HIGH MODIFIABILITY MVC FRAMEWORK WITH COMBINED SPRING FRAMEWORK AND MODEL TRANSLATOR

Vittayasak Rujivorakul

School of Information Technology, Mae Fah Luang University, Chiangrai, Thailand
Email: vittayasak@gmail.com

ABSTRACT

This paper proposes a highly modifiable web application framework based on the J2EE platform that applies Model-View-Control (MVC) design pattern and combined Spring Framework with Model Translator to generate more than 14 related classes of source code and more than 6 configuration files for each Create-Read-Update-Delete (CRUD) module. This can reduce impact between layers changed and software development team can start project faster than other modern MVC framework. The result, a software development team can modify in every layer and reduce the time to develop 67-73%, reduce time to initiate project 60%.Moreover the system developed from the proposed framework provides multi-language user interface, unit testing for each layer and can deploy to many platforms that support java.

Index Terms-- *MVC; Spring; J2EE; Framework; Model Translator; Modifiability*

1. INTRODUCTION

The MVC design pattern[1] -is most commonly used for today's web application development. However, there are many available types of technology. This means decisions have to be made regarding the selection of the best of components for use in each layer which are up to experience of the software development team. For the first time of software development team, researcher or students that need to implements their first software project many of them have a problem to select the best components, re-configuration and add more languages to the system.

To solve the problem, this paper proposes the highly modifiable web application framework based on J2EE[2] platform, MVC design pattern, Spring Framework dependency injection and model translator by using model driven architecture tools (MDA)[3]. That can configure through XML configuration and key-value properties file to enable a high degree of modifiability without re-compiling the whole project. All configuration files support major web application needs: loosely couple between layers, support multi-platform and support unit test on each layer.

The paper is organized as follows: section 2 is related work and technology; section 3 is the design concept; section 4 applies the proposed framework; section 5 is the reference system; and section 6 concludes.

2. RELATED WORK AND TECHNOLOGY

2.1. MVC Architecture

MVC[1] is architectural design pattern for interactive applications. MVC separates into three modules: Model- interacts with persistence storage such as database or other systems; View- renders the model into a form suitable for interaction; Controller-receives input and initiates a response by making calls on model objects.

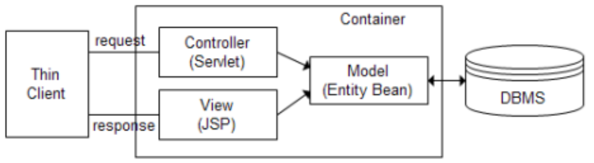


Figure 1. Modern MVC architecture.

From the modern MVC architecture it can be tightly coupled between Model and View, Controller and Model or View and Controller. Furthermore these days we have more choice of technology to make decision to selected for Model (JDBC, DAO, Hibernate, JPA, EJB), View (JSP, JSF, Struts, Spring MVC, Swing) and Controller (Servlet, Struts, Spring Controller)

2.2. Spring Framework

Spring Framework [4] is an open source application framework for the Java platform. The center of Spring Framework is Inversion of Control (IoC) container, which provides a consistent means of configuration and managing Java object using callbacks. The container is responsible for managing object lifecycles: creating objects, calling initialization method, and configuring objects by wiring them together from XML configuration files.

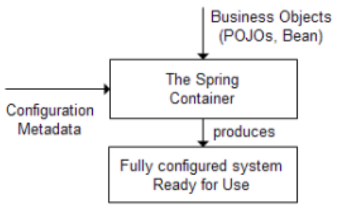

Business Objects (POJOs, Bean)
The Spring Container
Configuration Metadata
produces
Fully configured system Ready for Use

Figure 2. Spring IoC architecture.

The main problem of Spring is many possible configurations and which may cause human error for manual configuration.

2.3. Model Translator

The concept of Model Translator are make a model first by using class diagram and create template to convert graphical model to the target source code or framework. The well know open source Model Translator is AndroMDA[5], which uses Model Driven Architecture (MDA) concept and provides many templates (AndroMDA call “cartridge”) for current web application technology.

