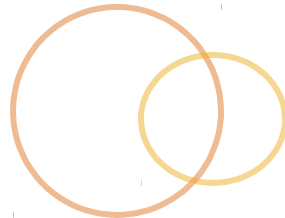
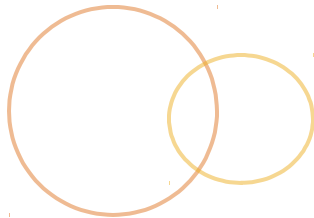
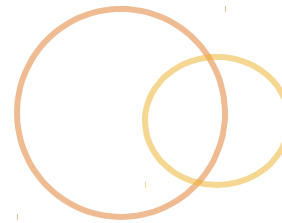
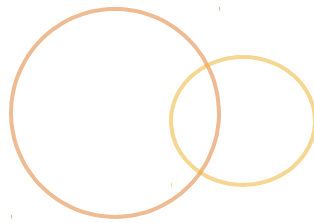


Core JAX-RS Features



Objectives

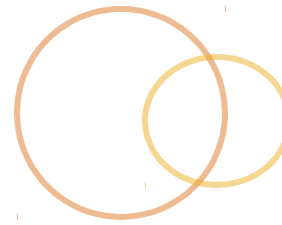


- ② Write a JAX-RS service method that uses an `@Path` annotation to define a JAX-RS variable and injects the String value of that variable into a method using `@PathParam`
- ② Write a JAX-RS service method that receives a query parameter using the annotation `@QueryParam`
- ② Write a JAX-RS service method that receives a header information using the annotation `@HeaderParam`
- ② Use the annotation `@DefaultValue` to inject non-null values for query parameters and/or headers that are absent from the request

Objectives

- 🕒 State the requirements for a Java data type to be convertible by JAX-RS prior to injection using `@PathParam`, `@QueryParam`, `@HeaderParam`
- 🕒 Write a JAX-RS service method that responds to a DELETE request
- 🕒 Use a Java Regular Expression in an `@Path` annotation to restrict the URIs that JAX-RS will route to a given service method
- 🕒 Give an overview of the mechanism by which JAX-RS matches URI and HTTP method of an incoming request to annotations on service classes and methods to select the service method to invoke

Path Parameters



- REST often selects an item by a primary key embedded in the URI
 - GET /customers/1234
- Part of the URI must be treated as a variable, even though that part still participates in routing
- JAX-RS defines and injects these like this:

```
@GET
```

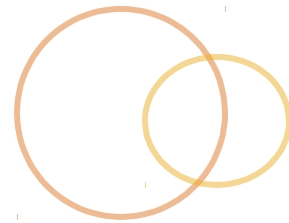
```
@Path("/{id}")
```

```
public String getOneCustomer(
```

```
    @PathParam("id") String customerPK) {
```

```
    ...
```

Query Parameters



- ◎ Query parameters can be injected into service methods too
- ◎ To pick up .../stuff?name=Fred+Jones

```
@Path("/stuff")
```

```
@GET
```

```
public String(
```

```
    @QueryParam("name") String name) {
```

```
    ...
```

Header Injection

- ◎ Headers in the request can be injected too
- ◎ To read the value of the header `intuit-tid`

```
@Path("/stuff")  
@GET  
public String getStuff(  
    @HeaderParam("intuit-tid")  
    String transactionId) {  
    ...  
}
```

More About Parameter Injection



- ◎ JAX-RS can inject more data too, including cookies and form data
- ◎ If a parameter is missing, null is injected
- ◎ If desired, missing parameters can be set to a specific String value by default

```
public String doStuff(  
    @QueryParam("name")  
    @DefaultValue("John Doe")  
    String name) {
```

Injecting Non-String Types



- Injecting a String allows control of the conversion and recognition of missing data
- Alternatively some types can be converted and injected directly by JAX-RS
- `doX(@QueryParam("count") Integer count)`
`{ ...`
- Rules govern which types can be injected

Injection / Conversion Rules



- ⦿ In addition to String, a type X can be converted and injected if:
 - ⦿ It's a primitive
 - ⦿ It's a wrapper of a primitive
 - ⦿ If X contains:
 - ⦿ A constructor taking a single String argument
 - ⦿ static X fromString(String s)
 - ⦿ static X valueOf(String s)
 - ⦿ It's a List<Y>, Set<Y>, or SortedSet<Y> that would be acceptable individually

Other HTTP Methods



- ⦿ JAX-RS provides several annotations that can be applied to a method, identifying it as a potential target for that HTTP method
- ⦿ @GET
- ⦿ @POST
- ⦿ @PUT
- ⦿ @DELETE
- ⦿ @OPTIONS
- ⦿ @HEAD
- ⦿ A service method requires ***exactly one*** of these

