**Spring MVC View Resolver**

In this example we shall talk about Spring MVC View Resolvers. View Resolvers are usually provided by all MVC Frameworks, so that models can be rendered in a browser, without being tied to a specific view technology. Spring MVC Framework provides the ViewResolver interface, that maps view names to actual views.

It also provides the View interface, which addresses the request of a view to the view technology. So when a ModelAndView instance is returned by a Controller, the view resolver will resolve the view according to the view name.

Below, we will discuss about three important View Resolver implementations provided by Spring MVC, InternalResourceViewResolver, XmlViewResolver andResourceBundleViewResolver. We will also see how we can make use of them all together.

## Create the View

The view is a simple jsp page, placed in /WEB-INF/ folder. It shows the value of the attribute that was set to theController.

<html>

<body>

<h1>Spring MVC view resolvers example</h1>

<h3>${msg}</h3>

</body>

</html>

## Create the Controller

The HelloWorldController extends the AbstractController provided by Spring, and overrides thehandleRequestInternal(HttpServletRequest request, HttpServletResponse response) method, where aorg.springframework.web.servlet.ModelAndView is created by a handler and returned to be resolved by theDispatcherServlet.

**public** **class** HelloWorldController **extends** AbstractController {

@Override

**protected** ModelAndView handleRequestInternal(HttpServletRequest request,

HttpServletResponse response) **throws** Exception {

ModelAndView model = **new** ModelAndView("helloWorld");

model.addObject("msg", "hello world!");

**return** model;

}

}

## InternalResourceViewResolver

The InternalResourceViewResolver maps the jsp and html files in the WebContent/WEB-INF/ folder. It allows us to set properties such as prefix or suffix to the view name to generate the final view page URL. It is configured as shown below in spring-servlet.xml.

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*"/WEB-INF/views/"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

<context:component-scan base-package=*"in.spring4buddies.application.controller"* />

<bean class=*"in.spring4buddies.application.controller.HelloWorldController"* />

<bean

class=*"org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"* />

When the Controller returns the "helloworld" view, the InternalResourceViewResolver will create the url of the view making use of the prefix and suffix properties that are set to it, and will map the "helloworld" view name to the correct "helloworld" view.

## XmlViewResolver

XmlViewResolver is an implementation of ViewResolver that accepts a configuration file written in XML, where the view implementation and the url of the jsp file are set. Below is the configuration file, views.xml.

views.xml

<bean id=*"helloWorld"* class=*"org.springframework.web.servlet.view.JstlView"*>

<property name=*"url"* value=*"/WEB-INF/views/helloWorld.jsp"* />

</bean>

<bean class=*"org.springframework.web.servlet.view.XmlViewResolver"*>

<property name=*"location"*>

<value>/WEB-INF/views.xml</value>

</property>

</bean>

Now, when the Controller returns the "helloworld" view, the XmlViewResolver will make use of the views.xml file to get the view class and the url of the view that will be mapped to the name "helloworld".

## ResourceBundleViewResolver

The ResourceBundleViewResolver uses bean definitions in a ResourceBundle, that is specified by the bundle basename. The bundle is typically defined in a properties file, located in the classpath. Below is the views.properties file.

**Views.properties**

helloworld.(class)=org.springframework.web.servlet.view.JstlView

helloworld.url=/WEB-INF/helloworld.jsp

The ResourceBundleViewResolver is defined in spring-servlet.xml, and in its definition the basename property is set to view.properties file.

<bean class=*"org.springframework.web.servlet.view.ResourceBundleViewResolver"*>

<property name=*"basename"* value=*"views"* />

</bean>

## Configure multiple View Resolvers together

In order to set multiple Resolvers together in the same configuration file, you can set the order property in all definitions, so that the order that they are used will be defined, as shown below:

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*"/WEB-INF/views/"* />

<property name=*"suffix"* value=*".jsp"* />

<property name=*"order"* value=*"2"* />

</bean>

<context:component-scan base-package=*"in.spring4buddies.application.controller"* />

<bean class=*"in.spring4buddies.application.controller.HelloWorldController"* />

<bean

class=*"org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"* />

<bean id=*"helloWorld"* class=*"org.springframework.web.servlet.view.JstlView"*>

<property name=*"url"* value=*"/WEB-INF/views/helloWorld.jsp"* />

</bean>

<bean class=*"org.springframework.web.servlet.view.XmlViewResolver"*>

<property name=*"location"*>

<value>/WEB-INF/views.xml</value>

</property>

<property name=*"order"* value=*"1"* />

</bean>

<bean class=*"org.springframework.web.servlet.view.ResourceBundleViewResolver"*>

<property name=*"basename"* value=*"views"* />

<property name=*"order"* value=*"0"* />

</bean>