**Spring MVC Exception Handling**

This is an example of how to handle exceptions in Spring MVC. In Spring MVC Framework exceptions may occur when a controller is invoked. These exceptions may be handled either with the help ofSimpleMappingExceptionResolver class, or with the @ExceptionHandlerannotation.

SimpleMappingExceptionResolver is a HandlerExceptionResolverimplementation for mapping exception class names to view names, either for a set of given handlers or for all handlers in the DispatcherPortlet. It is set and defined in the container xml configuration file. On the other hand, when xml configuration needs to be avoided, the @ExceptionHandlerannotation can be used on Controller methods that will thus be invoked to handle the exception.

Below you can check on how to implement both ways. We will create a simple project with two Controller classes, each one for a case. We will also create a custom Exception class to be used as the exception to handle, and an error page that will be the view part of our MVC project example.

## Create the error page

The error page is a simple jsp page, placed in /WEB-INF/ folder. It shows the value of the attribute that was set to theException when thrown by the controller.

<html>

<body>

<h1>Spring 4.0.2 MVC HelloWorld Controller</h1>

<h3>Error page..: "${exception.message}"</h3>

</body>

</html>

## SimpleMappingExceptionResolver case

In this case a simple controller is created that throws an exception when invoked. Its configuration will be shown in mvc-dispatcher-servlet.xml file below. FirstController.java class extends theorg.springframework.web.servlet.mvc.AbstractController and overrides its handleRequestInternal(HttpServletRequest request, HttpServletResponse response) method, where the exception is thrown.

CustomException.java class is the exception that will be thrown in both cases. Note that the same steps can be followed in the controllers when we need to handle all types of Java exceptions, such as java.lang.Exception.

**public** **class** CustomException **extends** Exception {

**private** **static** **final** **long** ***serialVersionUID*** = -4744430183523721711L;

**private** String message = "This is an exception..";

**public** CustomException(String message) {

**this**.message = message;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

}

## @ExceptionHandler case

The second controller makes use of the @ExceptionHandler annotation. SecondController.java class is an @Controllerannotated class, thus declaring that it is a Controller. It has a method, second(), annotated with the @RequestMappingannotation. This annotation maps a URL to either an entire class or a particular handler method. This method also throws aCustomException.

In this case the exception is handled by the @ExecptionHandler annotation, which is set in thehandleCustomException(CustomException ex) method. This method returns the error view created above, making use of theorg.springframework.web.servlet.ModelAndView class.

## Configure spring-servlet.xml file

The spring-servlet.xml file must be enriched with configuration concerning both controllers.  
For the first controller, the org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping is used, which is defined as a bean. It is used to map any URL requests that start with first or first\* to theFirstController.java class. The first controller is also defined as a bean here.

The important bean to define and configure for the first controller is theorg.springframework.web.servlet.handler.SimpleMappingExceptionResolver bean. It has a property namedexceptionMappings. Its value is a key-value pair, consisting of the CustomException class declaration and the error.jsppage declaration. This is where the mapping of the exception and the view is configured for the first controller.

For the second controller, the context:component-scan base-package tag is used, so that the container will scan the base package declared to search for the annotated class. The mvc:annotation-driven tag is also used, to declare explicit support for annotations in the controller.

<bean

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

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

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

</bean>

<bean

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

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

<bean

class=*"org.springframework.web.servlet.handler.SimpleMappingExceptionResolver"*>

<property name=*"exceptionMappings"*>

<props>

<prop key=*"com.javacodegeeks.snippets.enterprise.CustomException"*>

error

</prop>

</props>

</property>

</bean>

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

<mvc:annotation-driven />

http://localhost:8080/sf-mvc-exception-handling-xml/first.htm