Restricting @Path Variable Matching



- ◎ The annotation `@Path("/{id}")` matches any one segment
 - ◎ Any number of characters, but not including "/" itself
- ◎ JAX-RS allows restricting the match using standard Java regular expressions
- ◎ E.g. to match only digits:
`@Path("/{id: \\d+}")`
- ◎ If the regular expression doesn't match, JAX-RS looks elsewhere for a potential service method

Overlapping @Path Specifications



- Regular expressions might “overlap” literal ones
- Two regular expressions might both match the same input string

```
@Path(“/banana”)
```

```
@Path(“/{f: [ban]}+”)
```

```
@Path(“/{f: \\p{Lower}+}”)
```

```
@Path(“/{f: [ban]}+”)
```

Overlapping @Path Specifications



- ⦿ JAX-RS 2.x always prefers the literal match
- ⦿ Conflicting literals should fail to deploy
 - ⦿ Or at least issue warnings
- ⦿ Selecting between two overlapping regular expressions is unpredictable
 - ⦿ They might overlap only for some input data
- ⦿ All these conflicts probably reflect poor API design
 - ⦿ They're potentially very confusing for client programmer

More Complex @Path Matches

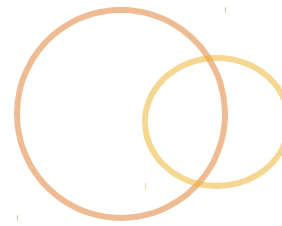


- An @Path annotation can contain multiple variables, with regular expressions
- An @Path annotation can contain multiple segments
- Regular expressions can span segments

```
@Path("/customers/{id: \\d\\d-\\d\\d\\d}"  
      + "/suppliers/{idx: \\d+}")
```

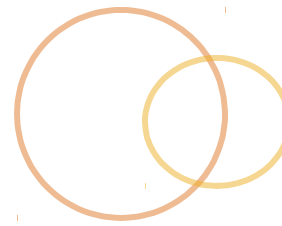
```
@Path("/documents/{path: [a-zA-Z0-9/]+}")
```

JAX-RS Routing



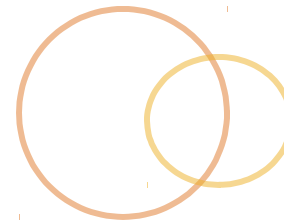
- Given an input request URI and HTTP method, JAX-RS selects which method to invoke by:
 - Looking for a class annotated `@Path("/blah")` where `/blah` (possibly multi-segment) matches the beginning of the URI
 - This might produce more than one class, but it's probably bad design if it does
 - Looking for methods that match the remaining path, and have the right HTTP method annotation (`@GET`, etc.)
 - If one is a literal match on `@Path` use that
 - If there's no literal match but multiple regex matches, use the one that has the most literal match points

JAX-RS Routing



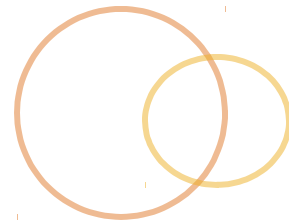
- Other factors will affect this later, including the type of data being sent, or requested
 - HTTP headers “Content-type” and “Accept”
- We’ll see these later
- Note that query parameters do ***not*** affect routing
 - Create a single method, and test if the query parameter is injected as null

Lab Exercise



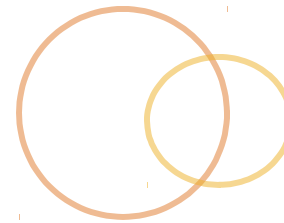
- Create a service endpoint that responds to a GET request on a URI of this form:
.../fruits?color=red
and returns the name fruit of the color specified
- Modify the service so that if the color is specified as “any”, one fruit is chosen at random.
- Arrange that if no color query parameter is specified the services responds as if “any” were specified

Lab Exercise



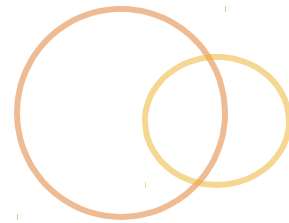
- Create a service endpoint that responds to a GET request on a URI of this form:
.../fruits/3
where the “3” can be any single digit. Return the name of a fruit taken from a small array (ten or fewer fruits)
- Arrange that this service checks the header “accept-language”. If this has the value “fr” report the fruit name in French, otherwise in English

Lab Exercise



- Create a service endpoint that responds to a DELETE request on a URI of this form:
.../fruits/3
where the “3” can be a sequence of digits.
Return the message “Deleting fruit: xxx” where xxx is the name of the fruit with the index specified by the digits in the URI

Lab Exercise



- You might find this helpful (define your own compatible Fruit class):

```
Fruit [] fruits = {  
    new Fruit("raspberry", "framboise", "red"),  
    new Fruit("orange", "orange", "orange"),  
    new Fruit("lemon", "citron", "yellow"),  
    new Fruit("lime", "citron vert", "green"),  
    new Fruit("blueberry", "myrtille", "blue"),  
    new Fruit("blackberry", "mûre", "black"),  
};
```