



**BOZA**

**Web & Software Solutions &  
Consultancy**

**BOZA CODE GENERATION SOLUTION  
USING  
THE ZEND FRAMEWORK**

**Documentation version 2.1**

Authors: Aram Hovsepyan, Bart Schelfhout

{aram, bart}@bozasolutions.com

<http://www.bozasolutions.com>

# Table of Contents

<b>Table of Contents .....</b>	<b>2</b>
<b>1 Introduction .....</b>	<b>3</b>
1.1 “Model once, generate everywhere” .....	3
1.2 Eclipse .....	3
<b>2 Installation .....</b>	<b>4</b>
2.1 Pre-Packaged Installation.....	4
2.2 Installing BOZA Framework as an Eclipse plug-in.....	4
2.2.1 <i>Required plugins</i> .....	4
2.2.2 <i>BOZA Framework plugin</i> .....	7
2.2.3 <i>Optional plugins</i> .....	7
2.2.4 <i>Installation Check</i> .....	7
<b>3 Use .....</b>	<b>9</b>
3.1 Process.....	9
3.2 Detailed Design aka Modeling .....	9
3.2.1 <i>UML model</i> .....	9
3.2.2 <i>Data types</i> .....	12
3.2.3 <i>Associations as attributes</i> .....	12
3.2.4 <i>Alternative Modeling Tool: <a href="#">MagicDraw</a></i> .....	13
3.3 Code Generation: “just click a button” .....	13
3.3.1 <i>Click a button</i> .....	13
3.3.2 <i>Generated structure</i> .....	14
3.3.3 <i>Manually modified code: protected code sections</i> .....	16
3.4 Generator framework customization.....	17
<b>4 Common Problems and FAQ .....</b>	<b>18</b>
4.1 The BOZA PHP Generator menu does not appear.....	18
4.2 I click on Generate... and I get an error message: «The chosen operation is currently not available» 18	18
4.3 Can I use a database different from MySQL? Can I use a different structuring of my folders? Can I have some extra helper libraries? Can I do....	18

## 1 Introduction

The document contains a short description on how to install and use the BOZA code generator for the Zend Framework. The generator provided is completely free of charge for both personal and commercial use.

### 1.1 “Model once, generate everywhere”

**Model-driven development (MDD)** is a software development methodology that focuses on creating models, or abstractions, more close to some particular domain concepts rather than computing (or algorithmic) concepts. It is meant to increase productivity by maximizing compatibility between systems, simplifying the process of design, and promoting communication between individuals and teams working on the system. In a MDD paradigm models are more than mere documentation artifacts. Models are first class development entities and are used to generate semi-automatically the system in development.

We have developed our own MDD framework and the presented code generator is part of this framework.

### 1.2 Eclipse

Eclipse is an open source project that provides a universal toolset for development. The Eclipse project features numerous different plugins including a php plugin, web developer tools plugin, etc. Using an appropriate configuration one can obtain a very powerful open source IDE. This is one of the reasons we have decided to implement the BOZA code generator as an Eclipse plugin rather than a standalone tool.

The next section will present a detailed overview on how to install the BOZA code generation framework.

## 2 Installation

This section outlines the installation process of Eclipse and the BOZA code generator.

### 2.1 Pre-Packaged Installation

One way to install the BOZA Framework is by downloading and unzipping a pre-packaged Eclipse that already contains the framework. In order to do so please visit our website on [www.bozasolutions.com](http://www.bozasolutions.com) and download the standalone framework. Unzip it in the location you like (e.g. C:/Program Files) and run the eclipse.exe file. As you can notice it comes with a sample project called myWebApplication that contains a sample UML model of a simple web application.

You may further use the Help->Check for updates function (see Figure 1) in order to obtain the latest versions of all available plugins:

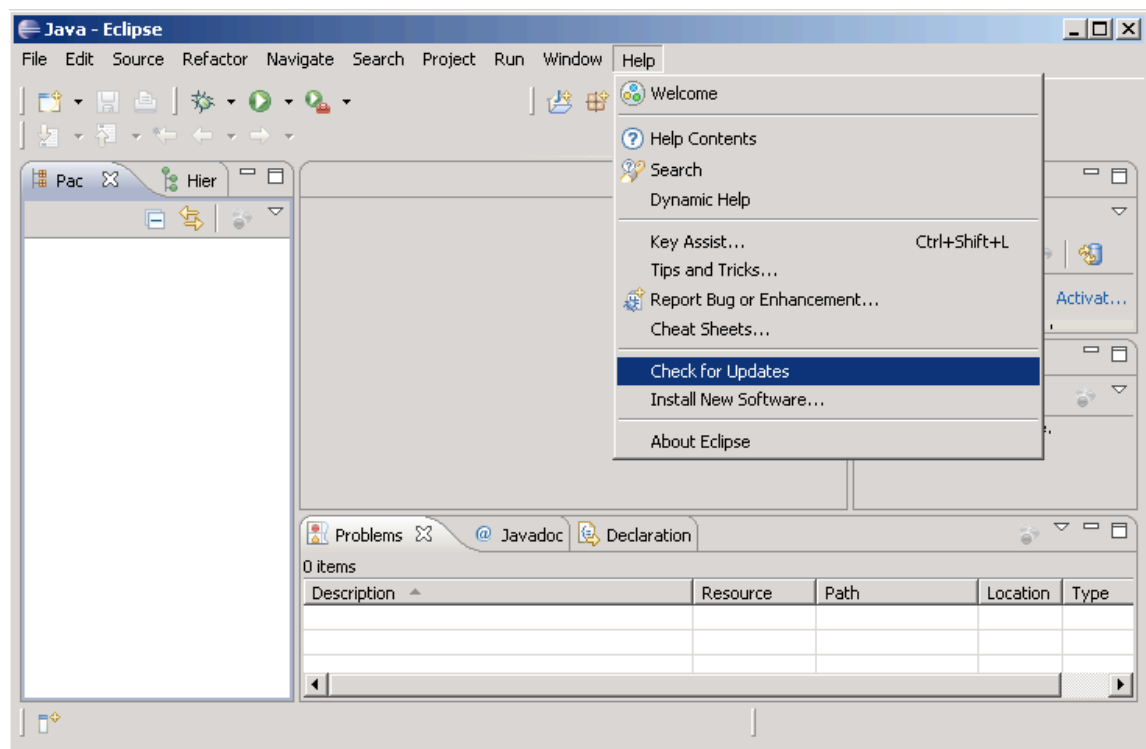


Figure 1

### 2.2 Installing BOZA Framework as an Eclipse plug-in

If you already have a running Eclipse or you are using an OSX or Linux, you can install BOZA Framework as a plugin.

