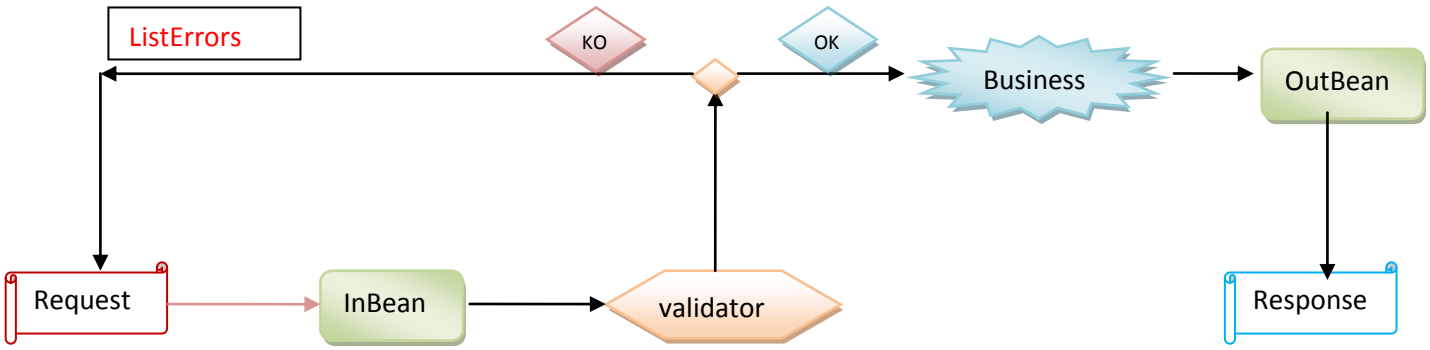
Zerocouplage Tutorial: example of a desktop application

The purpose of this tutorial is to explain how to set up the Zerocouplage framework: configuration files, implementation of business, views and JavaBeans. We will build an application that allows a user to enter his firstname and lastname, based on the Zerocouplage framework.

Overview of Zerocouplage 1.0.0

Zerocouplage 1.0.0 is designed to meet the limitations of various existing MVC frameworks, especially the strong coupling of business logic with the presentation layer.
Indeed, Zerocouplage implements the MVC design pattern in Java, providing a total separation (Zero (0) coupling) between the presentation layer and the business layer. The ultimate goal is to have the opportunity to develop the business logic once and for all, and then through the different presentation layers available you can run the application in multiple environments: web and desktop, for this first version.

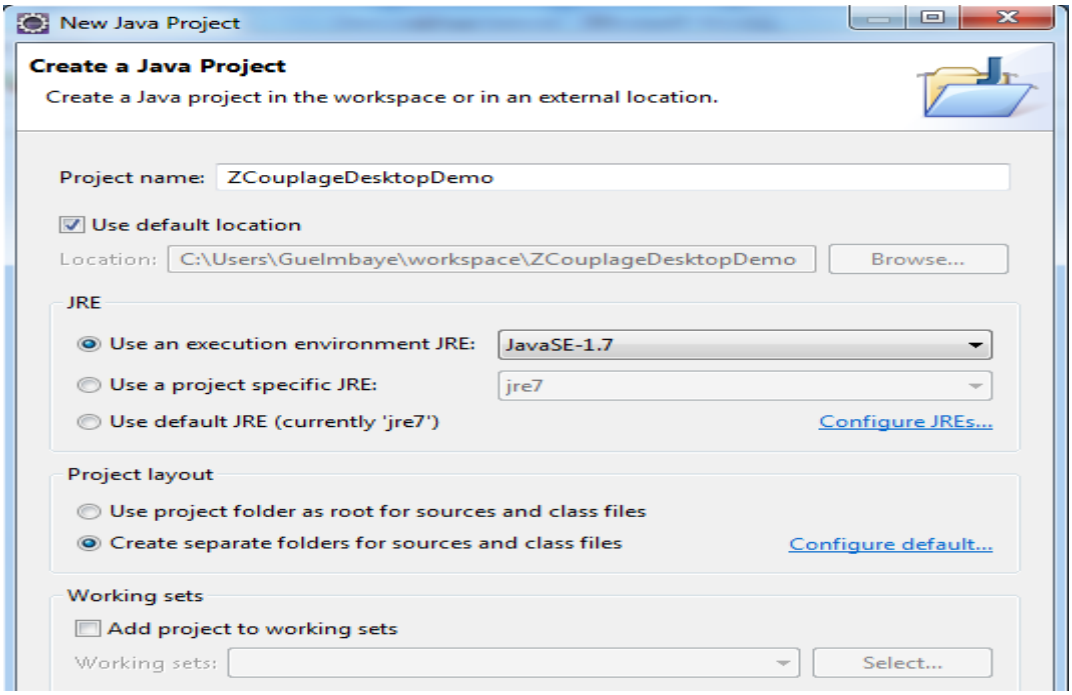
The diagram below defines a request lifecycle in Zerocouplage applications:



