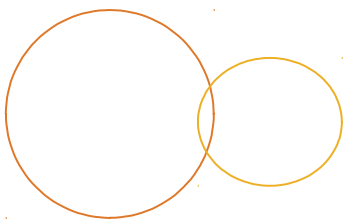
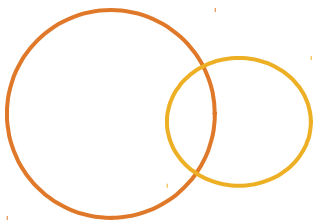
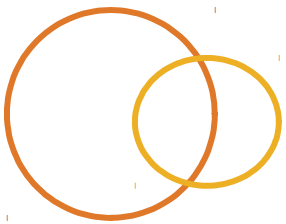
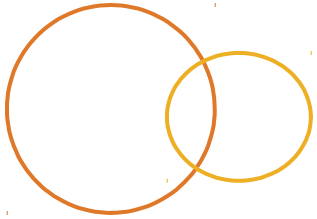
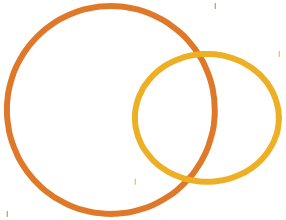




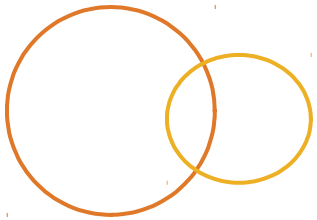

Handling HTTP entities

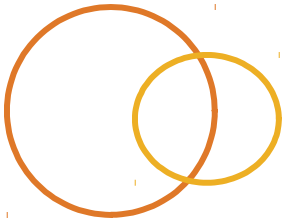




Objectives

- Give an overview of the role of MessageBodyReader and MessageBodyWriter classes in conversion of HTTP entity bodies to and from Java objects.
- Give an overview of the installation of MessageBodyReader/Writer Provider objects into a JAX-RS system to enable conversion of a particular MIME data type to and from Java objects
- Send data structures to a client using JSON format
- Receive JSON data structures from a client
- Use the @Produces annotation to inform the JAX-RS dispatch/routing infrastructure that a particular service method is appropriate for returning a given MIME data type to the caller

Objectives





- Give an overview of how JAX-RS can return binary data to a client using either an InputStream, a byte [], or a StreamingOutput
- State the limitation of JAX-RS 2.x with respect to handling multipart/form-data
- State the significance of charset for web-based operations involving text and textual representations of structured data
- Locate documentation for the Jackson conversion libraries

REST And Structured Entities

- REST services often send and receive entities representing structured data in textual forms
- Often done using XML or JSON
  - JSON is becoming more popular
- Intuit's standards call for "JSON first"

JAX-RS And Structured Entities

- JAX-RS expects to convert Java objects to and from text-based wire formats
  - Generally, no manual intervention is needed
- It does this using special classes annotated @Provider implementing MessageBodyReader and MessageBodyWriter interfaces
  - These can be programmer defined (but not often)
  - Reader/Writer for XML are required as part of the JAX-RS specification
  - JSON converters are readily available, distributed with some implementations, but might need to be enabled

Supplying The Provider

- The Provider class(es) that implement MessageBodyReader/Writer must be known to the JAX-RS infrastructure
  - Check the documentation for your particular implementation
  - Package scanning
  - Explicit classes in web.xml
  - Pure annotations in Servlets 3.x containers
  - Set<Class<?>> getClasses() in Application
  - Set<Object> getSingletons() in Application

The “Jackson” Provider

- The “Jackson” JSON support provider is preconfigured in TCRS
- More information on Jackson may be found at:

```
https://github.com/FasterXML/jackson

https://github.com/FasterXML/jackson-docs
```

- Older Jackson releases were at codehaus.org

Giving Permission For Conversions

- For a response entity to be converted to a target type (such as JSON):
  - The client should have indicated that it can handle the target
    - This is done by the client sending the Accept: header, though if absent, this implies Accept: */*
  - The service method should be annotated to indicate that the programmer permits this conversion
    - E.g. @Produces({MediaType.APPLICATION_JSON})
  - JAX-RS must have a suitable Provider that can convert from the Java class of the entity provided by the service method, and the desired target type

Giving Permission For Conversions

- For an entity from the client to be converted to a target type:
  - The client ***must*** have indicated the target type
    - This is done by the client sending the Content-type:
  - The service method should be annotated to indicate that the programmer permits this conversion
    - E.g. @Consumes(“application/json”)
  - JAX-RS must have a suitable Provider that can convert from the Java class of the entity provided by the service method, and the desired target type