#### 2.2.1 Required plugins

There are a number of required plugins. Please make sure you have them installed before proceeding with the next section. The required plugins are as follows:

- Eclipse Modeling Framework (EMF)
- Unified Modeling Language (UML) and Incubation tools
- MOFScript

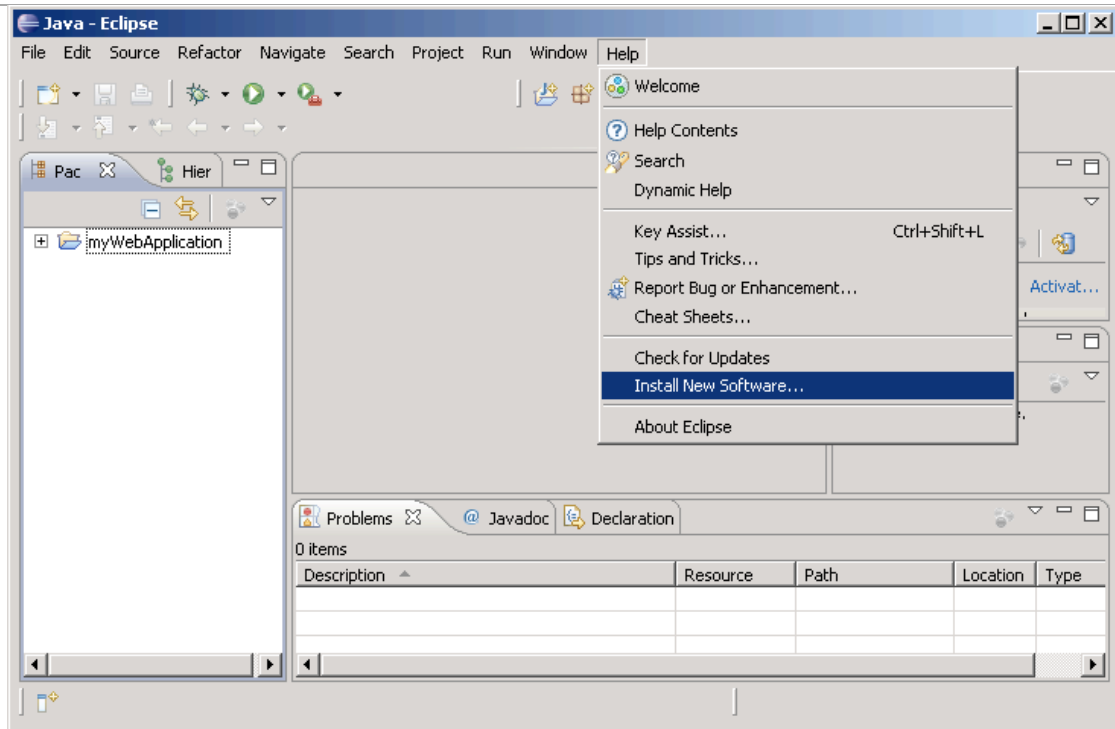
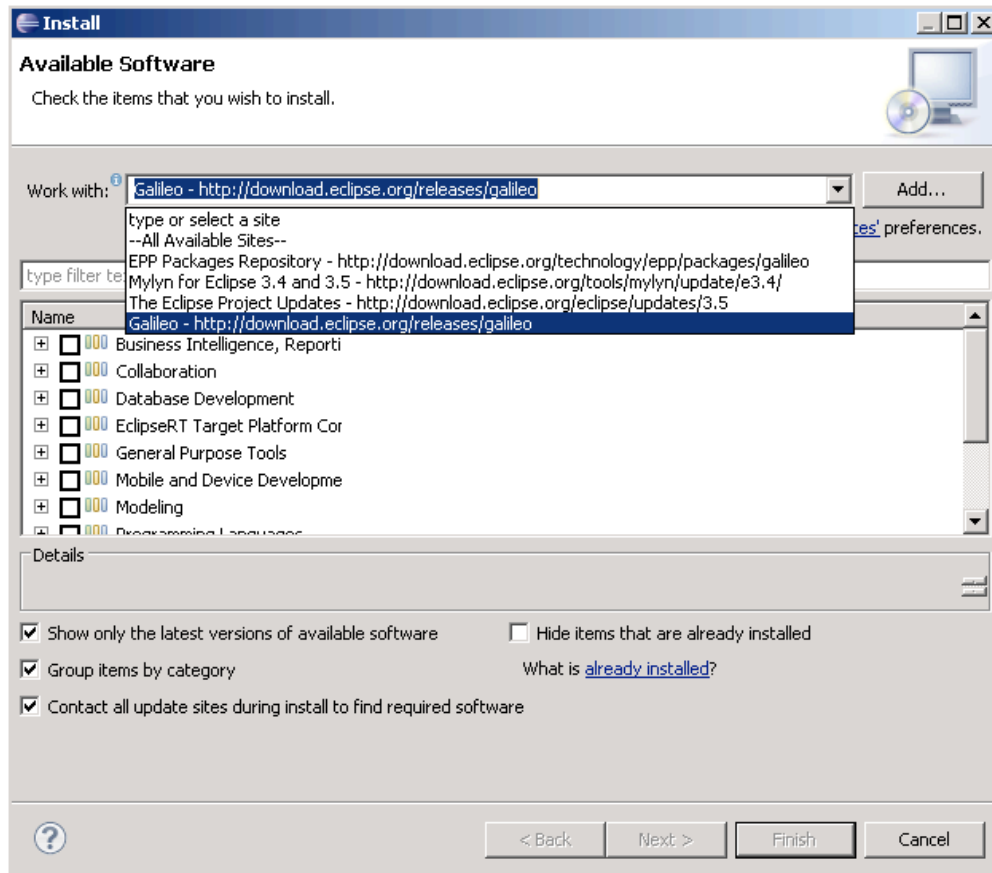


Figure 2

Then select from the available repositories the Galileo repository (see Figure 3).



