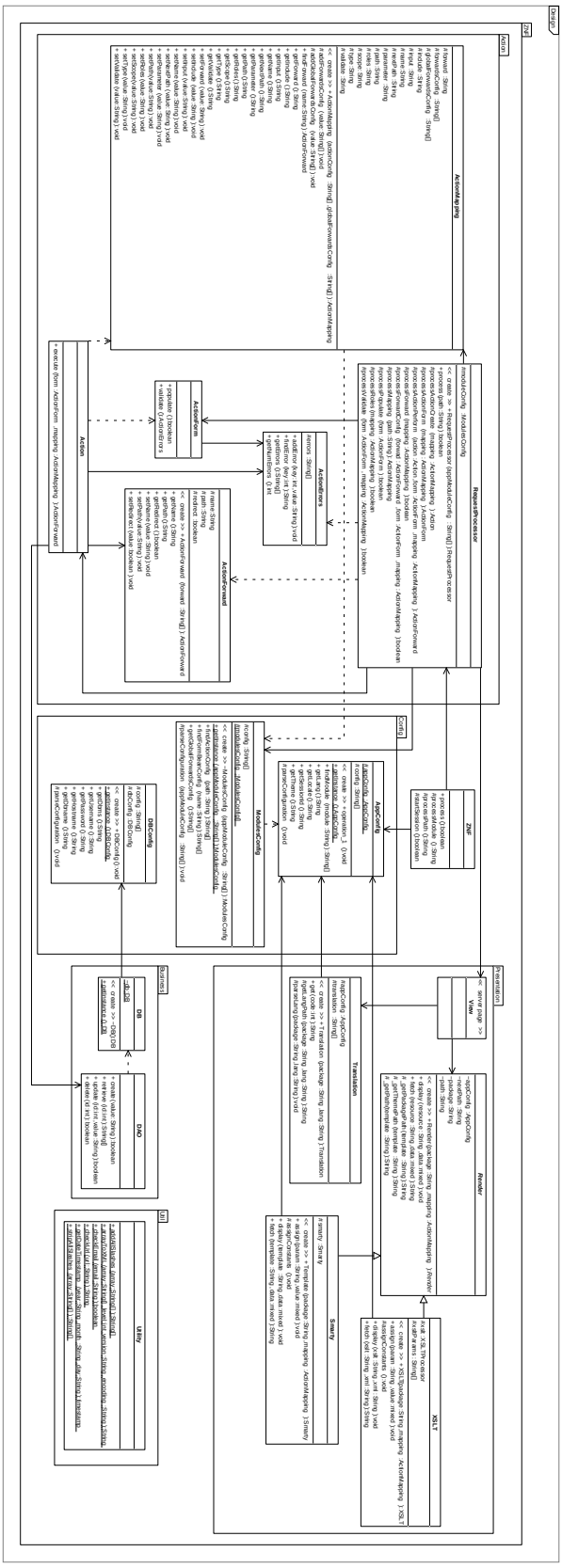


Figure 5.6: Design: Class diagram

5.7.2 Sequence diagram

The sequence diagram is presented in figure 5.7.

Looking at this diagram, especially if you are familiar with UML, you can find atypical that we used a lot of create messages to represent object instantiations, instead of having all object already instantiated, in few words all object aligned at the top of the diagram.

Typically a sequence diagrams represents applications with a well defined life cycle. For example in J2EE a web application is typically deployed in an application container that starts its lifecycle and make it ready to receive requests. In PHP we cannot have a "real" lifecycle for an application, for each request variables/objects are created and destructed, so the resulting sequence diagram is the following.

