Session 12

RESTful Services

Lecture Objectives

- Understand the fundamental concepts of Web services
- Become familiar with JAX-RS annotations
- Be able to build a simple Web service
Reading & References

- **Reading**
  - **Tutorials**
    - https://javabrains.io/courses/javaee_jaxrs/
    - docs.oracle.com/javaee/7/tutorial/webservices-intro.htm#GIJTI
    - (Chapters 27 and 29.1-29.3)

- **Reference**
  - **Java EE API**
    - docs.oracle.com/javaee/7/api/javax/ws/rs/package-summary.html
  - **Book**
    - RESTful Java Web Services, 3rd Edition,

---

Client - Servlet Model

- Requires logic in servlet to route each request to a service method
- Does not directly use URL and other http data to route to a service
- Mapping of the URL to a servlet is handled with web.xml or Java Annotation in servlet class

```html

Servlet identified by the “helloyou.html” URL string usually acts as a controller, and routes to a service handler
Client/Server Interaction

Our definition of client/server interaction to date

<table>
<thead>
<tr>
<th>HTML DDM</th>
<th>JavaScript functions</th>
<th>XMLHttpRequest objects</th>
<th>Servlets</th>
<th>Service Handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HTTP, TCP/IP</td>
</tr>
</tbody>
</table>

Parts of the client/server interaction are abstracted by tools/libraries

RESTful Web Services

- **Representational State Transfer**
- **Architectural style for distributed systems**
- **Architecturally consistent with http**
- **Provides a standard means of interoperating between software applications running on a variety of platforms and frameworks**
- **Use existing W3C and IETF standards (HTTP, XML, URI, MIME)**

A service is a software component provided through a network-accessible endpoint
Types of Web Services

- JAX-WS
  - Communication using XML
  - Provides for message-oriented and RPC services
  - Uses SOAP messages
  - includes standards for security and reliability

- JAX-RS
  - Standard
  - Semantics of the data to be exchanged is understood by client and server

JAX-RS

- Java API for RESTful Web Services
- A standard – not a product
- Reference implementations
  - Jersey, RESTeasy, et al, along with some application servers
  - No requirement to implement on top of servlets, but many implementation do
Client/Server Interaction

Think of JAX-RS as extending the abstraction to the Service handlers

Parts of the client/server interaction are abstracted by tools/libraries

Principles of REST Architectural Style

- Resource identification through URI
- Uniform interface - CRUD access defined in HTTP methods (PUT, GET, POST, and DELETE)
- Self-descriptive messages - content can be accessed in a variety of formats (e.g., HTML, XML, plain text, PDF, JPEG, JSON, etc.). Metadata about the resource is available
- Stateful interactions through links - Interactions are stateless (request messages contain state info)

CRUD = Create, Read, Update, and Delete
Implications of REST Style

- Interactions are predominantly computer-computer, not human-computer
- Resource based URI
- Typically published as an API, so design and URI naming important
- Expanded and more precise use of http methods
- Expanded use of http status codes
- Content negotiation between client and server

Example

- We start by building a very simple RESTful service
- In the next session, we will extend this by
  - Passing parameters to the server
  - Negotiating content
  - Returning content

For all the examples, think of accessing the resources from your html/JavaScript running in your browser

Hello, World!!
Creating a RESTful Root Resource Class

- Root resource classes are POJOs (plain old Java objects)
- Annotated with @Path or a request method designator (@GET, @PUT, @POST, or @DELETE)

JAX-RS uses Java Annotations

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@PATH</td>
<td>Relative URI indicating where the class will be hosted. Can also embed variables (e.g., /helloworld/{username})</td>
</tr>
<tr>
<td>@GET</td>
<td>Corresponds to the HTTP GET method. A Java method annotated with @GET will handle GET requests</td>
</tr>
<tr>
<td>@POST</td>
<td>Corresponds to the HTTP POST method. Intended for new resources.</td>
</tr>
<tr>
<td>@PUT</td>
<td>Corresponds to HTTP PUT method. Intended for resource updates</td>
</tr>
<tr>
<td>@DELETE</td>
<td>Corresponds to HTTP DELETE method</td>
</tr>
</tbody>
</table>
JAX-RS Annotation Summary

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@HEAD</td>
<td>Corresponds to HTTP Method.</td>
</tr>
<tr>
<td>@PathParam</td>
<td>Parameter extracted from the request URI. Parameter names correspond to the URI path template variable names specified in the @PATH annotation.</td>
</tr>
<tr>
<td>@QueryParam</td>
<td>Extracted from the query string.</td>
</tr>
<tr>
<td>@Consumes</td>
<td>Specifies the MIME type sent by client.</td>
</tr>
<tr>
<td>@Produces</td>
<td>Specifies the MIME type produced (e.g., “text/plain”)</td>
</tr>
<tr>
<td>@ApplicationPath</td>
<td>Defines the URL mapping. Base URI for all resource URIs specified by @Path.</td>
</tr>
</tbody>
</table>

Web Services With NetBeans ...

- Create a new project (or use an existing one)

  Choose Project

  Categories:
  - Java
  - JavaFX
  - Java Web
  - Java
  - HTM

  Name and Location
  - Project Name: CSE136-Services
  - Project Location: C:\Users\RobertKelly\Documents\NetBeansProjects\CSE136-Services
  - Project Folder: C:\Users\RobertKelly\Documents\NetBeansProjects\CSE136-Services
... Web Services With NetBeans ...

- Set libraries
- No need to declare any frameworks

Note that Glassfish includes a reference implementation of JAX-RS

![Server and Settings](image)

You are now set to define your first JAX-RS application

![Initial application folder](image)
NetBeans includes a feature to create a new RESTful Web Service.

Start to create a helloworld application with a right click on project.

Specify resource class.

NetBeans will set up with some starter code.
Session 12 – RESTful Services

ApplicationConfig Class