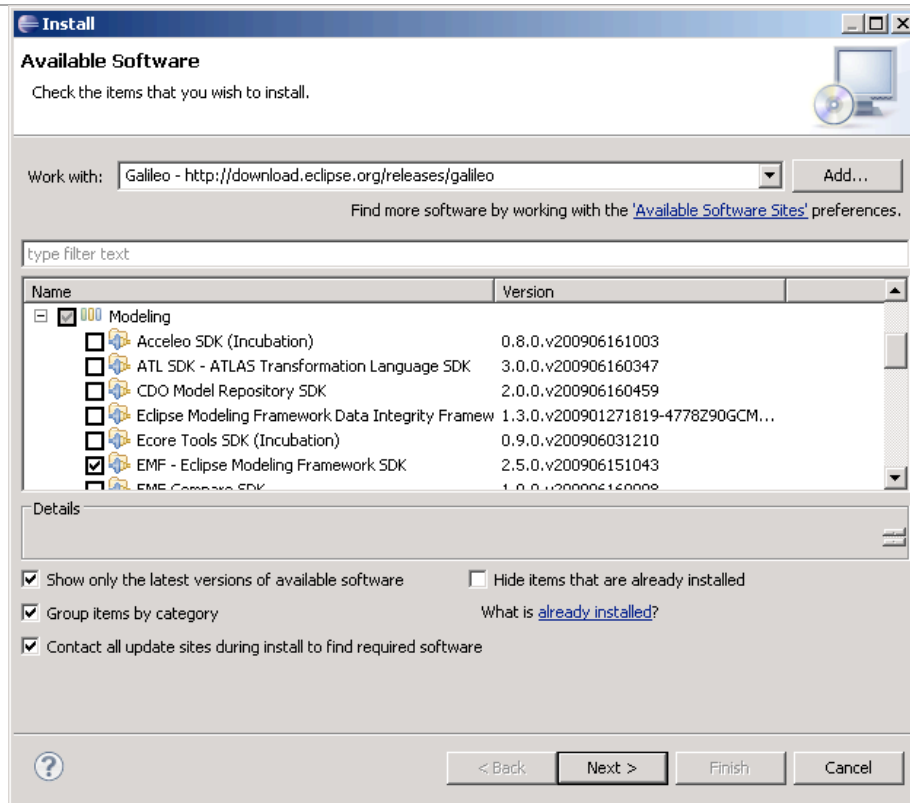


Figure 4

### 2.2.1.2 Installing MOFScript

In order to install MOFScript you need to follow a similar procedure, however first you will need to add the MOFScript repository to the available repositories. Go to Help->Install New Software... again and click on Add... Fill in the Name (arbitrary) and the Location (<http://download.eclipse.org/modeling/gmt/mofscript/update/>) of the MOFScript repository (see Figure 5). Proceed by selecting the newly added repository and installing MOFScript.

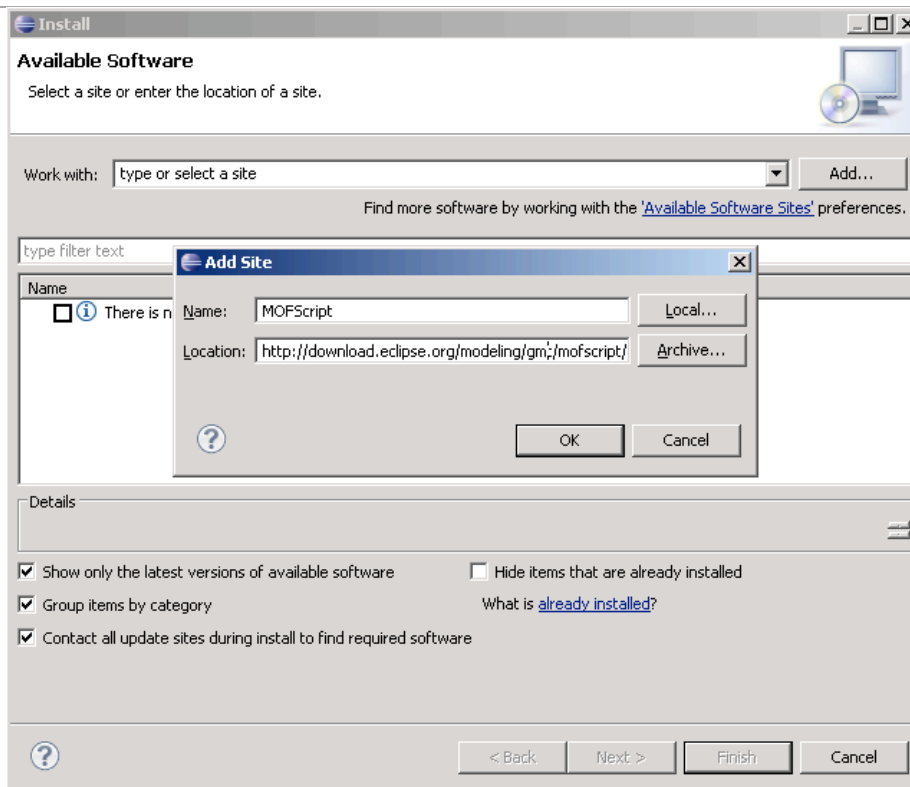


Figure 5

### 2.2.2 BOZA Framework plugin

In order to do so use Help->Install New Software... then click on Add... and add the BOZA Framework repository site as follows:

Name: BOZA Framework

Location: <http://www.bozasolutions.com/generator/update>

Proceed by installing the latest version of the BOZA Framework (this step is similar to the installation of other plugins).

Restart Eclipse when prompted.

### 2.2.3 Optional plugins

A number of Eclipse plugins are not required in order to run the code generator, however can be very useful if you plan on using Eclipse for the complete development of online applications.

#### Eclipse PHP Development Tools Project

There are a number of PHP plugins for Eclipse, however in our view [PDT](#) is one of the most mature ones (Zend Studio is actually based on it). Please consult the [PDT](#) website for more information on PDT and its installation.