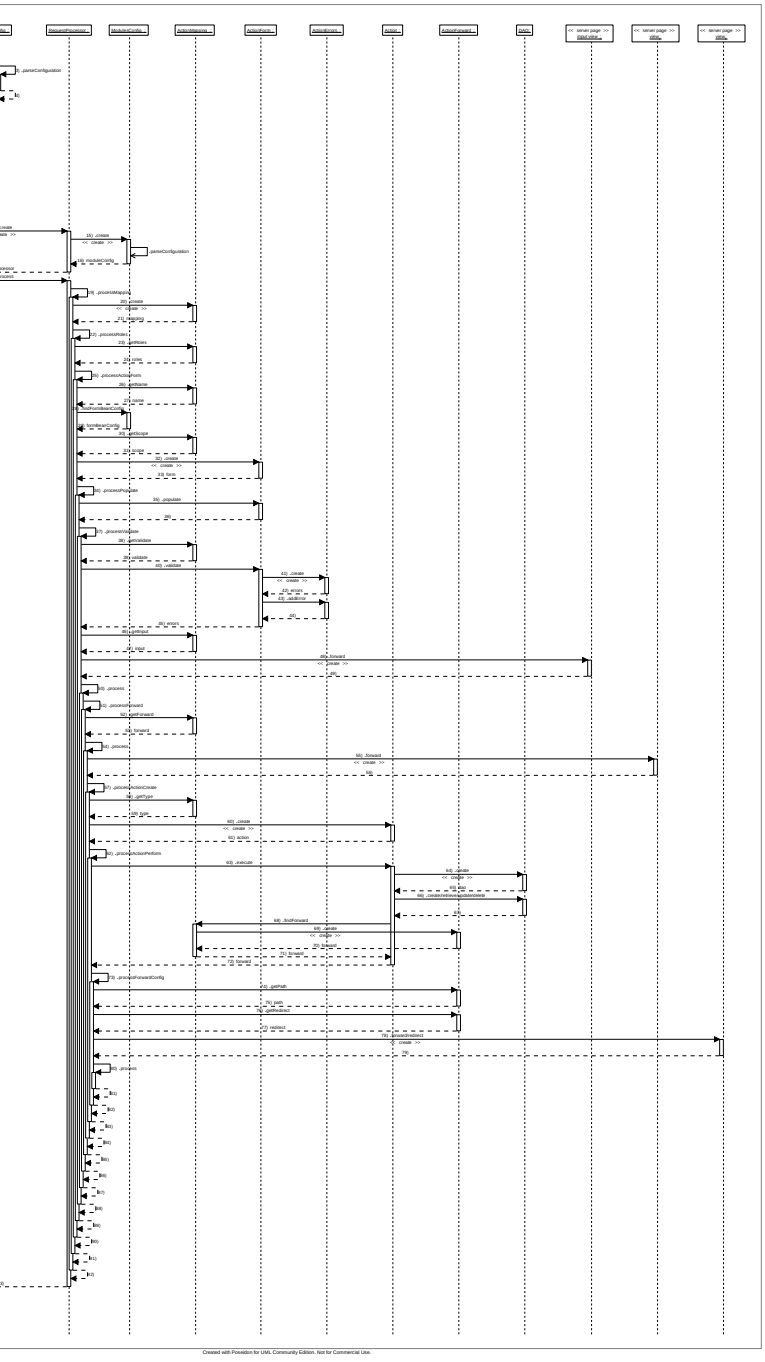


Figure 5.7: Design: Sequence diagram

5.8 Sample application requirements

In order to be complete we also list the requirements that we used for the creation of the news manager system.

0.1. The system provides functions to manage news and categories

0.2. Elements of a category are name, description, image, time and date

0.3. Elements of a news are category, title, summary, body, link, image, views, time and date

0.4. News are grouped by categories

0.5. Each time a news is read the associated views counter must be incremented

0.6. Two kind of actors are supported by the system: guest and administrator

0.7. Guest can list, search and get RSS feeds of the news

0.8. Administrator can also create, retrieve, update and delete news and categories

0.9. During the deletion of a category it is possible to delete all associated news or associate these news to another category

0.10. During updating and deletion of a category it is possible to view the list of categories

0.11. During creation and updating of a news it is possible to view the list of categories

Note that no UML diagrams related to the news manager system are available anymore because the old one were too much old and different compared to the actual implementation, and we did not have time to rebuild them from scratch with a new CASE tool.

