

## Introduction:

XMoon comes out from the need to simplify and make quicker the work of development of web applications through an Xml-based metalanguage and through different script languages.

Xmoon exploits the Apache Struts framework <http://struts.apache.org/> as basic architecture in compliance to the MVC paradigm.

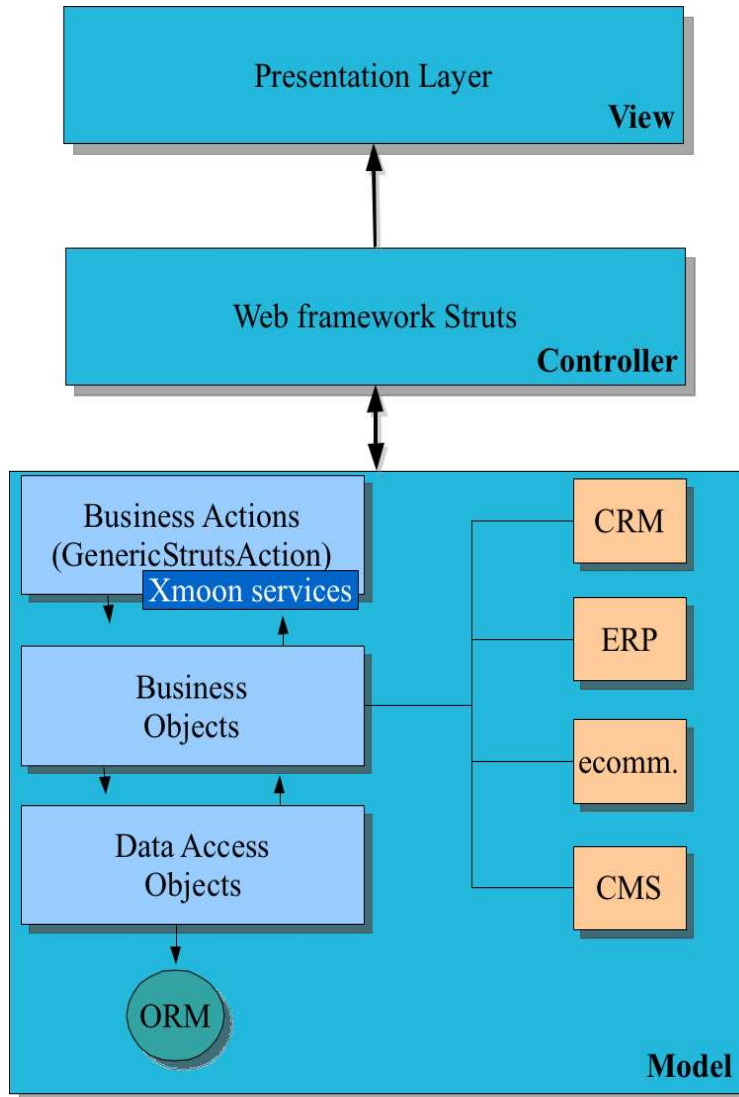


figure1.1

In figure1.1 we can observe an example of an MVC architecture, in which we find the three layers described by the MVC paradigm.

Inside the layer of the Models we find the Xmoon framework which fits into the Business Actions category. Xmoon increases the functionalities of the Actions and allows us to bring outside from these classes the applicative logic thanks to an Xml-based configuration.

## XmoonService:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<service>

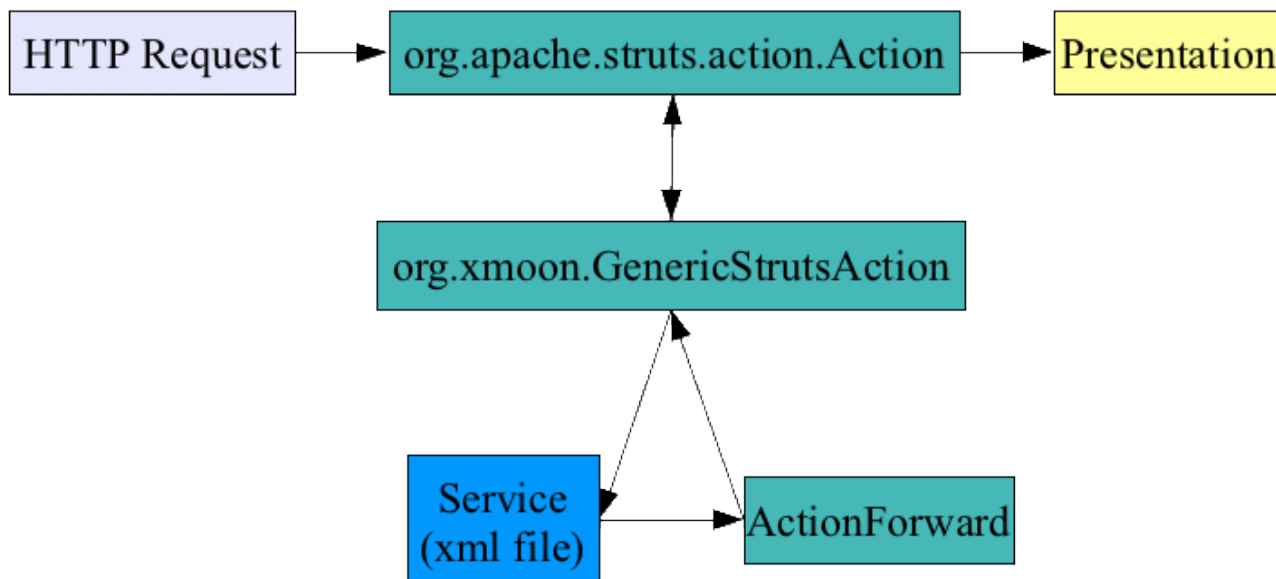
  <bean name="person"
  type="org.apache.commons.beanutils.LazyDynaBean"
  scope="session"/>