#### Eclipse Web Tools Platform Project

The WTP Project is a handy plugin for all kinds of syntax highlighting for JavaScript, CSS and HTML documents. Please consult the [WTP](#) website for more information and installation.

### 2.2.4 Installation Check

In order to check for a correctly installed Eclipse with the generator plugin you should see the BOZA Generator menu item (BOZA PHP Generator->Generate...) on top of your Eclipse (see Figure 6).

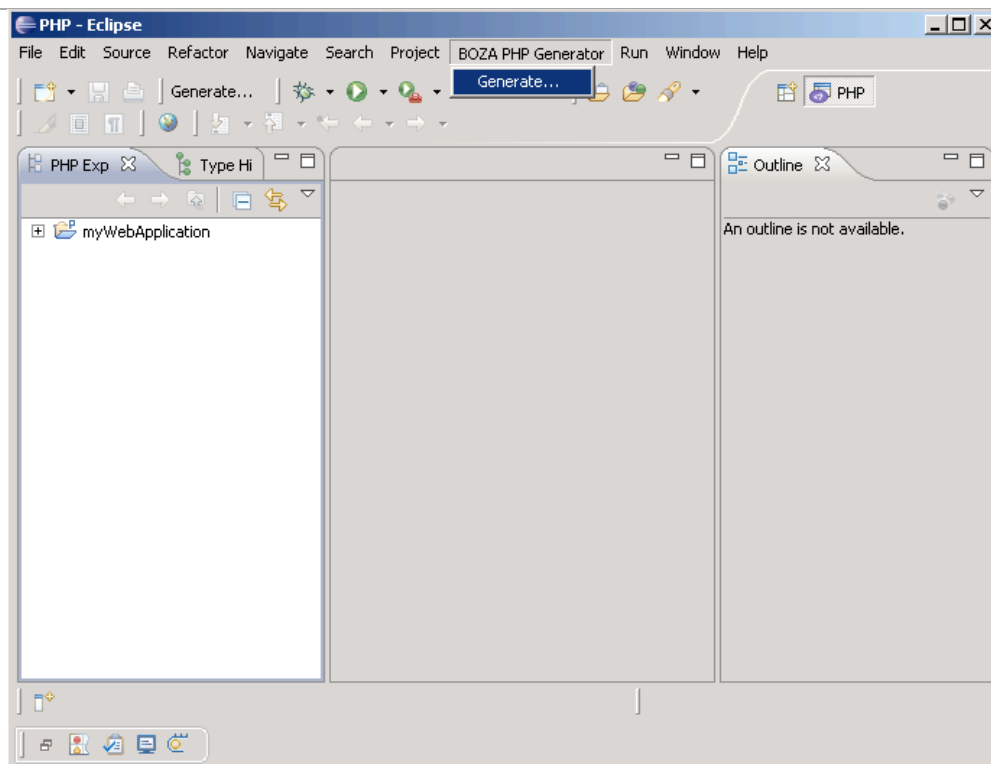


Figure 6

## 3 Use

In this section we provide a detailed description how to use the generator once it has been successfully installed as an Eclipse plugin.

### 3.1 Process

Figure 7 shows a sample development process that could be used in a web-application development. Needless to mention that this process is iterative. Only the Detailed application design and Partial code generation processes are of interest in this document. In the next subsections we will illustrate what each of these activities represents.



Figure 7

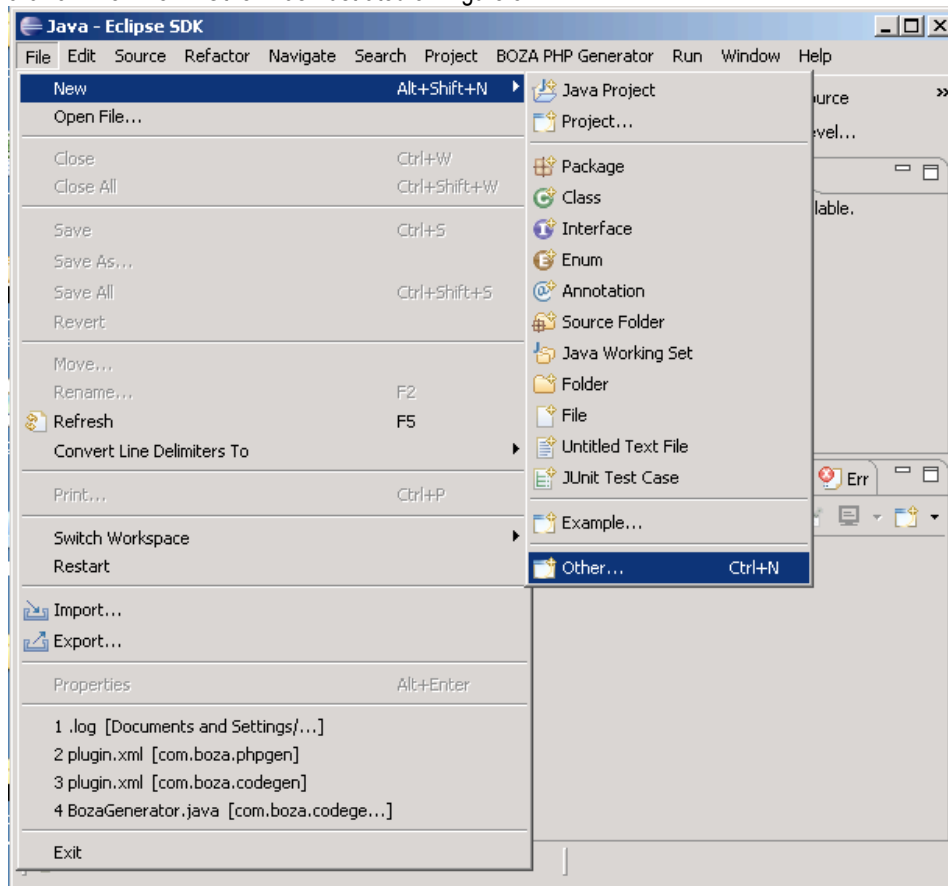
### 3.2 Detailed Design aka Modeling

This section provides a quick description on creating a UML structural model for your project. This model is then feeded to the generator that will generate partial source code.