* A user sends a Request (from a form). This request is handled by ZCManagerImpl, which determines the appropriate Business.
* ZCManagerImpl InBean creates and stores data entered by the user
* The InBean is then validated using the validator.
- If the data are not all valid: ZCManagerImpl retrieves the list of errors and redisplay the form with the corresponding error (or other error page)
- If all data are valid: processing continues
* The Business method is executed, and the data generated are stored in the OutBean
* The Response is displayed to the user using the OutBean containing the data to be displayed

Application

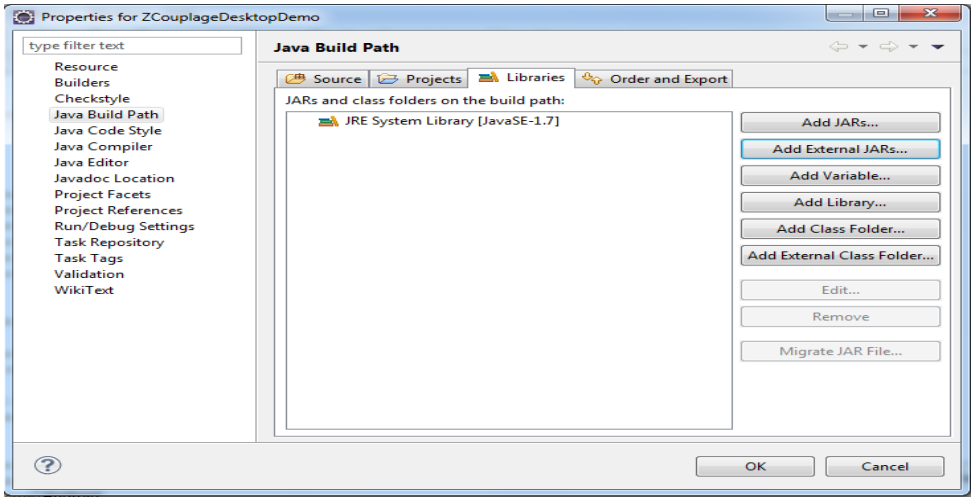
Let's start by creating a new Eclipse Java project "***ZCouplageDesktopDemo***"



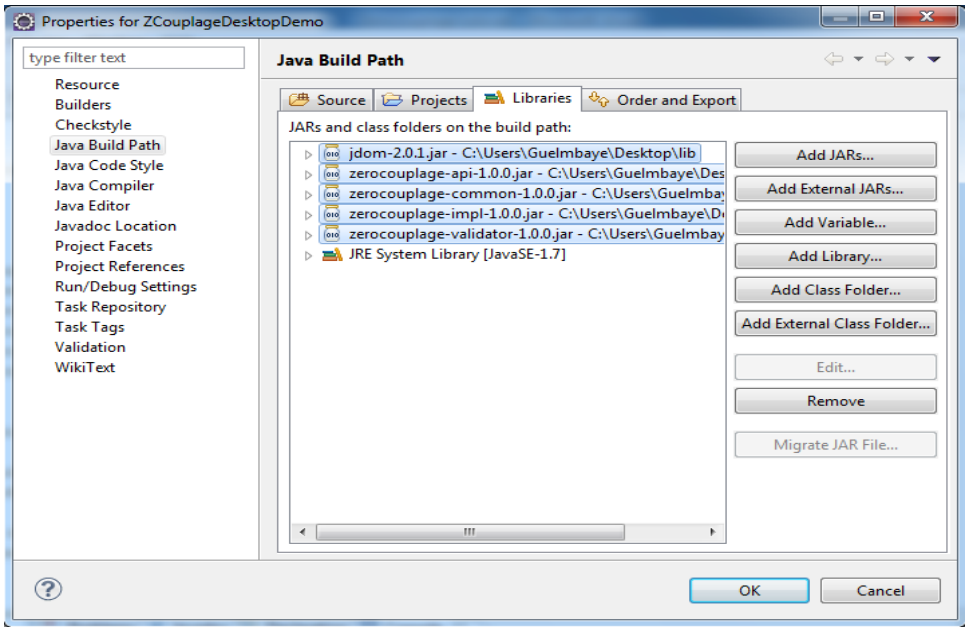
Then, add the following libraries to the project, previously downloaded from the Zerocouplage website (http://zerocouplage.com/):

