Restful

Summary

RESTful implemented through Spring MVC uses URI for approach to the resource and since it uses meaning of methods such as PUT, GET, POST and DELETE of HTTP, it can be accessed simply.

Description

web.xml Setting

```
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.xml</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.json</url-pattern>
</servlet-mapping>

<filter>
        <filter-name>httpMethodFilter</filter-name>
        <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>httpMethodFilter</filter-name>
        <url-pattern>/springrest/*</url-pattern>
</filter-mapping>
```

Detailed explanation is below.

Request Mapping

▪ Setting

The URL of REST is designed to be used in the clas structure like '/cgr', '/cgr/CATEGORY-00000000001'. Accordingly, it is required to define DispatcherServlet in web.xml and designate the URL pattern for mapping with '/'. **DispatcherServlet URL mapping**sample is as follows:

```
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>/springrest/*</url-pattern>
</servlet-mapping>
```

DispatcherServlet URL mapping can be used in the following method:

```
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.xml</url-pattern>
```

```
</servlet-mapping>
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.json</url-pattern>
</servlet-mapping>
```

▪ Use

The REST support functions provided by Springare based on Spring MVC. Since the service exposed in REST type is the method of controller, it is not that different from the type of developing web application in the past.

To map URI, the ID of Resource to Controller class or method, use **@RequestMapping**. Since @RequestMapping supports URI Template, following sample code can be used.

```
@Controller
@SessionAttributes(types=CategoryVO.class)
public class EgovCategoryController {
        //…
@RequestMapping(value="/springrest/cgr/{ctgryId}", method=RequestMethod.GET)
publicStringupdtCategoryView(@PathVariableStringctgryId, Model model) throws Exception{
            // …
    }
}
```

To use all HTTP method, @RequestMapping provides 'method' property. Accordingly, if URI '/springrest/cgr/CATEGORY-00000000001' receives request from GET, the above updtCategoryView ( ) method will be mapped.

▪ Add @PathVariable annotation

When URI request comes in '/springrest/cgr/CATEGORY-00000000001', @PathVariable is used and bound to 'ctgryID' input factor.

```
@RequestMapping(value="/springrest/cgr/{ctgryId}", method=RequestMethod.GET)
publicStringupdtCategoryView(@PathVariableStringctgryId, Model model) throws Exception{
        // …
}
```

HTTP Method Conversion

▪ Setting

Browser-based HTMLs support GET and POST only. In general, HTTP uses POST and designate HTTP METHOD with hidden type input value. Following is the example of defining **HiddenHttpMethodFilter** in web.xml.

```
<filter>
        <filter-name>httpMethodFilter</filter-name>
        <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>httpMethodFilter</filter-name>
        <url-pattern>/springrest/*</url-pattern>
</filter-mapping>
```

▪ Use

Adding HiddenHttpMethodFilter setting to web.xml, **HTTP Method is POST. If parameter called _method exists, it changes the method of HTTP to the _method value.**

In addition, since Spring automatically adds the input field of hidden type that designates actual HTTP Method in <form:form>, you can use it much conveniently.

```
<form:form method="delete">
        <input type=submit value=Delete/>
</form:form>
```

If it is written in JSP as shown above, "_method=delete" is delivered in POST type internally.

Following is the sample code.

```
functionfncSubmit(method) {
    document.detailForm._method.value=method;
document.detailForm.submit();
}

//..

<form:form name="detailForm" method="${method}">
<a href="javascript:fncSubmit('delete');">삭제</a>
</form:form>
```

HTTP Method Conversion

This shows other views such as Xml and json and Spring provides **ContentNegotiatingViewResolver**. Since ContentNegotiatingViewResolver should be used together with other View Resolver, the order must be defined when setting View Resolver. Of course, ContentNegotiatingViewResolver should have the highest priority (the smallest number). defaultView is used as a default view if the view was not found.

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
<property name="defaultViews">
        <list>
        <bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView">
                <property name="prefixJson" value="false"/>
        </bean>
        </list>
</property>
</bean>
```

Views

· MarshallingView

This uses Spring OXM Marshaller to return xml response to the client. Spring oxm enables to define Marshaller EASILY USING JAXB2, XMLBeans, JiBX and Castor. The Restful example uses JAXB2. (OXM example uses Castor)

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
        <property name="mediaTypes">
                <map>
                        <entry key="html" value="text/html" />
                        <entry key="xml" value="application/xml" />
                        <entry key="json" value="application/json" />
                </map>
        </property>
        <property name="order" value="0" />
//..
</beans>
```

```
<bean name="cgr/egovCategoryRegister"
class="org.springframework.web.servlet.view.xml.MarshallingView">
        <property name="marshaller" ref="marshaller" />
</bean>

<oxm:jaxb2-marshaller id="marshaller">
        <oxm:class-to-be-bound name="egovframework.rte.tex.cgr.service.CategoryVO" />
</oxm:jaxb2-marshaller>
```

▪ MappingJacksonJsonView

This is a view to be able to deliver a response to JSON.

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
        <property name="mediaTypes">
                <map>
                        <entry key="html" value="text/html" />
                        <entry key="xml" value="application/xml" />
                        <entry key="json" value="application/json" />
                </map>
        </property>
        <property name="order" value="0" />
        //..
</beans>

<bean name="cgr/egovCategoryList"
class="org.springframework.web.servlet.view.json.MappingJacksonJsonView" />
```

Example of Actual Use

▪ jsp

```
functionfncSubmit(method) {
        document.detailForm._method.value=method;
        document.detailForm.submit();
}

//..

<form:form name="detailForm" method="${method}">
//..
</form:form>

//..
<a href="javascript:fncSubmit('post');">Register</a> //---------- controller 2.
<a href="javascript:fncSubmit('put');">Update</a> //------------ controller 3.
<a href="javascript:fncSubmit('delete');">Delete</a> //---------- controller 4.
<a href=" /springrest/cgr/{id}.xml">View xml</a> // ContentNegotiatingViewResolver setting
<a href=" /springrest/cgr/{id}.json">json(defaultView) View</a> // ContentNegotiatingViewResolver
setting
<a href=" /springrest/cgr.html">List</a> //----------------- controller 1.
<a href=" /springrest/cgr.json">List(json)</a> // ContentNegotiatingViewResolversetting
```

▪ controller

```
// 1. List
@RequestMapping(value="/springrest/cgr", method=RequestMethod.GET)
publicStringselectCategoryList(..) throws Exception {
        //..
}

// 2. Register
```

```
@RequestMapping(value="/springrest/cgr", method = RequestMethod.POST, ..)
publicString create( ..) throws Exception {
		//..
}

// 3. Update
@RequestMapping(value = "/springrest/cgr/{ctgryId}", method = RequestMethod.PUT, ..)
publicString update(..) throws Exception {
		//..
}

// 4. Delete
@RequestMapping(value = "/springrest/cgr/{ctgryId}", method=RequestMethod.DELETE)
publicStringdeleteCategory(@PathVariableStringctgryId, SessionStatus status) throws Exception{
		//..
}
```

Reference

- RESTful Example