#### 3.2.1 UML model

BOZA framework makes use of standard UML 2.1 advocated by the OMG. We will be using only the structural part of the UML standard, hence the first step would be creating a structural model of the application you are developing. It is out of the scope of this document to discuss how to create (good) UML structural models.

Before creating the UML model of the application we need to setup a new Eclipse project that will hold all the resources. In order to do so startup Eclipse, click on File->New->Other... as illustrated on Figure 8.



Then select a General->Project (if you have PDT plugin installed you may as well select a new PHP project) and click on Next (Figure 9).

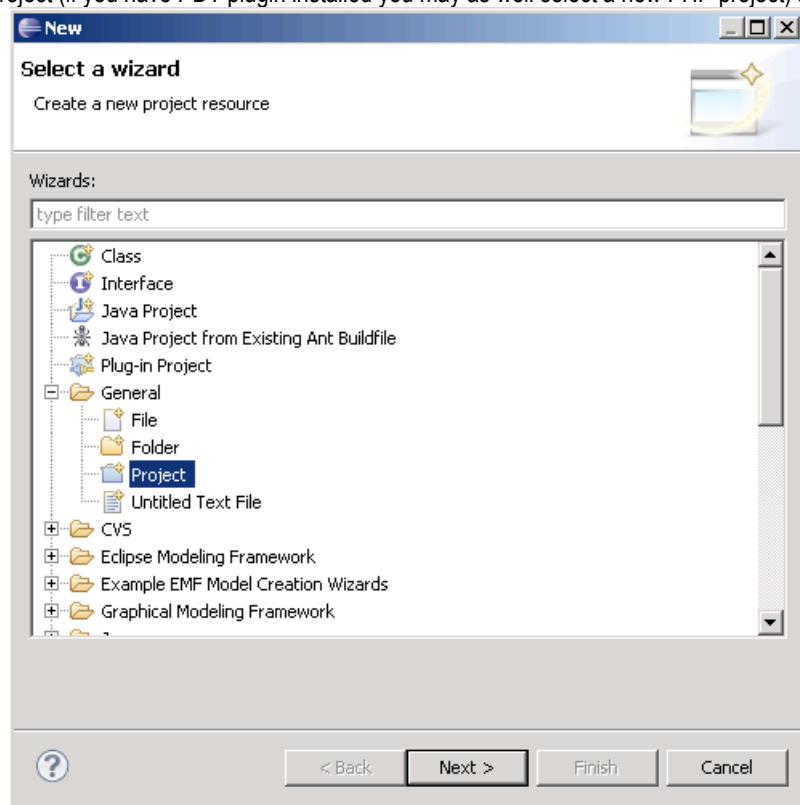
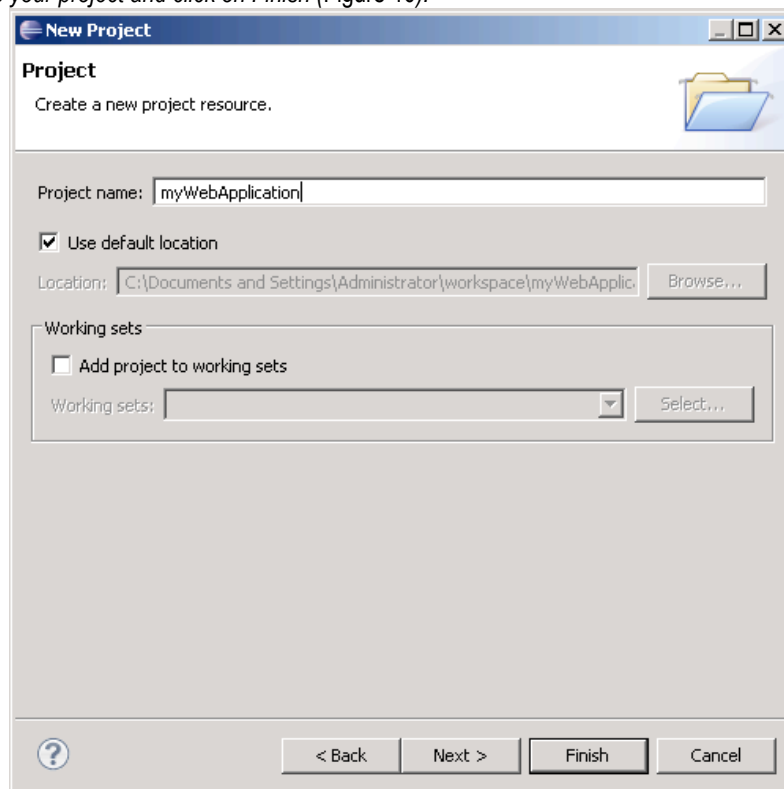


Figure 9

Give a reasonable name to your project and click on Finish (Figure 10).



You should see your project on the left in the Eclipse package explorer. Create then a new UML structural model by selecting File->New->Other..., then selecting UML Class Diagram and clicking Next (Figure 11).

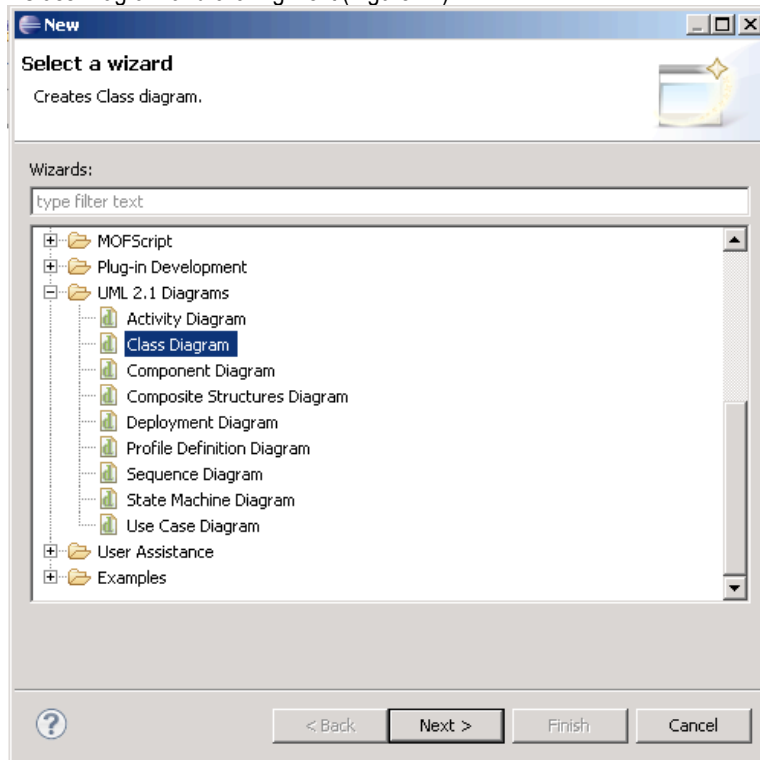


Figure 11

Give your UML diagram and UML model an appropriate name. As a last step the wizard will ask you to select a model object, which is set by default to Package. Change it to Model and click on Finish (Figure 12).