1. jdom-2.0.1.jar
2. zerocouplage-api-1.0.0.jar
3. zerocouplage-common-1.0.0.jar
4. zerocouplage-impl-1.0.0.jar
5. zerocouplage-validator-1.0.0.jar

Right click on the project, select **"Properties".** Then in the new window, click **"Libraries"** and **"Add External JARs"**:

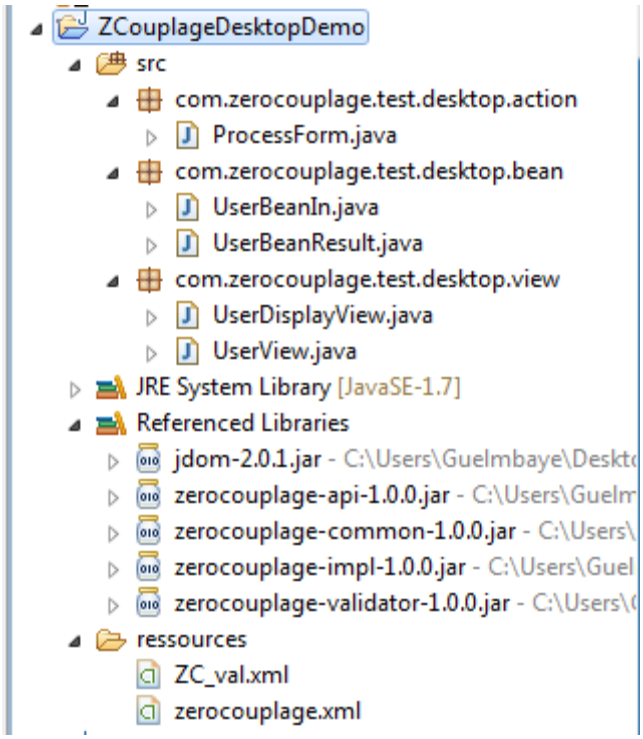


Add the JAR files downloaded and click on "OK"



➔ These libraries are required to create a desktop project based on Zerocouplage

The project structure is described as follows:



We will describe in detail the configuration file of the project.

Configuration file *zerocouplage.xml*

zerocouplage.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<zcouplage DevModes="prod" context="desktop" >

<!-- Mapping   package -->
    <mapping-view-bean>
        <mapping isSame="false" ref-bean="UserBeanIn" ref-view="viewI">
            <key beanProperty="nom" viewProperty="firstName" />
            <key beanProperty="prenom" viewProperty="lastName" />
        </mapping>
    </mapping-view-bean>

<!-- Validators package -->
    <validators-package>
        <validator name="valForm" typeVal="file" valueTarget="ressources/ZC_val.xml" />
    </validators-package>

<!-- Beans-package -->
    <beans-package>
        <bean name="UserBeanIn" class="com.zerocouplage.test.desktop.bean.UserBeanIn" />
    </beans-package>

<!-- Business   package -->
    <business-package>
        <business name="processing" class="com.zerocouplage.test.desktop.business.ProcessForm" method="process" >
            <bean-ref ref-bean="UserBeanIn" />
            <validator-ref ref-val="valForm" />
            <view-results>
                <view-result name="success" bean-result="out" ref-view="viewR" />
            </view-results>
        </business>
    </business-package>