```java
package helloWorld;
import java.util.Set;
import javax.ws.rs.core.Application;
@javax.ws.rs.ApplicationPath("webresources")
public class ApplicationConfig extends Application {
    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> resources = new java.util.HashSet<>();
        addRestResourceClasses(resources);
        return resources;
    }
    // Do not modify addRestResourceClasses() method.
    // It is automatically populated with all resources defined in the project. You may comment out this method call in getClasses().
    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(helloWorld.HelloWorld.class);
    }
}
```

This config file is automatically generated by NetBeans.
The ApplicationPath serves as the base URL to locate the services.

You can also specify the base URI in the web.xml.

HelloWorld.java

```java
@Path("helloworld")
public class HelloWorld {
    @Context
    private UriInfo context;

    public HelloWorld() {
    }
    @GET
    @Produces(MediaType.TEXT_HTML)
    public String getHtml() {
        return "<html><body><h1>Hello, World!!</h1></body></html>";
    }
    @PUT
    @Consumes(MediaType.TEXT_HTML)
    public void putHtml(String content) {
    }
}
```

An http GET request will return the html.

Path relative to the URI path defined with ApplicationPath annotation.

Identifies the MIME type of the response.
(MediaType Class)

- javax.ws.rs.core.MediaType
- An abstraction for JAX-RS media types
- Contains String constants
- Examples
  - TEXT_HTML - "text/html"
  - TEST_PLAIN - "text/plain"
  - APPLICATION_JSON - "application/json"

Test the Web Service

- Start the application, then access your services through your browser
- Note the URL
  - Application (or project)
    - Specified in ApplicationConfig
  - Specified with @Path in HelloWorld class

Hello, World!!
Without a form, the http request is likely a GET
Test the Service in NetBeans

- Right click on service method.
- Response opens in default browser

Available Web Services

- Google Maps
- Flickr
- Yahoo News Search
- YouTube
- Facebook
- Lots more


The above URL will search (maybe?) the Web database for PDF files containing the term “finances”
Are We On Track?

- Verify that you can build a web service in your ISE
- Use the same example as shown in the slides

Assignment

- Use your Brooklyn Library html to make web service calls on a library resource
- Define a Java class whose data includes all the items in the Brooklyn Library form (plus a card # field).
- The data structure in the class should hold some collection of library card data
- When instantiated, the card data structure will initialize to 5 cards
- Write a GET service to retrieve card data when a user enters the card # and last name. If combo is found, returned data will populate the form.
- Write a POST service to add form data to the data structure when the Submit button is pressed
- Use a XMLHttpRequest to send the request to the service
Have You Achieved the Lecture Objectives?

- Understand the fundamental concepts of Web services
- Become familiar with JAX-RS annotations
- Be able to build a simple Web service