Sending A Structured Response

```
@GET @Path("/one")
@Produces({MediaType.APPLICATION_XML,
           MediaType.APPLICATION_JSON})
public Response getOne() {
  DataTO dto = new DataTO(99, “banana”);
  return Response.ok(dto).build();
}

public class DataTO {
  public Integer value; public String fruit;
  public DataTO(int v, String f) {
    value = v; fruit = f;
  }
}
```

Receiving Structured Data

```
@POST @Path("/one")
@Consumes({ MediaType.APPLICATION_XML,
            MediaType.APPLICATION_JSON})
public Response getOne(DataTO theData) {
  long primaryKey = DataTO.store(theData);
  return Response.ok(“Success”)
    .status(Response.Status.CREATED)
    .header(“Location”, “” + primaryKey);
    .build();
}
```

Helping / Hinting Converters

- Different conversion providers might require hints for making some conversions
  - XML conversion is built into JAX-RS, as a specification requirement, using JAX-B
  - JAX-B uses annotations to provide help/hints on conversion
  - Handling lists / arrays is not always transparent
  - Jackson (used in TCRS) also offers some annotations e.g. @JsonIgnore, @JsonProperty (see Jackson docs)
  - Handling missing fields might vary between converters

Helping / Hinting JAX-B

- Minimal JAX-B provisions:
  - Annotate the class @XmlRootElement
  - Provide a zero argument constructor
  - Provide public fields ***or*** public get/set method pairs (***not*** both!)

What's In A `null` Field?

- In a PUT (update) request, it's normal to treat a null in the input structure as "don't modify this element"
  - Sometimes, null means "set this to null"—this problem might be better addressed using PATCH
  - Omitting fields can reduce bandwidth usage
- Primitive fields in Java objects cannot be omitted, but wrappers (Integer, Double, etc.) can have null values

Handling Repeating Data

- Jackson handles T, List<T>, and T[] without help for JSON conversion for input and output
- JAX-B handles T[] for both input and output if it handles T
- JAX-B can also convert a sequence of T into List<T> for input
- JAX-B fails if given List<T> directly for output
  - Can you avoid List, and simply use an array?
  - Create a wrapper or “Transfer Object” that contains List<T> as a field (perhaps surprisingly, this works fine)
  - Could also use javax.ws.rs.core.GenericEntity

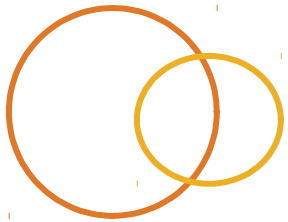
Transfer Objects

- Because of the differing needs of wire-transfer and business entity representation / validation, it's often a good idea to create specific transfer object or T.O.
  - Annotations about wire transfer are not placed on business entity objects
  - Validation is undesirable—this should be a responsibility of the domain entity
  - Fields can be public if desired
  - Provide utility methods in the T.O. class to create T.O. from domain entity, and vice-versa, and to modify a domain entity based on non-null T.O. fields

Other Data Types

- JAX-RS supports additional entity body types including:
  - InputStream—the stream is read and fed to the client
  - byte[]—the contents of the byte array are sent directly to the client
  - StreamingOutput—return an implementation of this interface and JAX-RS will call the implemented method
  - void write(OutputStream output)
  - allowing you to use the OutputStream to write data

Multipart Form Data



- JAX-RS 2 (the standard) does not provide a mechanism for directly handling multipart form data input to service methods
  - Multipart can be useful for uploading arbitrary binary data, such as images, though it is mostly convenient when using an HTML based client
- Implementations, including Jersey, provide implementation-specific extensions for this
  - Of course, these make your code implementation specific

Charset

- To ensure that the client handles international language text consistently
  - Ensure you're working with UTF-8 in your server
  - Arrange that your service method declares

```
@Produces(MediaType.APPLICATION_JSON
                + ";charset=UTF-8")
```

  to inform the client what is being sent

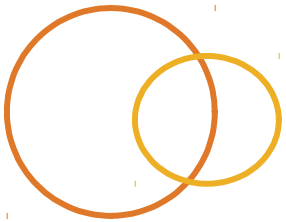
Lab Exercise

- Create a class to act as a Data Transfer Object
  - Give it fields:
    String name
    Integer count
    String [] more
  - Give it a constructor to initialize the fields
  - Give it a toString method to allow readable display of the object

Lab Exercise

- Create a service endpoint on the URI /structure
  - Respond to a GET request by returning one of these structures, with a status of 200 and a header named “x-structure” with the value “yes”
  - Respond to a POST request that receives one of these structures:
    - Print out the contents received on the console output
    - Return a simple text form of the same object that was received

Lab Exercise



- Ensure that the service offers and accepts both JSON and XML formats for this structure
- Observe the effect of a null field when responding to a GET
- Observe the effect of a missing element from the JSON format