REST using SpringMVC

INF5750/9750 - Lecture 4 (Part I)

Problem area

- Transferring state over the network
- HTTP is a stateless protocol

- CORBA
- RPC? RMI?
- Serialization as File?

- REST is an architectural style, a way to design web-services (WS) or web-api
- Not all Web-api are RESTful

- **We are skipping the details of what makes a WS RESTful**

HTTP & REST terms

- HTTP methods, also referred as “verbs” for web language
  - GET, POST, PUT, DELETE
  - … OPTIONS, HEAD, PATCH …

- HTTP Status codes
  - 1xx - Informational (100-continue ; 102-processing)
  - 2xx - Success (200-OK; 201-Created; 204-NC)
  - 3xx - Redirection (301-Moved; 302-Found;...)
  - 4xx - Client Error (400-bad request; 401-unauthorized..)
  - 5xx - Server Error (500-internal server error;...)

- Resources - sources of information
  - Are identified using URI.
- Representation - format of the information (using mimetype)

Content Negotiation

- The process by which the client determines the representation of the resource

- Can be done through URL extension
  - http://localhost:8080/restService/person.json
- Can be done through HTTP header
  - Accept: application/json

- Let us look at client-side behavior
  - http://apps.dhis2.org/demo/api/resources

Spring MVC for REST

- Spring MVC is well suited to create web services because it is based on URL mapping for requests and can flexibly respond different content types
- Resources become models for Controllers
- Representations are Views or RequestBody
- Spring integrates well with a number of serializers such as for JSON (Jackson) or XML (JAXB)

ContentNegotiatingViewResolver

- Does not resolve views itself like UrlBasedViewResolver, rather delegates to others
- Two strategies
  - Use a distinct URI for each resource (.xml; .json)
  - Use same URI, but set the Accept request header

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
   <property name="mediaTypes">
      <map>
         <entry key="html" value="text/html"/>
         <entry key="json" value="application/json"/>
      </map>
   </property>
   <property name="viewResolvers">
      <list>
         <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <property name="prefix" value="/WEB-INF/pages/"/>  <property name="suffix" value=".jsp"/>
         </bean>
      </list>
   </property>
   <property name="defaultViews">
      <list>
         <bean class="org.springframework.web.servlet.view.json.MappingJackson2JsonView" />
      </list>
   </property>
</bean>
```

Changing the Controller

- Remember from previous examples that Controllers return a String for view name
- Instead use ***@ResponseBody***, to respond with data in the format the client requested
- Spring will try to resolve the data into the format
- If Jackson-mapper is on classpath and JSON is Accept from client, then Spring will return a JSON string
- If JAXB is on classpath and XML is Accept from client, then Spring will return an XML string

```
<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-core</artifactId>
   <version>2.0.4</version>
</dependency>
<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-databind</artifactId>
   <version>2.0.4</version>
</dependency>
```

```
@RequestMapping(value="/student/{user}", method = RequestMethod.GET)
@ResponseBody
public Student getStudentByUsername(@PathVariable String user,
        HttpServletRequest request,
        HttpServletResponse response) {

    Student student = studentService.getStudent(user);
    return student;
}
```

Supported Message Converters

This is the complete list of HttpMessageConverters set up by mvc:annotation-driven:

- ByteArrayHttpMessageConverter converts byte arrays.
- StringHttpMessageConverter converts strings.
- ResourceHttpMessageConverter converts to/from org.springframework.core.io.Resource for all media types.
- SourceHttpMessageConverter converts to/from a javax.xml.transform.Source.
- FormHttpMessageConverter converts form data to/from a MultiValueMap<String, String>.
- Jaxb2RootElementHttpMessageConverter converts Java objects to/from XML — added if JAXB2 is present on the classpath.
- MappingJackson2HttpMessageConverter (or MappingJacksonHttpMessageConverter) converts to/from JSON — added if Jackson 2 (or Jackson) is present on the classpath.
- ...

Making the POJO serialize

- Sometimes you want to change the name of the property on a POJO to something else
- Use @JsonProperty (“<name>”) for naming the property
- Use @XmlRootElement(name = "form")

```
@XmlRootElement(name = "form")
public class Form
{
        private String label;

        @JsonIgnore // Ignored during serialization to JSON
        private String periodType;

        @Deprecated
        private Boolean allowFuturePeriods;

        private List<Group> groups = new ArrayList<Group>();

        public Form() { }

        @JsonProperty
        public String getLabel()
        {
            return label;
        }
}
```

Consuming WS using Spring

- RestTemplate is the core class for client-side access to RESTful services

- HttpMessageConverter used to marshal objects into the HTTP request body and to unmarshal any response back into an object

- getForObject() will perform a GET, convert the HTTP response into an object type of your choice and return that object
- postForLocation() will do a POST, converting the given object into a HTTP request and return the response HTTP Location header where the newly created object can be found

Resources

- Spring MVC docs - http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html
- Spring REST Client docs - http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/remoting.html#rest-client-access