org.springframework.web.servlet

## Class ModelAndView

public class **ModelAndView**

extends [Object](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html?is-external=true)

Holder for both Model and View in the web MVC framework. Note that these are entirely distinct. This class merely holds both to make it possible for a controller to return both model and view in a single return value.

Represents a model and view returned by a handler, to be resolved by a DispatcherServlet. The view can take the form of a String view name which will need to be resolved by a ViewResolver object; alternatively a View object can be specified directly. The model is a Map, allowing the use of multiple objects keyed by name.

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView--)()  Default constructor for bean-style usage: populating bean properties instead of passing in constructor arguments. |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView-java.lang.String-)([**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true) viewName)  Convenient constructor when there is no model data to expose. |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView-java.lang.String-java.util.Map-)([**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true) viewName, [**Map**](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html?is-external=true)<[**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true),?> model)  Creates new ModelAndView given a view name and a model. |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView-java.lang.String-java.lang.String-java.lang.Object-)([**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true) viewName, [**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true) modelName, [**Object**](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html?is-external=true) modelObject)  Convenient constructor to take a single model object. |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView-org.springframework.web.servlet.View-)([**View**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/View.html) view)  Convenient constructor when there is no model data to expose. |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView-org.springframework.web.servlet.View-java.util.Map-)([**View**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/View.html) view, [**Map**](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html?is-external=true)<[**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true),?> model)  Creates new ModelAndView given a View object and a model. |
| [**ModelAndView**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html#ModelAndView-org.springframework.web.servlet.View-java.lang.String-java.lang.Object-)([**View**](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/View.html) view, [**String**](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html?is-external=true) modelName, [**Object**](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html?is-external=true) modelObject)  Convenient constructor to take a single model object. |

## spring-servlet.xml

The name for **spring-servlet.xml** is must match the **servlet-name** we have provided on web.xml. In our case we have provided servlet-name as spring, so we need to append "-servlet.xml"

Web.xml

<display-name> A short name intended to be displayed by GUI tools.

<url-pattern> Describes a pattern used to resolve URLs. The portion of the URL after the http://host:port + ContextPath is compared to the <url-pattern> by WebLogic Server. If the patterns match, the filter mapped in this element is called.

Example patterns:

/soda/grape/\*  
/foo/\*   
/contents  
\*.foo

The URL must follow the rules specified in the Servlet 2.3 Specification.

<servlet-name> Defines the canonical name of the servlet, used to reference the servlet definition elsewhere in the deployment descriptor.

<servlet-class> The fully-qualified class name of the servlet.

Use only one of either the <servlet-class> tags or <jsp-file> tags in your servlet body.

<load-on-startup> WebLogic Server initializes this servlet when WebLogic Server starts up. The optional content of this element must be a positive integer indicating the order in which the servlet should be loaded. Lower integers are loaded before higher integers. If no value is specified, or if the value specified is not a positive integer, WebLogic Server can load the servlet in any order during application startup.

## servlet-mapping

The servlet-mapping element defines a mapping between a servlet and a URL pattern.

The following table describes the elements you can define within a servlet-mapping element.

|  |  |  |
| --- | --- | --- |
| **Element** | **Required/ Optional** | **Description** |
| <servlet-name> | Required | The name of the servlet to which you are mapping a URL pattern. This name corresponds to the name you assigned a servlet in a <servlet> declaration tag. |
| <url-pattern> | Required | Describes a pattern used to resolve URLs. The portion of the URL after the http://host:port + WebAppName is compared to the <url-pattern> by WebLogic Server. If the patterns match, the servlet mapped in this element will be called.  Example patterns:  /soda/grape/\* /foo/\*  /contents \*.foo  The URL must follow the rules specified in the Servlet 2.3 Specification.  For additional examples of servlet mapping, see [Servlet Mapping](http://docs.oracle.com/cd/E11035_01/wls100/webapp/configureservlet.html" \l "wp148787). |

## welcome-file-list

The optional welcome-file-list element contains an ordered list of welcome-file elements.

When the URL request is a directory name, WebLogic Server serves the first file specified in this element. If that file is not found, the server then tries the next file in the list.

For more information, see [Configuring Welcome Files](http://docs.oracle.com/cd/E11035_01/wls100/webapp/configurejsp.html#wp157851) .

The following table describes the element you can define within a welcome-file-list element.

|  |  |  |
| --- | --- | --- |
| **Element** | **Required/ Optional** | **Description** |
| <welcome-file> | Optional | File name to use as a default welcome file, such as index.html |

## jsp-config

The jsp-config element is used to provide global configuration information for the JSP files in a Web application. It has two sub-elements,taglib and jsp-property-group.

The following table describes the elements you can define within a jsp-config element.

|  |  |  |
| --- | --- | --- |
| **Element** | **Required/ Optional** | **Description** |
| <taglib> | Optional | Provides information on a tag library that is used by a JSP page within the Web application. |
| <jsp-property-group> | Optional | Used to group a number of files so they can be given global property information. All files so described are deemed to be JSP files. |

### taglib

This is an element within the [jsp-config](http://docs.oracle.com/cd/E11035_01/wls100/webapp/web_xml.html" \l "wp1071166).

