Content Negotiation

Sang Shin
JPassion.com
"Code with Passion!"

Topics

- HTTP media types
- @Produces and @Consumes
- Content handler
- Content type selection by client

HTTP Media Types (Formats)

Formats in HTTP

Request

```
GET /music/artists/beatles/recordings HTTP/1.1
Host: media.example.com
Accept: application/xml
```

Format

Response

```
HTTP/1.1 200 OK
Date: Tue, 08 May 2007 16:41:58 GMT
Server: Apache/1.3.6
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0"?>
<recordings xmlns="…">
  <recording>…</recording>
  …
</recordings>
```

State transfer

Representation

Media Types

- Client and server can send and receive data in multiple media types
- Format specification + Parsing rules
  - text/html – regular web page
  - application/json
  - application/xhtml+xml
  - application/rss+xml
  - application/rdf+xml
- You can create your own
  - application/foo+json
  - application/foo+json;application

HTTP Headers for Media Types

- How does a client specify acceptable response formats?
  - Through 'Accept' HTTP request header
- How does a client specify its request format?
  - Through 'Content-Type' HTTP request header
- How does a server specify the response format?
  - Through 'Content-Type' HTTP response header

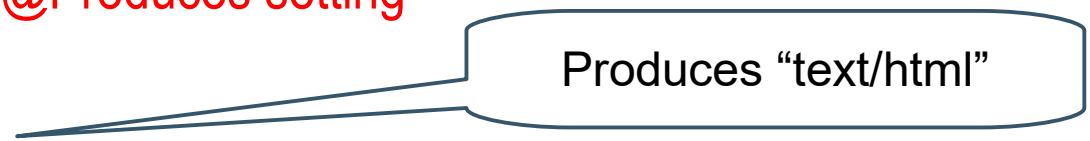
@Produces & @Consumes

@Produces

- Used to specify the media types of a response a resource can produce and send back to the client
- Can be applied at both the class and method levels
  - Method level overrides class level

Example: @Produces

```
@Path("/myResource")
@Produces("text/plain")
public class SomeResource {
   // defaults to the media type of the @Produces
   // annotation at the class level - “test/plain”
    @GET
    public String doGetAsPlainText() {
        ...
    }
    // overrides the class-level @Produces setting
    @GET
    @Produces("text/html")
    public String doGetAsHtml() {
        ...
    }
}
```



Choice of Media Type Based on Client Preference ("Accept" header)

- If a resource class is capable of producing more that one media type, then the resource method chosen will produce the most acceptable media type as declared by the client.
  - Accept header of the HTTP request
- For example, using the example code in previous slide
  - *Accept: text/plain* - *doGetAsPlainText()* method will be invoked
  - *Accept: text/plain;q=0.9, text/html* - *doGetAsHtml()* method will be invoked
- If no methods in a resource are able to produce the media type in a client request, the JAX-RS runtime sends back an HTTP "406 Not Acceptable" error