  <task>
    <code language="beanshell">
      person.set("name", "John");
      person.set("surname", "Smith");
      person.set("city", "London");

      workbench.setForward("default");
    </code>
  </task>
</service>
```

Figure2.1

A service is a container of applicative logic defined through different script languages inside xml files. It is executed under the control of the **org.xmoon.GenericStrutsAction** class which, at the end of the process, comes back under the control of **org.apache.struts.action.Action**.



#### WEBINF/struts-config.xml

```
<action path="/test" type="org.xmoon.GenericStrutsAction">
  <forward name="default" path="/pages/common/test.jsp"/>
</action>
```

Figure2.2

In **figure 2.2** GenericStrutsAction processes the **/test.do** request whose path is the same as “/test”, org.xmoon.GenericStrutsAction adds to this path an “.xml” suffix and the resultant string corresponds to the file which will be used to process the request.

```
Path “/test” + “.xml” = test.xml
```

Figure2.3

If we want to specify a file which is different from test.xml we can add the **parameter** attribute to the **<action>** tag by defining an alternative xml file.

#### WEBINF/struts-config.xml

```
<action path="/test" type="org.xmoon.GenericStrutsAction" parameter="test2.xml">
  <forward name="default" path="/pages/common/test.jsp"/>
</action>
```

Figure2.4

In **figure2.4** GenericStrutsAction will exploit the **test2.xml** service instead of **test.xml** for the **/test.do** request.

### Configuration:

#### WEBINF/classes/xmoon.properties

```
## SERVICE
xmoon.service.path=WEB-INF/xml/service/
xmoon.service.cache=false
xmoon.service.default_language=beanshell
```

Figure2.5

In **figure2.5** we can observe how to set the services:

1. xmoon.service.path defines the PATH where the xml files are located
2. xmoon.service.cache defines if the cache of the xml files is active/suspended.
3. xmoon.service.default\_language defines which is the default script language if a specific one is not requested.

A **service** is made up by 2 fundamental tag : **<bean>** e **<task>**. We'll see in the next pages the details of these elements.

### **<bean>tag:**

**<bean>** is the tag which has the task to search a determined object by his name and his, if this object has not been found a new one is instanced through a defined class.

The object is put in the BSF contest, the Xmoon scripting engine, and it will be visible and reachable inside the **tasks** defined by the **<task>** tag which we will see later on.

```
<bean
  name="beanInstanceName"
  scope="page | request | session | application"
  type="package.class"
</bean>
```

**Figure3.1**

The purpose of this tag is very much alike the `<jsp:useBean>` one which we find in the Jsp pages.

### <task>tag:

<task> is the tag which executes a portion of code defined inside the <code> tag. This code belongs to the script language defined by the **language** inside of tag <code>.

```
<task>
  <param name="action" value="user.action"/>
  <code language="javascript | ruby | judoscript | beanshell | jython">
  Script code
  </code>
</task>
```

Figure4.1

One task can have a parameter **param**.

- The **value** attribute determines if a task can be executed and this condition is verified when it is present inside the Http request if the attribute is not valorised the task will be always executed.
- The **name** attribute of parameter **param** determines the name of the command that must be run.

In the example reported below the http request will execute the first two tasks, but not the third one.

<b>http://localhost:8080/application/service.do?param1=1&amp;action=crea</b>	
The action of HTTP request is <b>crea</b>	
<pre>&lt;task&gt;   &lt;param name="action" value="crea"/&gt;   &lt;code&gt;// Script &lt;/code&gt; &lt;/task&gt; &lt;task&gt;   &lt;code&gt;// Script &lt;/code&gt; &lt;/task&gt;</pre>	Run correctly action = crea  Run correctly because there aren't any action
<pre>&lt;task&gt;   &lt;param name="action" value="cancella"/&gt;   &lt;code&gt;// Script &lt;/code&gt; &lt;/task&gt;</pre>	Don't run correctly because action = cancella is different to action = crea

Figure4.2

- The **language** attribute determines which script language will be used to evacuate the code contained inside <code>. The code contained in this last tag can be written in different script languages: **BeanShell, JRuby, JudoScript, Jython, Rhino**

```
<task>
  <code language="beanshell"> print("hello World !!!"); </code>
</task>
```

Figure4.3

WEBINF/classes/xmoon.properties

```
xmoon.deniedForward = denied
```

Figure4.4

WEBINF/struts-config.xml

```
<global-forwards>  
  <forward name="denied" path="/pages/common/denied.jsp"/>  
</global-forwards>
```

Figure4.5