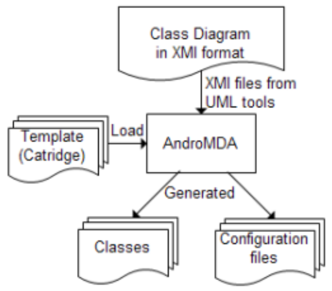


Figure 3. Model Translator architecture.

However many of the templates are too specific and hard to understand for first time software development team.

3. FRAMEWORK DESIGN CONCEPT

Objectives of the proposed framework with respect to research include:

- Can modify components in every layers.
- Supported configuration for multi-language UI.
- Supported unit testing.
- 100 percent generated from template.
- Supported Create Read Update Delete (CRUD) application.

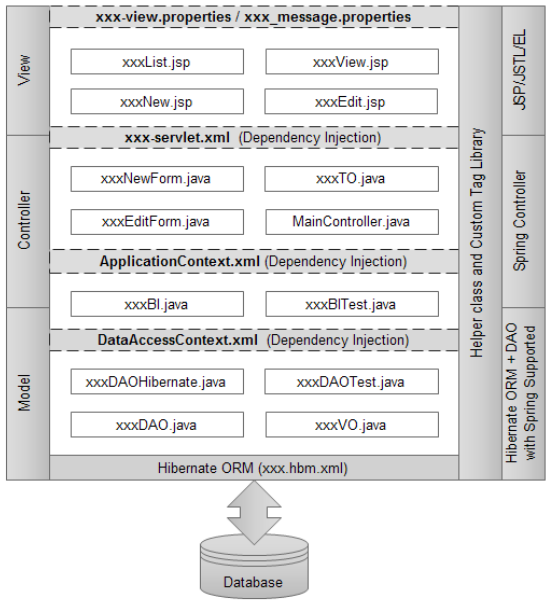

View
Controller
Model
xxx-view.properties / xxx_message.properties
xxxList.jsp
xxxView.jsp
xxxNew.jsp
xxxEdit.jsp
xxx-servlet.xml (Dependency Injection)
xxxNewForm.java
xxxTO.java
xxxEditForm.java
MainController.java
ApplicationContext.xml (Dependency Injection)
xxxBI.java
xxxBITest.java
DataAccessContext.xml (Dependency Injection)
xxxDAOHibernate.java
xxxDAOTest.java
xxxDAO.java
xxxVO.java
Hibernate ORM (xxx.hbm.xml)
Helper class and Custom Tag Library
JSP/JSTL/EL
Spring Controller
Hibernate ORM + DAO with Spring Supported
Database

Figure 4. The propose framework.

From design we use Spring as core configuration and Inject object to the context and applies many J2EE design patterns[5] such as: Data Access Object(DAO), Value Object(VO), Transfer Object(TO), Business Interface(BI), View Controller, View Helper and MVC

The model translator generates 14 classes and can be separated by layer:

- Persistence Layer: VO, DAO, DAOHibernate and DAOTest for unit test (by using jUnit[7]).
- Business Interface: BI and BITest.
- Controller: MainController, NewForm, EditForm and TO for collect data from VO to the View.
- Presentation Layer: ListPage, NewPage, ViewPage and EditPage. All of them use Java Server Page(JSP), Java Standard Tag Library(JSTL) and Expression Language(EL)

Other than classes the model translator also generates 6 configuration files that associate to 14 classes:

- xxx.hbm.xml for Hibernate O/R mapping[8] that associate between VO class and table from database.