You should now see the toolset to create a UML class models. We have created a sample UML structural model illustrating a very simple Content Management System (CMS) that consists of menu and content objects. Each menu has a caption and an order in which it appears. Typically web systems feature multiple menu levels where a menu can be further subdivided in submenus. This is implemented by the association from Menu to itself where the association end is called parent. Each content object belongs to a certain menu, hence the association to Menu. Contents have a title and a certain text. The model is shown on Figure 13.

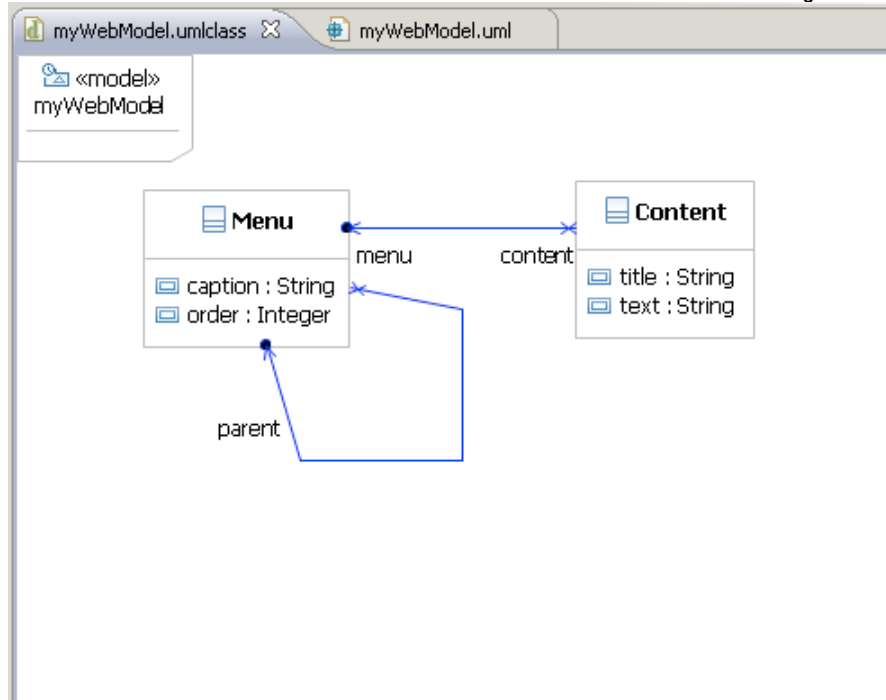


Figure 13

### 3.2.2 Data types

The UML standard currently supports only 4 primitive types: Integer, String, UnlimitedNatural and Boolean. Fortunately, the generator internally supports several additional types. These are currently Date, Text, Double, File and Image. In order to use these you need to create additional classes in your UML model and call them Date, Text, Double, File and Image. These extra classes will have no actual representation in your project but will only serve as types. Once you have done this «hack» you can use these classes as types. Note that the generator will discard these classes (Date, Text, Double, File, Image) from your project and will not generate any model classes or controllers. The influence of the extra types in the generated code is as follows:

**DATE** the mysql type is set to DATETIME; the generated form element in the admin module is a text field automatically linked to a jquery data selector.

**TEXT** the mysql type is set to TEXT; the generated form element in the admin module is a textarea automatically linked via Javascript to the FCKEditor.

**DOUBLE** the mysql type is set to DOUBLE(10,2); the generated form element in the admin module is a standard text field.

**FILE** the mysql type is set to VARCHAR(50); the generated form element in the admin module is a file input.

**IMAGE** same as FILE.

### 3.2.3 Associations as attributes

Associations and attributes are a source of common confusion when learning to use UML. Whenever you create an association between two classes a property is automatically created.

reverse the navigability where we have a one-to-many navigation from a menu to content this will generate additional functionality per each menu allowing to add and modify contents straight from the Menu object.

### 3.2.4 Alternative Modeling Tool: MagicDraw

Even though Eclipse offers a nice tool for creating UML models it is somewhat limited and not always handy to create UML models. The good news is that actually any UML modeling tool capable of exporting or saving to the EMF UML2 XML format will do the job. We suggest using the MagicDraw tool. Please consult the [MagicDraw](#) website in order to obtain a latest version of this tool. Although the tool is not for free and only a demo version is available, the evaluation period practically never expires. Upon expiration a new key can be requested. However, the tool is limited to a maximum of 25 classes. You will need the Architect or Enterprise edition in order to be able to export the created model to Eclipse.

Once you have created the UML model in MagicDraw you need to use its exporting functionality and export the model to EMF UML2. Select File->Export To->EMF UML2 (v2.x) XML File. Select the eclipse project folder in your workspace as the destination folder and hit Export. If you refresh your workspace you will see the exported .uml file along with some standard MagicDraw and UML profiles.

It is also possible to integrate MagicDraw within Eclipse. Please consult MagicDraw documentation on how this can be done.

MagicDraw does offer extra types such as float, double, date, file, etc. However, these extra types may get lost during the export step so we suggest always adding the DATE, TEXT, DOUBLE, FILE and IMAGE types.

## 3.3 Code Generation: “just click a button”

Once you have created the UML model using your favorite UML tool you are ready to generate the code. We assume that at this step you have the .uml file in the project that you have created in previous section.

