**Using @Consumes and @Produces to Customize Requests and Responses**

The information sent to a resource and then passed back to the client is specified as a MIME media type in the headers of an HTTP request or response. You can specify which MIME media types of representations a resource can respond to or produce by using the following annotations:

* javax.ws.rs.Consumes
* javax.ws.rs.Produces

By default, a resource class can respond to and produce all MIME media types of representations specified in the HTTP request and response headers.

**The @Produces Annotation**

The @Produces annotation is used to specify the MIME media types or representations a resource can produce and send back to the client. If @Produces is applied at the class level, all the methods in a resource can produce the specified MIME types by default. If applied at the method level, the annotation overrides any @Produces annotations applied at the class level.

If no methods in a resource are able to produce the MIME type in a client request, the JAX-RS runtime sends back an HTTP “406 Not Acceptable” error.

The value of @Produces is an array of String of MIME types. For example:

@Produces({"image/jpeg,image/png"})

The following example shows how to apply @Produces at both the class and method levels:

@Path("/myResource")

@Produces("text/plain")

public class SomeResource {

@GET

public String doGetAsPlainText() {

...

}

@GET

@Produces("text/html")

public String doGetAsHtml() {

...

}

}

The doGetAsPlainText method defaults to the MIME media type of the @Produces annotation at the class level. The doGetAsHtml method’s @Produces annotation overrides the class-level @Produces setting and specifies that the method can produce HTML rather than plain text.

If a resource class is capable of producing more than one MIME media type, the resource method chosen will correspond to the most acceptable media type as declared by the client. More specifically, the Accept header of the HTTP request declares what is most acceptable. For example, if the Accept header is Accept: text/plain, the doGetAsPlainText method will be invoked. Alternatively, if the Accept header is Accept: text/plain;q=0.9, text/html, which declares that the client can accept media types of text/plain and text/html but prefers the latter, the doGetAsHtml method will be invoked.

More than one media type may be declared in the same @Produces declaration. The following code example shows how this is done:

@Produces({"application/xml", "application/json"})

public String doGetAsXmlOrJson() {

...

}

The doGetAsXmlOrJson method will get invoked if either of the media types application/xml and application/json is acceptable. If both are equally acceptable, the former will be chosen because it occurs first. The preceding examples refer explicitly to MIME media types for clarity. It is possible to refer to constant values, which may reduce typographical errors. For more information, see the constant field values of MediaType at <https://jsr311.dev.java.net/nonav/releases/1.0/javax/ws/rs/core/MediaType.html>.

**The @Consumes Annotation**

The @Consumes annotation is used to specify which MIME media types of representations a resource can accept, or consume, from the client. If @Consumes is applied at the class level, all the response methods accept the specified MIME types by default. If applied at the method level, @Consumes overrides any @Consumes annotations applied at the class level.

If a resource is unable to consume the MIME type of a client request, the JAX-RS runtime sends back an HTTP 415 (“Unsupported Media Type”) error.

The value of @Consumes is an array of String of acceptable MIME types. For example:

@Consumes({"text/plain,text/html"})

The following example shows how to apply @Consumes at both the class and method levels:

@Path("/myResource")

@Consumes("multipart/related")

public class SomeResource {

@POST

public String doPost(MimeMultipart mimeMultipartData) {

...

}

@POST

@Consumes("application/x-www-form-urlencoded")

public String doPost2(FormURLEncodedProperties formData) {

...

}

}

The doPost method defaults to the MIME media type of the @Consumes annotation at the class level. The doPost2 method overrides the class level @Consumes annotation to specify that it can accept URL-encoded form data.

If no resource methods can respond to the requested MIME type, an HTTP 415 (“Unsupported Media Type”) error is returned to the client.

The HelloWorld example discussed previously in this section can be modified to set the message by using @Consumes, as shown in the following code example:

@POST

@Consumes("text/plain")

public void postClichedMessage(String message) {

// Store the message

}

In this example, the Java method will consume representations identified by the MIME media type text/plain. Note that the resource method returns void. This means that no representation is returned and that a response with a status code of HTTP 204 (“No Content”) will be returned.