Restful Applications

What is a RESTful interface?

REST is a stateless client-server communication model. Using a combination of HTTP CRUD operations and the url path to easily describe the capabilities of a web service.

Example News Service

GET /news/ -> Fetches the top headlines for today.

GET /news/{date} -> Fetches headlines for a particular date.

GET /news/politics -> Fetches today's headlines in politics.

POST /news/politics/{id}/comment -> Posts a new comment to the article specified by id.

DELETE /news/politics/{id}/comment -> Deletes a comment from the article specified by id.

Format Examples

GET /news/politics -> Returns articles in the default XML format.

GET /news/politics/json -> Returns articles in JSON.

GET /news/politics/plain -> Returns articles in plain text.

Back to Servlets

Servlets are the HTTP workhorse of Java EE. They allow us to deal with HTTP via API calls rather than arbitrary string parsing. However, simply using servlets does not offer a easy way to implement a RESTful service.

REST Implementations

Like json parsers, REST implementations are not offered in the standard Java EE SDK.

JAX-RS

JAX-RS stands for Java API for XML-RESTful Services. There are many implementations of JAX-RS. In this class we will focus on Jersey. Others include RESTEasy, Apache Wink, and CXF.

Benefits of JAX-RS

JAX-RS enables Java POJO classes to be able to easily define their RESTful interface without needing to analysing the URL in a servlet.

Jersey Required Files

- https://jersey.java.net/

Back to JSON Processing

We've seen some basics on parsing objects into json. However our previous implementation (org.json) required some amount of coding (ie toJSONString).

JAXB

A new Java API called Java Architecture for XML Binding (JAXB) can make it easier to access XML documents from applications written in the Java programming language. In our class the implementation of JAXB will be *jackson JAXB.*

Jackson Required Files

- http://wiki.fasterxml.com/JacksonHome

Deeper into JAXB

So far we have only seen one annotation.

XmlRootElement(name = “rootElementname”)

XmlAccessorType

This annotation can be used to control what gets serialized in a class.

XmlAccessType.PUBLIC_MEMBER

public fields

annotated fields (@XmlAttribute)

properties

XmlAccessType.PROPERTY

annotated fields

properties

XmlAccessType.FIELD

fields

annotated properties

XmlAccessType.NONE

annotated fields

annotated properties

Ordering Elements

```
@XmlType(propOrder = {"fieldA", "fieldB"})
```

Results in fieldA occuring before fieldB.

Lists of Other Elements

@XmlElementWrapper(name = "randomMessage")

Specifies that this class wraps another serializable object.

JAX-RS Annotations

@POST

The analogue to @GET tells the server that the call being made is a POST.

@POST

@POST can be combined with @FormParam to enable including variables directly from an html form (ie the <form> tag).

Remember to use ‘application/x-www-form-urlencoded’ as the mime type.

Parameterized Paths

Remember @Path(“/something”)?

We can include parameters as well like id’s, names, etc

@Path("/xml/{id: [0-9]+}")

Turning URL Params Into Variables

(@PathParam("id") int id)

Results in turning the param in the @Path matched with ‘id’ into the input variable id.

The Response Object

The Response object is used as a general purpose return value for JAX-RS.

return Response.ok().build()

return Response.status(200).msg(json).build()

@Consumes

The analogue to @Produces. This specifies that any REST call for this method requires a specific mime type. Specifying ‘json’ or ‘xml’ will let you take an object as an input variable. If an unexpected value is given then status 415 is returned.

Additional Resources

https://docs.oracle.com/cd/E19776-01/820-4867/6nga7f5n6/index.html