### 3.3.1 Click a button

Once the structural diagram is complete we can generate the source code using BOZA PHP Generator->Generate... menu.

When invoked the generator allows you to browse for the model file (Figure 14). Note that the actual UML model is contained within the .uml file rather than the .umlclass file. Select the .uml file and click on Open.

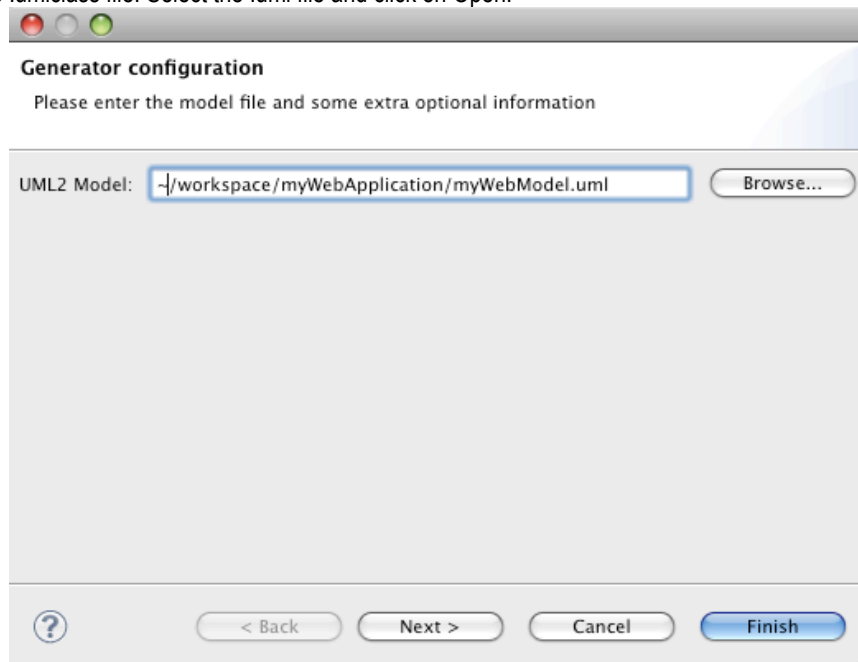


Figure 14

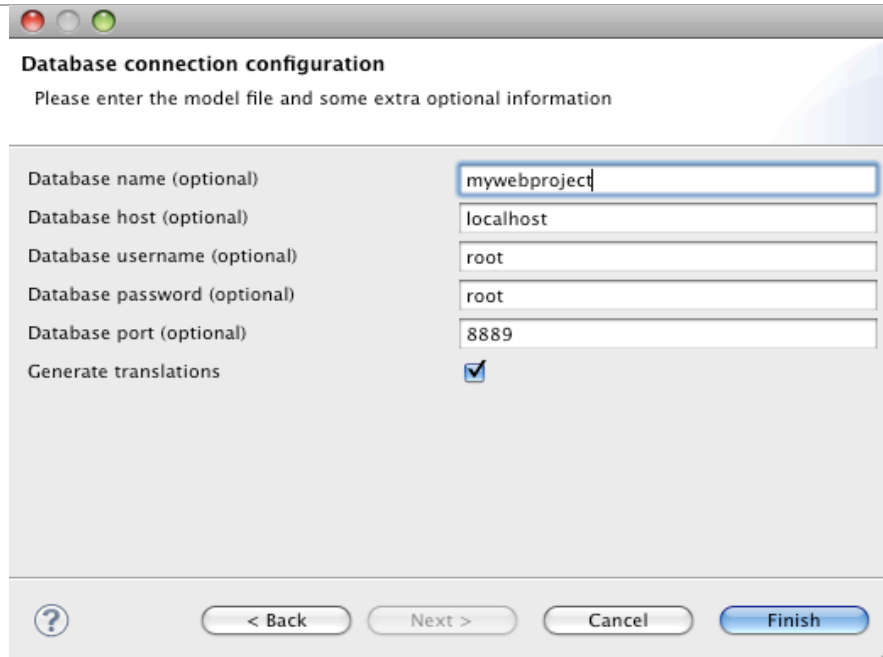


Figure 15

Since the framework version 1.2.6 there is an extra option that you could use to generate the so-called translations module. The translations module allows you to define multilingual terms that are saved in the database. Comment out the code in `/application/controllers/IndexController.php` to load the translations that you define into your session variable. Once the structural diagram is complete we can generate the source code using BOZA PHP Generator->Generate... menu. When invoked the generator asks for the model file. Note that the actual UML model is contained within the `.uml` file rather than the `.umclass` file. Select the `.uml` file and click on Open. The generator runs and creates part of the modeled application.

### 3.3.2 Generated structure<sup>1</sup>

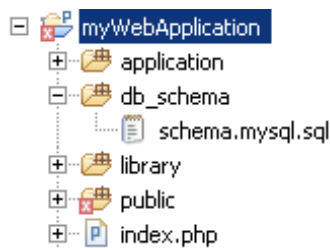
The generation process creates an initial structure of your project along with the complete model files, a MySQL data scheme, a backend admin module with a login. In this section we will explain briefly the structure of the generated system.

#### 3.3.2.1 Application configuration

The application configuration information is contained within the `/application/config/application.ini` file. Please consult Zend documentation to see the different configuration options. One of the things you will probably need to adjust is the database connection information in the `[development:production]` section in the `application.ini` file.

#### 3.3.2.2 MySQL data scheme

The MySQL data scheme is generated in the `/db_schema` directory as shown on Figure 16. The generated schema is compliant with the MySQL norms and will not be discussed further.



### 3.3.2.3 Application startup

Figure 17 gives an overview of the `/public` folder. This is the starting point of the generated application is contained within the `/index.php` file. The `/public` folder contains the CSS, JavaScript and image files grouped by modules. Currently the generation framework supports 2 modules. The default module is the web project front-end. The admin module is the back-end. Obviously, `/public/default` contains the files used in the front-end and the `/public/admin` – the files used in the admin module. A number of helpful JavaScript utilities (such as the FCKEditor, Datepicker, jQuery libraries, etc.) are by default added to the admin module.