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-mapping>
 <class  name="...vo.LecturerVO"  table="Lecturer">
  <id name="id" type="long">  <generator class="assigned"/>  </id>
 <property name="name"  type="String" column="NAME" />
 <property name="phone" type="String" column="PHONE"  />
 <property name="email" type="String" column="EMAIL" />
 <property name="research" type="String" column="RESEARCH"/>
  </class>
</hibernate-mapping>
```

Figure 5. Hibernate O/R mapping configuration file.

- DataAccessContext.xml for setting object and O/R mapping configuration files and associate to DAO class.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans ...>
<bean id="mySessionFactory" class="...LocalSessionFactoryBean">
      <property name="mappingResources">
         <list> <value>hbm/Lecturer.hbm.xml</value> </list>
      </property>
   </bean>                        ORM mapping
   <bean id="lecturerDAO"
         class="….dao.hibernate.LecturerDAOHibernate">
   </bean>
</beans>
```

Figure 6. DataAccessContext file for DAO configuration.

- ApplicationContext.xml for setting DAO to BI class before using in Controller class.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans ...>
<bean id="lecturerBI" class="….TransactionProxyFactoryBean">
      <property name="target">
         <bean class="...bi.LecturerBI">
            <property name="dao"> <ref bean="lecturerDAO"/>
            </property>
         </bean>
      </property>              From DataAccessContext.xml
</bean>
</beans>
```

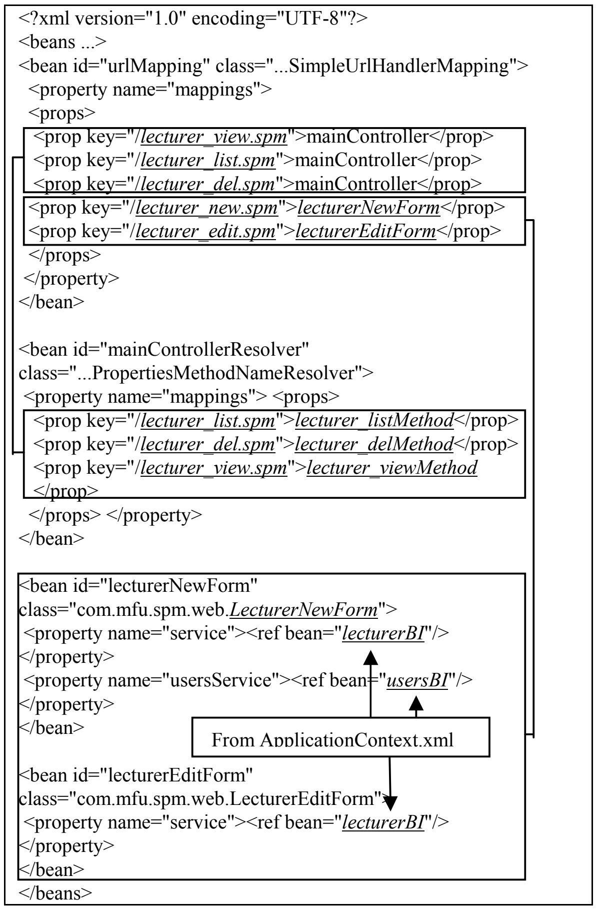
Figure 7. ApplicationContext file for BI configuration.

- xxx-view.properties for associated between JSP page and alias view name that use in Controller configuration file.

```
lecturer_editForm.class=...servlet.view.JstlView
lecturer_editForm.url=/WEB-INF/jsp/Lecturer_edit.jsp
lecturer_newForm.class=...servlet.view.JstlView
lecturer_newForm.url=/WEB-INF/jsp/Lecturer_new.jsp
lecturer_listView.class=...servlet.view.JstlView
lecturer_listView.url=/WEB-INF/jsp/Lecturer_list.jsp
lecturer_viewView.class=...servlet.view.JstlView
lecturer_viewView.url=/WEB-INF/jsp/Lecturer_view.jsp
```

Figure 8. View mapping between JSP page and alias view name.

- xxx-servlet.xml for setting BI and some attribute value to Controller class and associated multi-languages configuration file.