The required taglib element provides information on a tag library that is used by a JSP page within the Web application.

This element associates the location of a JSP Tag Library Descriptor (TLD) with a URI pattern. Although you can specify a TLD in your JSP that is relative to the WEB-INF directory, you can also use the <taglib> tag to configure the TLD when deploying your Web application. Use a separate element for each TLD.

The following table describes the elements you can define within a taglib element.

|  |  |  |
| --- | --- | --- |
| **Element** | **Required/ Optional** | **Description** |
| <taglib-location> | Optional | Gives the file name of the tag library descriptor relative to the root of the Web application. It is a good idea to store the tag library descriptor file under the WEB-INF directory so it is not publicly available over an HTTP request. |
| <taglib-uri> | Optional | Describes a URI, relative to the location of the web.xml document, identifying a Tag Library used in the Web application.  If the URI matches the URI string used in the taglib directive on the JSP page, this taglib is used. |

**javax.xml.bind.annotation   
Annotation Type XmlRootElement**

[@Retention](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Retention.html)([value](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Retention.html#value())=[RUNTIME](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/RetentionPolicy.html#RUNTIME))

[@Target](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Target.html)([value](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Target.html#value())=[TYPE](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/ElementType.html#TYPE))

public @interface **XmlRootElement**

Maps a class or an enum type to an XML element.

**Usage**

The @XmlRootElement annotation can be used with the following program elements:

* a top level class
* an enum type

See "Package Specification" in javax.xml.bind.package javadoc for additional common information.

When a top level class or an enum type is annotated with the @XmlRootElement annotation, then its value is represented as XML element in an XML document.

This annotation can be used with the following annotations: [XmlType](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlType.html), [XmlEnum](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlEnum.html), [XmlAccessorType](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlAccessorType.html), [XmlAccessorOrder](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlAccessorOrder.html).

**Example 1:**Associate an element with XML Schema type

// Example: Code fragment

@XmlRootElement

class Point {

int x;

int y;

Point(int \_x,int \_y) {x=\_x;y=\_y;}

}

//Example: Code fragment corresponding to XML output

marshal( new Point(3,5), System.out);

<!-- Example: XML output -->

<point>

<x> 3 </x>

<y> 5 </y>

</point>

The annotation causes an global element declaration to be produced in the schema. The global element declaration is associated with the XML schema type to which the class is mapped.

<!-- Example: XML schema definition -->

<xs:element name="point" type="point"/>

<xs:complexType name="point">

<xs:sequence>

<xs:element name="x" type="xs:int"/>

<xs:element name="y" type="xs:int"/>

</xs:sequence>

</xs:complexType>

## javax.xml.bind.annotation  Annotation Type XmlAttribute

[@Retention](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Retention.html)([value](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Retention.html#value())=[RUNTIME](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/RetentionPolicy.html#RUNTIME))

[@Target](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Target.html)([value](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/Target.html#value())={[FIELD](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/ElementType.html#FIELD),[METHOD](http://java.sun.com/j2se/1.5/docs/api/java/lang/annotation/ElementType.html#METHOD)})

public @interface **XmlAttribute**

Maps a JavaBean property to a XML attribute.

**Usage**

The @XmlAttribute annotation can be used with the following program elements:

* JavaBean property
* field

A static final field is mapped to a XML fixed attribute.

See "Package Specification" in javax.xml.bind.package javadoc for additional common information.

The usage is subject to the following constraints:

* If type of the field or the property is a collection type, then the collection item type must be mapped to schema simple type.
* // Examples
* @XmlAttribute List<Integer> items; //legal
* @XmlAttribute List<Bar> foo; // illegal if Bar does not map to a schema simple type

* If the type of the field or the property is a non collection type, then the type of the property or field must map to a simple schema type.
* // Examples
* @XmlAttribute int foo; // legal
* @XmlAttribute Foo foo; // illegal if Foo does not map to a schema simple type

* This annotation can be used with the following annotations: [XmlID](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlID.html), [XmlIDREF](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlIDREF.html), [XmlList](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlList.html), [XmlSchemaType](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlSchemaType.html), [XmlValue](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlValue.html), [XmlAttachmentRef](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlAttachmentRef.html), [XmlMimeType](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlMimeType.html),[XmlInlineBinaryData](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/XmlInlineBinaryData.html), [XmlJavaTypeAdapter](http://docs.oracle.com/javaee/5/api/javax/xml/bind/annotation/adapters/XmlJavaTypeAdapter.html)

.

**Example 1:**Map a JavaBean property to an XML attribute.

//Example: Code fragment

public class USPrice {

@XmlAttribute

public java.math.BigDecimal getPrice() {...} ;

public void setPrice(java.math.BigDecimal ) {...};

}

<!-- Example: XML Schema fragment -->

<xs:complexType name="USPrice">

<xs:sequence>

</xs:sequence>

<xs:attribute name="price" type="xs:decimal"/>

</xs:complexType>