<!-- Views   package -->
    <views-package>
        <view name="viewI" method="createGui" methodError="processError" target="com.zerocouplage.test.desktop.view.UserView" />
        <view name="viewR" method="createResultGui" methodError="" target="com.zerocouplage.test.desktop.view.UserDisplayView" />
    </views-package>
</zcouplage>
```

Zerocouplage.xml file contains the entire application configuration. This file must be in the ***Ressources*** folder, at the root of the project.

The root element ***zcouplage*** contains an attribute ***context*** used to indicate the type of application being developed: desktop or web. In our case, context is set to desktop.
The second attribute ***DevModes*** takes *prod* or *dev* as values. If *DevModes* is set to *prod* zerocouplage.xml file is loaded once for all at the start of the application, otherwise it is reloaded for each change (*dev*).

The application configuration is fragmented into packages:

• The tag ***mapping-view-bean*** defines mappings between views and beans (mapping between the names of the attributes of a bean and form fields for example). ***IsSame*** attribute indicates whether the names of the view properties, and attributes of the bean are the same or not. The ***ref-bean*** and ***ref -view*** attributes are bean and view references involved in the mapping. For example, the first ***key*** element indicates that the bean field "nom" is mapped to the view field "firstName".

• The tag ***validators -package*** contains a set of ***validator*** for the validation of a bean. The attribute ***typeVal*** allows to choose the type of validator: based on XML file (***file***) or Java class (***class***). ***ValueTarget*** attribute indicates the file path or class name, depending on the chosen validator. In this application, we will use an XML file named ***ZC_val.xml***, which will be described later in this tutorial.

• The tag ***beans-package*** contains a set of ***beans*** whose name and class are listed as values of the attributes ***name*** and ***class***.

• The tag ***business-package*** defines the configuration of the various ***businesses*** that will perform the request processing. Business element has three attributes: ***name*** set to the name of the business (associated with a form); ***class*** indicates the business class name, and ***method***, the method that will

perform the business processing. The element ***bean-ref*** is the reference to the bean containing the data entered by a user of the application. The ***validator-ref*** element is used to reference the validator. And finally the ***view-results*** element defines several ***view-result*** elements whose ***name***, ***bean-result*** and ***ref-view*** attributes correspond respectively to the result of the execution of the business method, the name of an instance of bean containing the data generated by the treatment, and the reference to the view to display as result page.

• The tag ***views-package*** contains the views representing the presentation layer of a project.The ***name*** attribute contains the name of the view. The ***method*** and ***methodError*** attributes respectively indicate the names of methods to execute when displaying the home page or result page, and the error page when the data entered by a user are not all valid. The ***target*** attribute is the class name, or file name representing the view.

Validation file ZCval.xml

ZC_val.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bean-validators>
    <global-message>Some fields are required</global-message>
    <bean-validator>
        <field name="nom" constraint="[a-zA-Z]+">
            <message>Firstname is invalid</message>
        </field>
        <field name="prenom" constraint="[a-zA-Z]+">
            <message>Lastname is invalid</message>
        </field>
    </bean-validator>
</bean-validators>
```

Two types of validation are possible: based on Java class or via an xml file.
We will use the second method, but the principle is the same for a class.
This file is named ***ZC_val.xml***.

The root element is ***bean-validators*** that defines a set of ***bean-validator***, and a global message. Each ***bean-validator*** element allows to validate a bean. The ***field*** element represents a bean field, whose name is the value of the ***name*** attribute, and this field must match the regular expression defined by the ***constraint*** attribute. The child element ***message*** of the field contains the error message. The constraint here is that the “nom” and “prenom” must contain only lowercase and / or uppercase letters.

Views (presentation layer)

Here we will create two views: UserView.java and ResultView.java

1. **UserView.java** presents a form to the user to enter his firstname and lastname.
In this class, method defined in ***ZCManagerImpl*** is called, ZCManagerImpl implements the IZCManager interface (See the Zerocouplage API for more information). The ***executeBusiness ("businessname"***) method is called if an event is triggered by a button on the form. To do this, you must create an instance of ***IZCManager***, here named "manager" to run this method.

UserView.java

```java
import com.zerocouplage.api.controller.IZCManager;
import com.zerocouplage.impl.controller.ZCManagerFactory;

public class UserView extends JPanel {

    private static final long serialVersionUID = 1L;

    private IZCManager manager;
```

Here is how to initialize the instance in the constructor of the class UserView:

```java
    public UserView() {
        manager = ZCManagerFactory.getNewManager(this);
    }
```

In our case, the name of the business declared in the configuration file, is *"processing"*.
This business handles the event triggered by the "Submit" button on the form.
Once the button (JButton) "submit" is declared, here is how to associate an event listener and the corresponding treatment:

```java
        submit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent arg0) {
                frame.setVisible(false);

                manager.executeBusiness("processing");
            }
        });
```

Note that the processing to display form is performed by the method ***"createView"*** declared in the configuration file.
The ***"processError"*** method of the class UserView, performs processing for displaying the error page.

```java
    public void processError(HashMap<String, String> listError) {
        StringBuilder builder = new StringBuilder();
        this.frame.setVisible(true);
        for (String fieldName : listError.keySet()) {
            builder.append(fieldName + " : " + listError.get(fieldName)
                    + " \n<br/> ");
        }
        this.errorLabel.setText("<html><font color = red >"
                + builder.toString() + "</font><html>");
    }
```

Each error method declared takes a list of errors as parameter.
In short, in our application: we must define the ***createView*** method to display form, and ***processError*** method for displaying error page.

