The Spring Thing

The blog covers different topics on Spring Framework including IOC, AOP, MVC, JDBC, JMS and others. Each topic would try to provide examples along with the explanation.

WEDNESDAY, NOVEMBER 17, 2010

How does @ModelAttribute work?

@ModelAttribute is a Spring-MVC specific annotation used for preparing the model data. It is also used to define the command object that would be bound with the HTTP request data. The annotation works only if the class is a Controller class (i.e. annotated with @*Controller*).

ModelAttribute can be applied on both methods as well as method-parameters. It accepts an optional *"value"*, which indicates the name of the attribute. If no value is supplied to the annotation, then the value would be defaulted to the return type name in case of methods and parameter type name in case of method-parameters.

The way Spring processes this annotation is,

1. Before invoking the handler method, Spring invokes all the methods that have @ModelAttribute annotation. It adds the data returned by these methods to a temporary *Map* object. The data from this Map would be added to the final Model after the execution of the handler method.
2. Then it prepares to invoke the the handler method. To invoke this method, it has to resolve the arguments. If the method has a parameter with @ModelAttribute, then it would search in the temporary *Map* object with the value of @ModelAttribute. If it finds, then the value from the Map is used for the handler method parameter.
3. It it doesn't find it in the Map, then it checks if there is a SessionAttributes annotation applied on the controller with the given value. If the annotation is present, then the object is retrieved from the session and used for the handler method parameter. If the session doesn't contain the object despite of the @SessionAttributes, then an error is raised.
4. If the object is not resolved through Map or @SessionAttribute, then it creates an instance of the parameter-type and passes it as the handler method parameter. *Therefore, for it to create the instance, the parameter type should be a concrete-class-type (interfaces or abstract class types would again raise an error).*
5. Once the handler is executed, the parameters marked with @ModelAttributes are added to the Model.

Let me explain you further with the helps of some examples.

Example 1:

```
@Controller
public class MyController {

    @ModelAttribute("myobject")
    public MyObject getInitializedMyObject() {
        return myService.getInitializedObject();
    }

    @RequestMapping(value="/handle.htm", method=RequestMethod.GET)
    public ModelAndView handleRequest() {
        return new ModelAndView("myView");
    }

}
```

In this example, the value returned by getInitializedMyObject is added to the Model. The View would be able to retrieve this object using the key "myobject" from the request attributes.

Example 2 :

```
@Controller
public class MyController {

    @ModelAttribute("myobject")
    public MyObject getInitializeMyObject() {
        return myService.getInitializedObject();
    }

    @RequestMapping(value="/handle.htm", method=RequestMethod.GET)
    public ModelAndView handleRequest(@ModelAttribute("myobject") MyObject myObject) {
        myObject.setValue("test");
        return new ModelAndView("myView");
    }

}
```

In this case, the getInitializeMyObject is executed first and the result is stored in a temporary map. This value is then passed as a parameter to the handler method. And finally myObject is added to the model for the views.

Example 3 :

```
@Controller
@SessionAttributes("myobject")
public class MyController {

    @RequestMapping(value="/handle.htm", method=RequestMethod.GET)
    public ModelAndView handleRequest(@ModelAttribute("myobject") MyObject myObject) {
        myObject.setValue("test");
        return new ModelAndView("myView");
    }

}
```

In this case, Spring searches for "myobject" in the session and pass it as the parameter to the handler method. If "myobject" is not found in the session, then HttpSessionRequiredException is raised.

Example 4 :

```
@Controller
public class MyController {

    @RequestMapping(value="/handle.htm", method=RequestMethod.GET)
    public ModelAndView handleRequest(@ModelAttribute("myobject") MyObject myObject) {
        myObject.setValue("test");
        return new ModelAndView("myView");
    }

}
```

In this case, a new instance of MyObject is created and then passed to handler method. If MyObject is an interface or an abstract class, then a BeanInstantiationException is raised.