Session 15

RESTful Services

Part 3

Lecture Objectives

- Understand how to pass parameters from the URL to a Web service
- Understand how to return values from a Web service using the @Produces annotation
**Data Exchange**

- We define the data exchanged through annotation for Produces and Consumes
- Content format is negotiated by the client and server based on the annotation and the ability of each to handle various formats

---

**@Consumes**

- `@javax.ws.rs.Consumes`
- Defines the MIME type the class methods can accept
- Defined at either the class level or the method level
- Selected values
  - `application/json`
  - `application/octet-stream`
  - `text/html`
  - `text/plain`
  - `multipart/form-data`
  - `application/x-www-form-urlencoded`  
  
  Strings defined in `javax.ws.rs.MediaType`

  **Example**
  ```java
  @Consumes({MediaType.TEXT_PLAIN, MediaType.TEXT_HTML})
  ```

  Typical browser encoding of the form data set
@Produces

- @javax.ws.rs.Produces
- Defines the MIME type that a REST resource class method can return to the client
- Defined at either the class level (defaults for all methods) or method level
- Selected values
  - application/json
  - application/octet-stream
  - text/html
  - text/plain

Example

```java
@Produces("image/jpeg,image/png")
```

Path Templates

- A path can be defined with a path template – essentially a placeholder for a value to be defined by the user
- Parameter is obtained in the following example

```java
@Path("/users/{username}")
public class UserResource {
    @GET
    @Produces("text/html")
    public String getUser(@PathParam("username") String userName) {
        ...
    }
}
```
Example

In the QueryParameters example, we obtained the card number from the form data set

```java
public class LibrarycardsResource {
    ...
    @GET
    @Produces(MediaType.TEXT_HTML)
    public String getText(@QueryParam("cnum") int cardNumber)
```

Alternatively, we can obtain it directly from the URL with a call such as `localhost:8080/CSE336-Services/library/librarycards/124`

```java
@GET
@Produces(MediaType.TEXT_HTML)
public String getCard( @PathParam("cnum") int cardNumber) {
```

Web Resources Style

The PathParameter annotation provides a different style in requesting Web resources

Example

`localhost:8080/CSE336-Services/library/librarycards/124`

Made to appear as a data retrieval where the path (e.g., librarycards) appears as a plural data resource, and the path parameter (e.g., 124) appears as if it were an index in the repository for the data resource
@Produces Annotation

The above example returned html that displays as

```java
@GET
@Produces(MediaType.TEXT_HTML)
public String getCard(@PathParam("cnum") int cardNumber) {
    String s1 = "<html><body><h1>";
    String s2 = "</h1></body></html>";
    String message = "";
    if (cardNumber==123){
        message="{"num:123, nickname:'Alonzo' type:'Adult'}";
        return s1+message+s2; }
    else {return s1 + "Would you like to apply for a library card?" + s2; }
}
```

@Produces Example

If we change the @Produces annotation, the response is not evaluated as html, and only appears as plain text

```java
@GET
@Produces(MediaType.TEXT_Plain)
public String getCard(@PathParam("cnum") int cardNumber) {

    <h1>{num:123, nickname:'Alonzo', type:'Adult'}</h1>
```
@Produces Example

If we again change the @Produces annotation, when called with
localhost:8080/CSE336-Services/library/librarycards/125
it returns the JSON string

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public String getCard(@PathParam("cnum") int cardNumber) {
    String s1 = "<html><body><h1>";
    String s2 = "</h1></body></html>";
    String message="{num:123, nickname:'Alonzo' type:'Adult'}";
    if (cardNumber==123){
        return s1+message+s2;
    } else {
        return message;
    }
}
```

How Do We Deal With the Same Origin Policy?

In JAX-RS, you add a class that sets the allow headers, as in

```
import javax.ws.rs.container.*;
import javax.ws.rs.ext.Provider;
import java.io.IOException;
@Provider
public class CORSFilter implements ContainerResponseFilter {
    @Override
    public void filter(ContainerRequestContext requestContext,
            ContainerResponseContext cres) throws IOException {
        cres.getHeaders().add("Access-Control-Allow-Origin", "*");
        cres.getHeaders().add("Access-Control-Allow-Headers", "origin, content-type, accept, authorization");
        cres.getHeaders().add("Access-Control-Allow-Credentials", "true");
        cres.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS, HEAD");
        cres.getHeaders().add("Access-Control-Max-Age", "1209600");
    }
}
```
Example - Ajax Access - Web Services ...

- User enters a card #, and tabs out of the field
- Onblur event causes a request to the Web service
- Response is a JSON string containing the values of the component fields
- The string is parsed, and name field updated to show the name

function requestCard() {
    c = document.getElementById("cnum").value;
    var url = "http://localhost:8080/CSE336-Services/library/librarycards/"+c;
    req = new XMLHttpRequest();
    req.open("GET", url, true);
    req.onreadystatechange = update;
    req.send(null);
}
...
<input type="text" id="cnum" name="cnum" onblur="requestCard()" />
...
<input type="text" id="nickname" name="nickname" maxlength="12" />

... Example - Ajax Access - Web Services ...

Note the path parameter approach
... Example - Ajax Access - Web Services ...

```java
@GET
@Produces(MediaType.TEXT_PLAIN)
public String getCard(@PathParam("cnum") int cardNumber) {
    String message="{"num":123, "nickname":"Alonzo", "type":"Adult"}";
    if (cardNumber==123){
        return message;
    } else {
        return "";
    }
}
```

Remember. Single quote in Java defines a char literal, so we need to escape the nested double quote.

---

```javascript
function update() {
    n = document.getElementById("nickname");
    if (req.readyState==4 && req.status==200){
        var obj = JSON.parse(req.responseText);
        n.value=obj.nickname;
    }
}
```

Be careful with your single and double quotes.

update is the XMLHttpRequest callback function.

---

Library Card Application

Complete this application and click the Submit button. You will be redirected to the library site to complete your registration.

* Required

<table>
<thead>
<tr>
<th>Library Card</th>
<th>Card Number (3 digits)</th>
<th>Name</th>
<th>Card Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123</td>
<td>Alonzo</td>
<td>Young Adults (Ages 13 - 16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Adult (Ages 17 and over)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Seniors (Ages 62 and over)</td>
</tr>
</tbody>
</table>
Assignment 5 – Part a

- 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).
- Your service should be able to access a collection (minimum of 5 cards) of the card data (serialized in some appropriate format to a file).
- When the user hits submit after entering only the card # and last name, if combo is found in server, return data just as a string of Json.

Have You Achieved the Lecture Objectives?

- Understand how to pass parameters from the URL to a Web service.
- Understand how to return values from a Web service using the @Produces annotation.