2. ***UserDisplayView.java*** defined the ***createResultView (UserBeanResult out)*** method declared in the configuration file. This method takes as a parameter an instance of the bean ***UserBeanResult*** (presented in the following section) resulting from business processing. This view displays a message by concatenating the user's firstname and lastname.

```
    public void createResultGui(UserBeanResult out) {
        setLayout(new BorderLayout());
        String message = "Hello " + out.getFirstname() + " "
                + out.getLastname();
        studentDetail = new JLabel(message);
        this.add(studentDetail, BorderLayout.CENTER);
        JFrame frame = new JFrame();
        frame.setContentPane(this);
        frame.setTitle("Hello");
        frame.setPreferredSize(new Dimension(190, 100));
        frame.pack();
        frame.setVisible(true);
        frame.setLocationRelativeTo(null);

    }
```

JavaBeans

There are two classes representing Javabeans: UserBeanIn.java and UserBeanResult.java

The data entered by a user are encapsulated in ***UserBeanIn.***

UserBeanIn.java

```
package com.zerocouplage.test.desktop.bean;

public class UserBeanIn {

    private String nom;
    private String prenom;

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

}
```

UserBeanResult contains data generated after the execution of the executeBusiness ("processing") method;

UserBeanResult.java

```java
package com.zerocouplage.test.desktop.bean;

public class UserBeanResult {

    private String firstname;
    private String lastname;

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

}
```

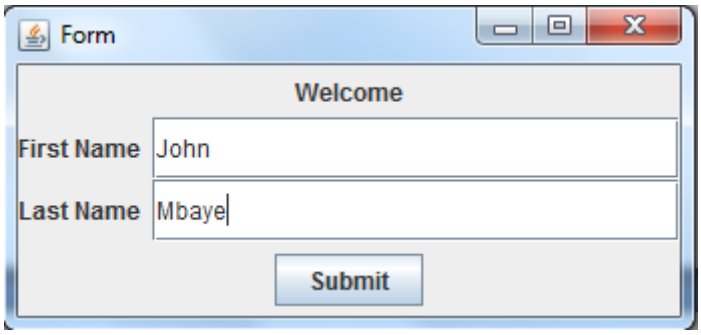
The Business

We will create the class that represents the business logic of the application ***ProcessForm.java***
The business method ***process(UserBeanIn in)*** performing the treatment must return a String, in this case ***"success"***. This method takes as a parameter an instance of the bean ***UserBeanIn*** which contains data entered by the user, and stores the data generated by the treatment in the bean ***UserBeanResult***.

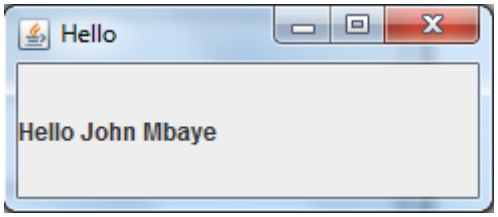
ProcessForm.java

```java
package com.zerocouplage.test.desktop.business;

import com.zerocouplage.test.desktop.bean.UserBeanIn;

import com.zerocouplage.test.desktop.bean.UserBeanResult;

public class ProcessForm {

    private UserBeanResult out;

    public UserBeanResult getOut() {
        return out;
    }

    public String process(UserBeanIn in) {
        out = new UserBeanResult();
        out.setFirstname(in.getNom());
        out.setLastname(in.getPrenom());

        return "success";
    }
}
```

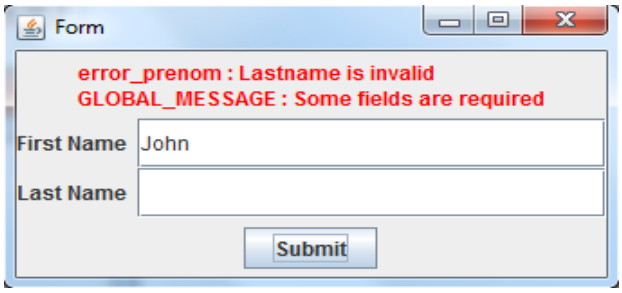
Application execution: when we execute this application, we will have the form (UserView) displayed as follows:



Here is the result page:



If the user does not fill one of the fields (“Last Name” for example) here is the error page :



Conclusion

This tutorial help you to create a desktop application based on the Zerocouplage framework. In the next tutorial, we will see in detail the more advanced aspects. Indeed, we will make two applications, web and desktop, with the same business layer.
You can download the code source for this application here https://zerocouplage.googlecode.com/svn/trunk/code%20source/zerocouplage-1.0.0/zerocouplage-test-desktop-1.0.0/**.**