```
<?xml version="1.0" encoding="UTF-8"?>
<beans ...>
<bean id="urlMapping" class="...SimpleUrlHandlerMapping">
  <property name="mappings">
  <props>
   <prop key="/lecturer_view.spm">mainController</prop>
   <prop key="/lecturer_list.spm">mainController</prop>
   <prop key="/lecturer_del.spm">mainController</prop>
  <prop key="/lecturer_new.spm">lecturerNewForm</prop>
  <prop key="/lecturer_edit.spm">lecturerEditForm</prop>
  </props>
 </property>
</bean>

<bean id="mainControllerResolver"
class="...PropertiesMethodNameResolver">
 <property name="mappings"> <props>
  <prop key="/lecturer_list.spm">lecturer_listMethod</prop>
  <prop key="/lecturer_del.spm">lecturer_delMethod</prop>
  <prop key="/lecturer_view.spm">lecturer_viewMethod
  </prop>
 </props> </property>
</bean>

<bean id="lecturerNewForm"
class="com.mfu.spm.web.LecturerNewForm">
 <property name="service"><ref bean="lecturerBI"/>
</property>
 <property name="usersService"><ref bean="usersBI"/>
</property>
</bean>

<bean id="lecturerEditForm"
class="com.mfu.spm.web.LecturerEditForm">
 <property name="service"><ref bean="lecturerBI"/>
</property>
</bean>
</beans>
```

Figure 9. xxx-servlet.xml for Controller configuration.

- xxx_message.properties to setting as key-value for multi-languages feature.

```
#xxx_message.properties
lecturer.id=Id
lecturer.name=Name
lecturer.phone=Phone
lecturer.email=Email
```

Figure 10. Resource bundle file for English UI.

```
#xxx_message_th_TH.properties
lecturer.id=รหัส
lecturer.name=ชื่อ-สกุล
lecturer.phone=เบอร์โทร
lecturer.email=อีเมล์
```

Figure 11. Resource bundle file for Thai UI

From present framework it has many classes and configurations, this means it is very complex and can take a long time for implementation. This paper proposes model translator by using AndroMDA[5] tools. But we need to create template for 14 classes and 6 configurations. Following this is an example 1 of 20 templates that are created by using Velocity language.