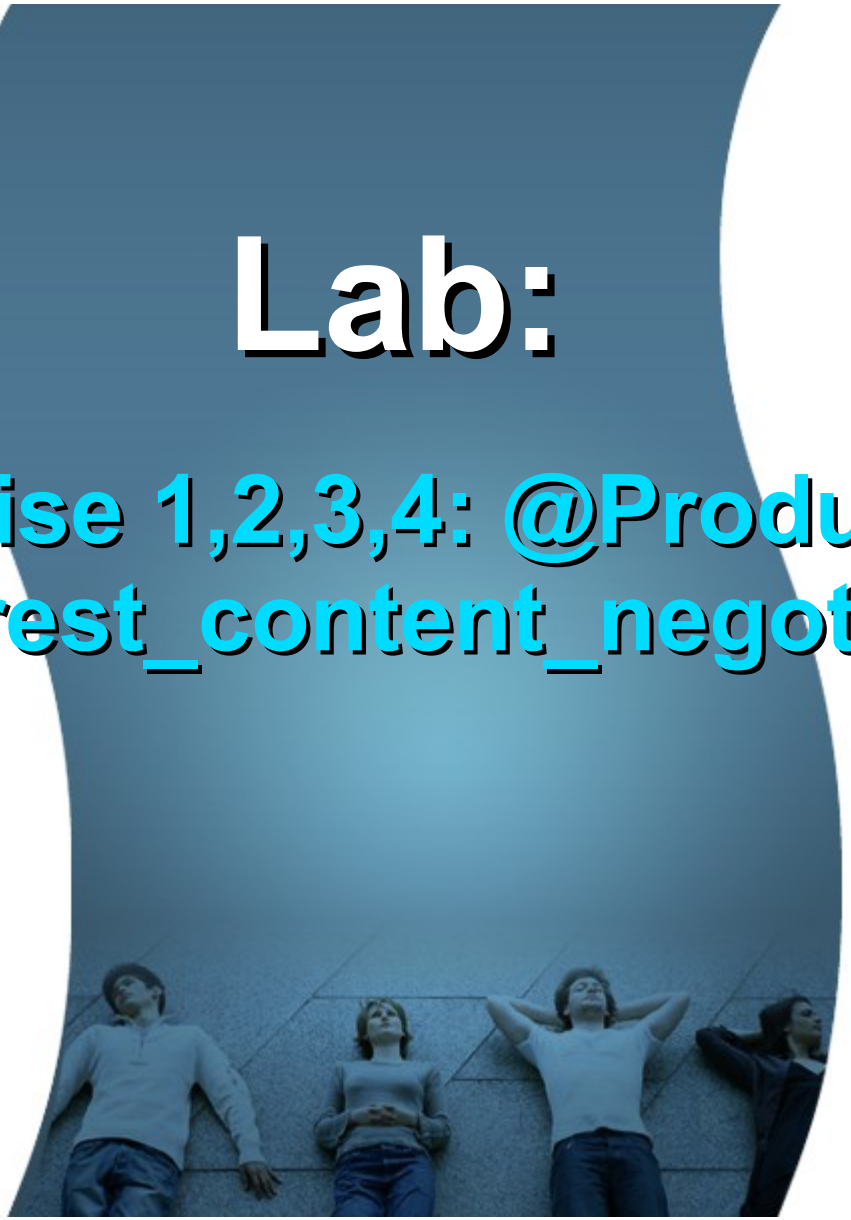
Server-side Content Negotiation

```
// The doGetAsXmlOrJson method will get invoked if either of the media types "application/xml"
// and "application/json" are acceptable. If both are equally acceptable then the former will be
// chosen because it occurs first.
@GET
@Produces({"application/xml", "application/json"})
public String doGetAsXmlOrJson() {
    ...
}
```

```
// if client accepts both "application/xml" and "application/json" (equally), then
// a server always sends "application/json", since "application/xml" has a lower quality factor.
@GET
@Produces({"application/xml; qs=0.9", "application/json"})
public String doGetAsXmlOrJson() {
    ...
}
```

Lab:

Exercise 1,2,3,4: @Produces
4367_javarest_content_negotiation.zip



@Consumes

- Used to specify the media types of representations a resource can consume
- Can be applied at both the class and method levels
  - Method level override a class level
- A container is responsible for ensuring that the method invoked is capable of consuming the media type of the HTTP request entity body, which is specified by 'Content-type' HTTP request header
  - If no such method is available, the JAX-RS runtime responds back with a HTTP "415 Unsupported Media Type"

@Consumes

```
@POST
// Consume representations identified by the MIME media type "text/plain".
@Consumes("text/plain")
public void consumeMessage(String message) {
    // Do something with the message
}
```

Lab:

Exercise 5: @Produces & @Consumes
4367_javarest_content_negotiation.zip

Content Handler

What is Content Handler?

- Content handler is responsible for marshalling and unmarshalling the message body to and from Java object
- JAX-RS provides built-in content handlers for basic types
  - String, char[]
  - byte[]
- JAX-RS also provides built-in content handlers for XML
  - XML content handler via JAXB
- Content handler for JSON is provided by 3rd-party library
  - Jackson or MOXy
- Custom content handler can be plugged in

Content Type Selection By Client

Content Negotiation schemes

- Accept header
- URL extension
- Request parameter

URL Extension Example #1

- Client
  - http://.../resources/customers/1.xml
  - http://.../resources/customers/1.json
- Server

```
@Path("{resourceID}.xml")
@GET
public Response getResourceInXML(@PathParam("resourceID") int resourceID) {
   return Response.ok(users.get(resourceID)).
          type(MediaType.APPLICATION_XML).
          build();
}
@Path("{resourceID}.json")
@GET
public Response getResourceInJSON(@PathParam("resourceID") int resourceID) {;
   return Response.ok(users.get(resourceID)).
          type(MediaType.APPLICATION_JSON).
          build();
}
```