Chapter 6

Contribute

ZNF is a free software project, which means that everyone can help improving it. Improving does not always mean writing code. It can also mean reporting bugs, giving feedback, submitting feature requests and even giving financial support.

If you have modified ZNF to expand its functionalities or to fix a bug, you should contribute your changes back to the community. Before creating the patch you must first obtain the latest sources of ZNF from SVN repository by running the follow command command:

```
$> svn checkout https://www.zeronotice.org/svn
```

Now that you have the latest sources you can add your changes. Make sure that your patch is fully compliant to the PEAR coding standards.

After you have finished adding/changing the code test it: we will not accept code that has not been carefully tested. When you are absolutely sure the new code does not introduce bugs, create a unified .diff file which contains your patch.

```
$> svn diff > patch.diff
```

Next step is to submit the patch. Send a mail to `znf-devel@lists.sourceforge.net`. The subject of the mail should be prefixed with "[Patch]" to make it clear you are submitting a patch. Also include a verbose explanation of what the patch does. Do not forget to attach the `.diff` file to the mail.

If you think that you have found a bug in ZNF, please take care that you are using the latest version of ZNF and that your system does meet the requirements. If the bug still persists do not hesitate to fill out a bug report sending a mail to `znf-devel@lists.sourceforge.net`. The subject of the mail should be prefixed with "[Bug]"

to make it clear you are submitting a bug. Also include a verbose explanation of the bug.

Good documentation is essential for users to fully understand any software. Translating documentation is another important task, additional languages are welcome. Help us to improve the internationalization of ZNF sending a mail to `znf-devel@-lists.sourceforge.net`. The subject of the mail should be prefixed with "[Translation]" to make it clear you are submitting a translation.

Chapter 7

FAQ

How does ZNF performs considering the absence of an application server?

Someone could think that ZNF is extremely slow because of the absence of an application server like in the J2EE architecture. This is wrong. We made a chaching mechanism that, at the first request, parses and loads the configurations in associative arrays and serializes them to the filesystem. For each successive request, if the XML configuration files are left untouched, the framework loads and works only with these arrays instead of XML trees, making it extremely fast compared to other solutions.

Is ZNF the best choice for every project?

Not. If you need to write a very simple application, with a handful of pages, then you might consider a Model 1 solution. But, if you are writing a more complicated application, with dozens of pages, that need to be maintained over time, then ZNF can help.

Is it mandatory to use all ZNF classes to develop an application?

Not. For the controller layer it is recommended to extend ZNF controller classes, for the model and view layer ZNF offers helper classes, you can use them or continue using your preferred technologies.

Chapter 8

Credits

Project managers and main developers:

- Alessandro Rossini http://www.alessandrorossini.org
- Graziano Liberati http://www.liberati.org

Main contributors:

- Tomasz Kuter
- Denis A. Konovalyenko

Contributors:

- Guéric Folliot
- Valentin David
- Markus Wigge
- Christian Kassab
- Emad B.

For questions related to the framework (installation problems, bug reports, feature requests...) please refer to our mailing lists, do not mail the authors directly

- General discussions mailing list: znf-general@lists.sourceforge.net
- Developer's mailing list: znf-devel@lists.sourceforge.net

Appendix A

GNU/LGPL License

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and

use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the “Lesser” General Public License because it does Less to protect the user’s freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure

that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable

source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you

must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of

the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision

will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

¡one line to give the library's name and a brief idea of what it does.¿ Copyright (C) ¡year¿ ¡name of author¿

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

¡signature of Ty Coon¿, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

Bibliography

[1] PHP http://www.php.net.

[2] PEAR http://pear.php.net.

[3] Smarty Template engine http://smarty.php.net.

[4] XML http://www.w3.org/XML/.

[5] XSLT http://www.w3.org/TR/xslt.

[6] Apache Struts http://struts.apache.org/.

[7] Sun Java BluePrints http://java.sun.com/blueprints/.

[8] MySQL http://www.mysql.com.