```
#set ($webpackagename =
      $packagename.replaceFirst("entity", "web") )
<bean id="urlMapping" class="...SimpleUrlHandlerMapping">
#set ($packagename = $transform.findPackageName(
      ${class.package}))
#foreach ( $onePackage in $model.packages )
#if($onePackage.name == $packagename)
#foreach ( $class in $onePackage.classes )
#set ($lclassname = ${str.lowerCaseFirstLetter(${class.name})})
<prop key="/${lclassname}_view.spm">mainController</prop>
<prop key="/${lclassname}_list.spm">mainController</prop>
<prop key="/${lclassname}_del.spm">mainController</prop>
<prop key="/${lclassname}_new.spm">
    ${lclassname}NewForm</prop>
<prop key="/${lclassname}_edit.spm">
   ${lclassname}EditForm</prop>
 #end
#end
#end
</props>
</property>
</bean>
```

Figure 12. Some part of xxx-servlet.xml template.

4. APPLICATION TO OTHER SYSTEMS

After finishing the creation of the proposed framework and tested. Researcher is setup training class for publish the propose framework. That can follow step-by-step to create the web application system.

- Create class diagram from UML tools (In this case using open source UML tools: ArgroUML) and save as XMI format.
- Convert model from XMI to source code and configuration files.
- Copy generated code and configuration to NetBean IDE[10] project.
- Build and deploy to Glass fish server.

To prove the proposed framework, researchers followed up all attendees to progress their software development project. The result is present in section 5.

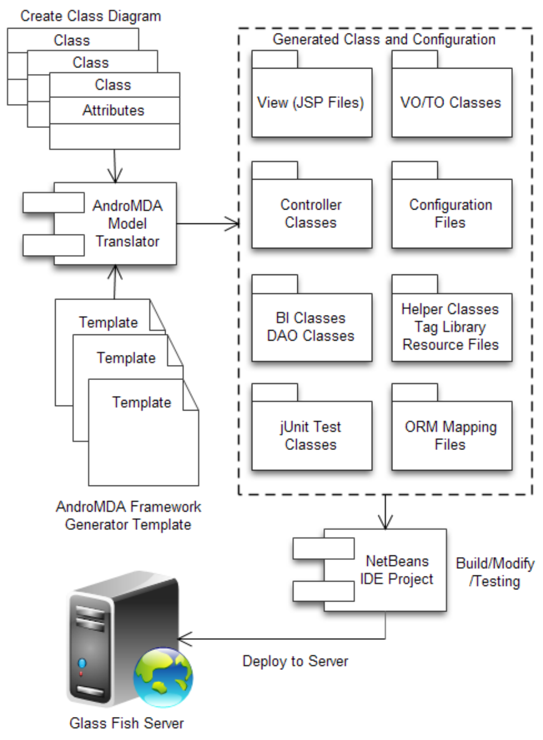


Figure 13. State to apply propose framework.

5. REFERENCE SYSTEM

To prove the proposed framework this paper uses Basic COCOMO[11] model (As an Organic projects) to estimate man-month from following formula.

$$\text{Effort Applied} = a_b \times (\text{KLOC})^\wedge b_b \qquad (1)$$

For Organic project get $a_b = 2.4$, $b_b = 1.05$ and from this framework average LOC to supported each table is 500LOC.This three value are use for next estimation.

- Management Information System for University Qualities Assurance – Developed by 5 developers supported 44 tables. From estimation model need 12.3 month. But by the propose framework they can finish in 4 month. That mean this framework can save time to develop more than 67%. They have more than 20 changes, refactoring and re-configure the system.
- Ubiquitous Learning Environment System- This system was developed to support other research by 2 developers has 17 tables. From estimation model need 11.3 month. But by the propose framework they can finish in 3 month. They can save development time 73%.They have more than 15 changes in the system and provide web-services on this system.

Moreover development times. They also reduce learning and initial project time from normal training process approximately 30 days (training, initial framework, standardize, etc) to just 10 days. That means 60% save time to initial project.

6. CONCLUSION AND FUTURE WORK

This paper proposes a framework that adds highly modifiable which also reduces development time particularly for initial projects. Furthermore each layer can be modified without impact to others layers.

This research can help many developers, researchers, lecturers and students who need to implement small and medium enterprise web application systems. They can use this proposed framework to focus on their research objectives.

For the future work it is possible to add Rich Internet Architecture (RIA)[12] into this framework by using Java Script Framework as jQuery[13] and may be provide adaptor for exchange data as web services in kind of SOAP[14] or REST[15].

REFERENCES

[1] Wikipedia. "Model-View-Controller". http://en.wikipedia.org/wiki/Model-view-controller. 2010.
[2] Oracle Sun Developer Network. "J2EE Framework". http://java.sun.com/javaee/. 2010.
[3] Wikipedia. "Model-driven architecture". http://en.wikipedia.org/wiki/Model-driven_architecture. 2010.
[4] Spring Source. "Spring Framework". http://www.springsource.org. 2010.
[5] AndroMDA.org. "AndroMDA". http://www.andromda.org.2010.
[6] D. Alur, D. Marks and J.Crupi, "Core J2EE Pattern", Prentice Hall, 2 editions, 2003.
[7] jUnit.org. "jUnit". http://www.junit.org/. 2010.
[8] Hibernate.org. "Hibernate". http://www.hibernate.org/. 2010.
[10] NetBeans.org. "NetBean IDE". http://www.netbeans.org. 2010.
[11] Wikipedia. "COCOMO". http://en.wikipedia.org/wiki/COCOMO.2010.
[12] Wikipedia. "Rich Internet Application". http://en.wikipedia.org/wiki/Rich_Internet_application. 2010
[13] jQuery.com. "jQuery". http://jquery.com. 2010.
[14] Wikipedia. "SOAP". http://en.wikipedia.org/wiki/SOAP. 2010
[15] Wikipedia. "REST". http://en.wikipedia.org/wiki/Representational_State_Transfer. 2010.