URL Extension Example #2

- Client
  - http://.../resources/customers/1.xml
  - http://.../resources/customers/1.json
- Server

```
@Path("{resourceID}.{type}")
@GET
public Response getResource(@PathParam("resourceID") int resourceID, @PathParam("type") String
type) {
    if ("xml".equals(type)) {
        return Response.ok(users.get(resourceID)).
                type(MediaType.APPLICATION_XML).
                build();
    } else if ("json".equals(type)) {
        return Response.ok(users.get(resourceID)).
                type(MediaType.APPLICATION_JSON).
                build();
    }
    return Response.status(Response.Status.NOT_FOUND).build();
}
```

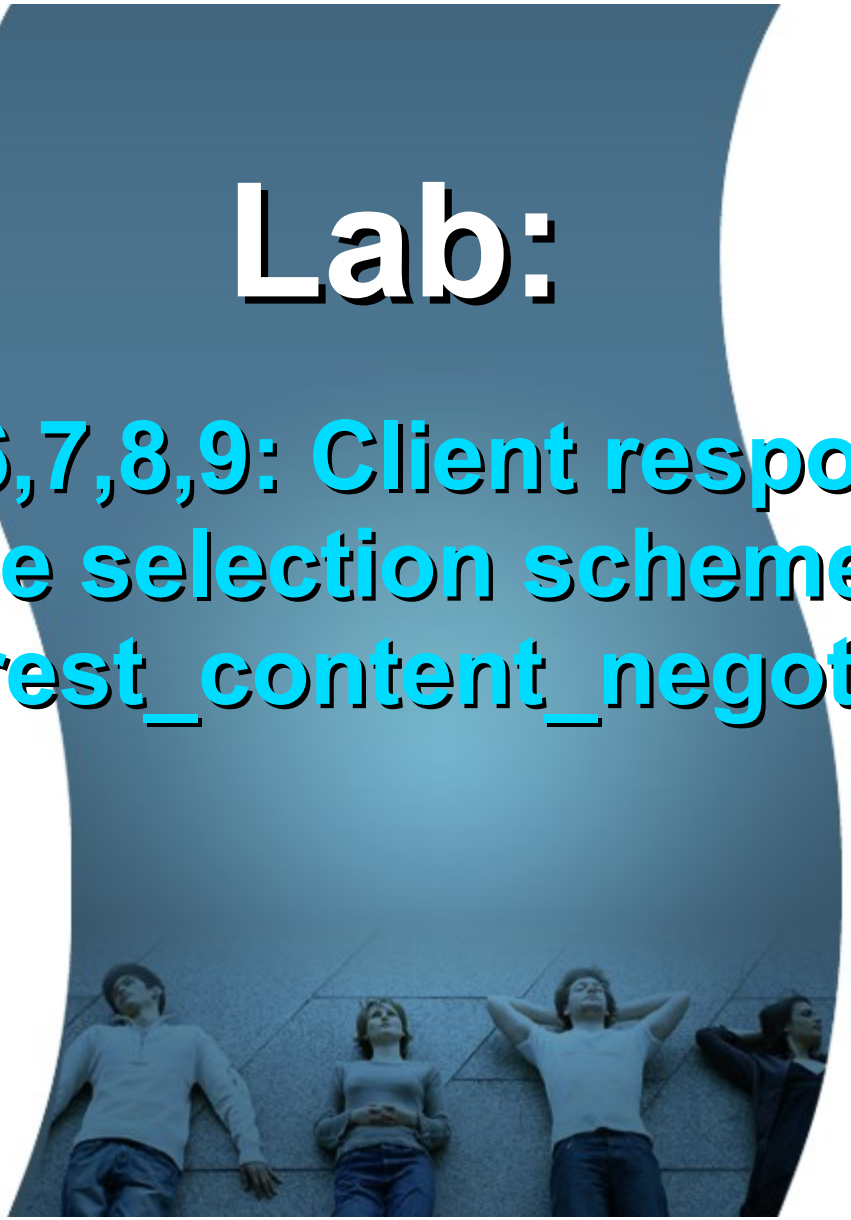
Using Request Parameter Example

- Client
  - http://.../resources/customers/1?format=xml
  - http://.../resources/customers/1?format=json
- Server

```
@Path("{resourceID}")
@GET
public Response getResource(@PathParam("resourceID") int resourceID,
        @QueryParam("format") String format) {
    if (format == null || "xml".equals(format)) {
        return Response.ok(users.get(resourceID)).
                type(MediaType.APPLICATION_XML).
                build();
    }
    else if ("json".equals(format)) {
        return Response.ok(users.get(resourceID)).
                type(MediaType.APPLICATION_JSON).
                build();
```

Lab:

Exercise 6,7,8,9: Client response data type selection schemes 4367_javarest_content_negotiation.zip



Code with Passion!
JPassion.com