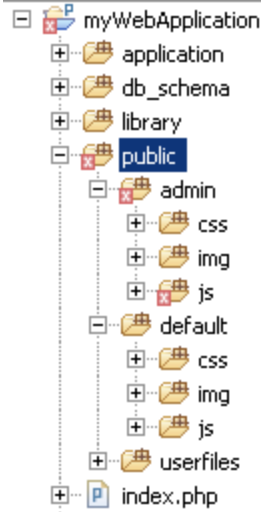


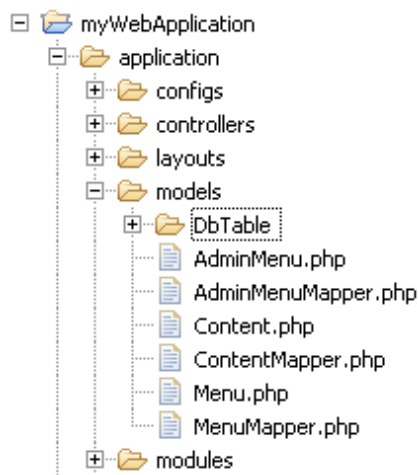
Figure 17

### 3.3.2.4 Zend library

The Zend Framework library version 1.8.4 is placed under `/library/Zend`. In addition to the Zend Framework there are a number of custom classes in `/library/BOZA`. `AuthAdapter.php` is a placeholder class for overriding the default authentication. Two extra library classes `LayoutLoader.php` and `Layout.php` are added to ensure the correct routing to the Admin module when you use the `www.mywebproject.mydomain/admin` URL.

### 3.3.2.5 Application Model (MVC)

The application model classes and model to database mapping classes are contained within the `/application/models` folder. The `/application/models/DbTable` folder contains the link between each class and its corresponding table in the database. The `Class.php` files are the actual model classes. Finally, each class has its corresponding `ClassMapper.php` that realizes the actual saving and retrieving from the database (see Figure 18).



**Controllers** reside within the *controllers* subfolder. The *IndexController.php* and *LoginController.php* are standard files that are copied from the library. The first one fetches and creates the admin menu structure. The *LoginController* handles the admin authentication. The rest of the controllers are generated. Each class comes with its own controller that has 4 actions: index, add, modify and delete.

**Forms** are contained within the *forms* subfolder. The *LoginForm.php* is copied from the library. All other forms are generated using the attributes of the modeled classes.

One **layout** called *admin.phtml* is copied from the library and creates the overall admin design. This file is contained in the *layouts/scripts* subfolder.

Finally, **views** are contained in the *views/scripts* subfolder. Besides copying the standard index and login views, the generator creates add, modify and index views for each class.

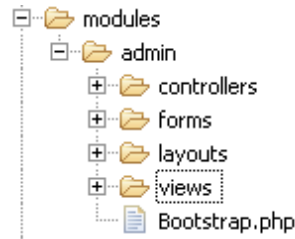


Figure 19

### 3.3.2.7 Application Front-end

Obviously, the application front-end is the main part of the application you will need to implement manually. Figure 20 illustrates the overall structure of the front-end. We will not discuss it further.

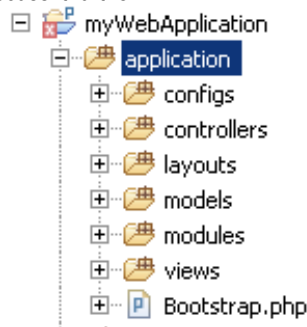


Figure 20

### 3.3.2.8 Translations Module (Optional)

BOZA Framework allows you to generate a standard out-of-the-box translations module that could be used for multilingual sites. The translations module should only be used for translating different messages, button labels, etc. and not the complete texts or menus. The module does not make use of the Zend\_Translate adapters.

The translation module consists of 3 classes: *Term*, *TermCategory* and *TermTranslation*. The class *Term* defines a term to be used on the website. *TermCategory* allows you to categorize the terms. Finally, *TermTranslation* allows you to create translations in different languages for the *Term* objects.

These translated terms are loaded in the */application/controllers/IndexController.php* and set available as a session variable. Note, you need to uncomment the code that actually performs this action.

This is an approach that works. We are still analyzing the pros and cons of it.

### 3.3.3 Manually modified code: protected code sections

Consider the following scenario: you have successfully generated the code for a first version of the application you are building. You have also manually completed the implementation by adding new classes and new operations to existing classes. What happens if you

### **3.4 Generator framework customization**

As we have mentioned the whole BOZA framework is open source and it is a relatively straightforward process to adjust the generator to generate a different (better) structure, use different standard data types, different backend design, etc. etc. etc. The generation engine has 2 passes.

The first pass copies an initial project structure along with the Zend library and some helpers, images, css files, JavaScript files, etc. The contents of the first pass are copied from the *org.boza.phpgen\_<version>/resources* in the Eclipse plugin folder. If you adjust the contents of this folder you will obtain a customized generated structure.

The second pass parses the UML model and generates the MySQL data model, the model classes, and the backend controllers and views. The template engine is currently not open for any customizations. Please contact us if you would like to contribute.

Any further customizations are out of the scope of this document as well. BOZA offers consultancy services what concerns any customized code generation. Please contact us if you would like to obtain any further information.

## **4 Common Problems and FAQ**

The BOZA code generator has been tested on Mac, Linux and Windows systems. Because of the portability of Java there should not be any problems related to the installation. However, here are a number of things that could go wrong. The Framework has been implemented for Eclipse version 3.5.

### **4.1 The BOZA PHP Generator menu does not appear**

This typically means that some of the required Eclipse plugins are missing. Make sure you have followed the installation procedure as described in the Installation section.

In rare occasions this may be caused by an incompatible Java version. Please upgrade your Java to version 1.6.

### **4.2 I click on Generate... and I get an error message: «The chosen operation is currently not available»**

This is caused by an incompatible Java version. Please upgrade your Java to version 1.6.

### **4.3 Can I use a database different from MySQL? Can I use a different structuring of my folders? Can I have some extra helper libraries? Can I do....**

Some of these questions have been answered in the «**Generator framework customization**» section. Some customizations are very easy to make (e.g. adding some extra helper libraries to be generated for each project). However, other customizations are currently not allowed (e.g. modifying the structure of the generated model classes). Please contact us if you would like to contribute or you have any other questions.