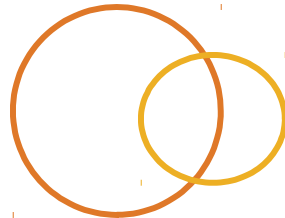
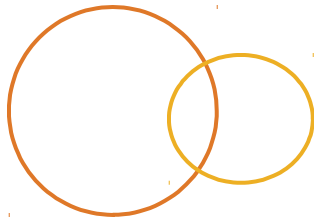
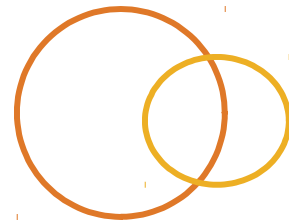
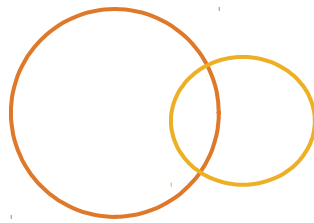


JAX-RS 2.x Client API

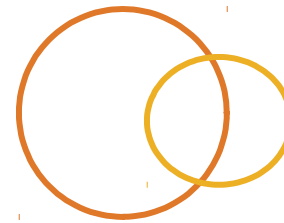


Objectives

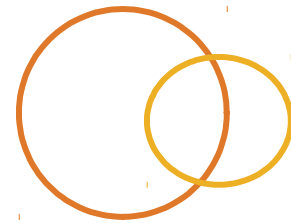


- ◉ Send a GET, PUT, POST, PATCH, or DELETE request and get a Response object
- ◉ Control the “Accept:” MIME type of a request
- ◉ Send an entity body with a request and specify the content-type to which should be converted
- ◉ Read the response status from a response
- ◉ Read headers from a response
- ◉ Read the returned Entity from a response object, including converting structured entities to Java objects from JSON

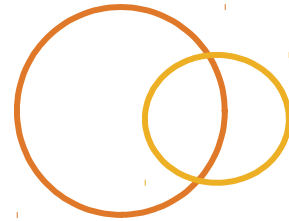
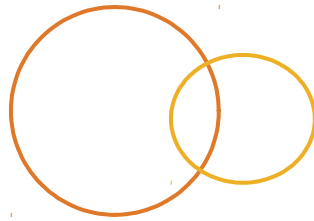
JAX-RS 2 Client



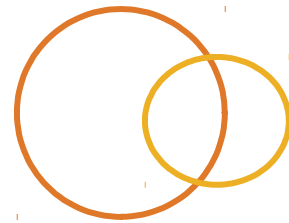
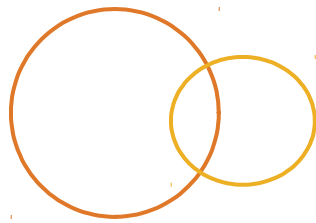
- ◉ JAX-RS 2 client is designed for fluent style programming
 - ◉ “Object flow” has some variations, but the basic form is:
 - ◉ ClientBuilder
 - ◉ Client
 - ◉ WebTarget
 - ◉ InvocationBuilder
 - ◉ Invocation
 - ◉ Response
- ← ◉ Entity, provided to Invocation / Invocation.Builder



- ◉ The ClientBuilder is obtained through a static factory
 - ◉ `ClientBuilder.newBuilder()`
- ◉ ClientBuilder is generally used to configure SSL/TLS related features, such as key/trust store
- ◉ `ClientBuilder.newClient` provides the Client object

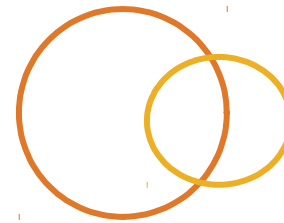


- The Client object is typically used to configure filters
- Client is then used to create one or more WebTarget objects



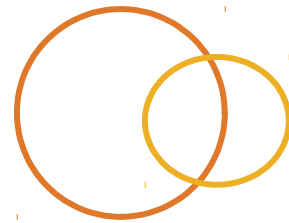
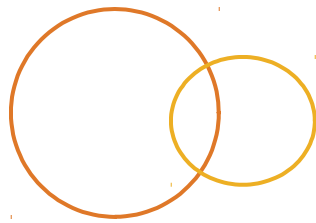
- ◉ WebTarget is used to describe a URL and can be used to create other WebTargets
 - ◉ In this sense, a URL can be used as a base URL for more specific requests
- ◉ WebTarget allows configuring of matrix and query parameters
 - ◉ Usually parameters will be configured on the “final” URLs, rather than on base URLs from which others will be derived
- ◉ WebTarget then creates an Invocation.Builder

Invocation.Builder

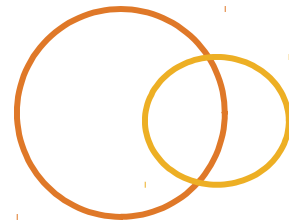
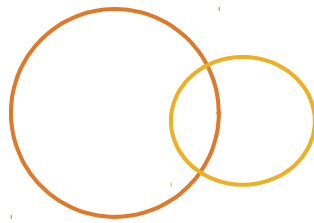


- ◉ Unsurprisingly, a builder for an Invocation
- ◉ The Builder may be used to manipulate headers that will be associated with the final invocation
 - ◉ Several methods are specific to particular, common, headers, such as accept
- ◉ Builder is then typically used to prepare an Invocation that's specific to a request type, e.g.:
 - ◉ `ib.buildPost(Entity e)`
 - ◉ `ib.buildGet()`

Invocation



- ◉ Invocation can be used to make the request immediately, resulting in a Response object, or allowing the response entity to be extracted directly
 - ◉ Extracting the entity directly prevents checking headers / status code
- ◉ Invocation can be used to launch the request asynchronously
 - ◉ Obtain a Future or a callback



- Entity is generic allowing it to represent structured data that will be converted to JSON or similar
- Entity has static factory methods allowing several entity variations
 - `form(MultivaluedMap<String,String> formData)`
 - `json(T entity)`
 - `text(T entity)`
 - `entity(T entity, MediaType mediaType)`

Example GET Receiving Text



```
Client c1 = ClientBuilder.newClient();  
WebTarget base =  
    c1.target("http://localhost:8080/"  
        + "jeecontext/v1/customers/");  
WebTarget oneCustomerName = base.path("/0/name");  
  
Invocation.Builder ib = oneCustomerName.request();  
ib.accept(MediaType.TEXT_PLAIN);  
  
Response resp = ib.get();  
String name = resp.readEntity(String.class);  
  
System.out.println("Response is " + name);
```

Example GET Receiving JSON



```
Client cl = ... [] as before
```

```
WebTarget base = ... [] as before
```

```
Invocation.Builder ib = ... [] as before
```

```
WebTarget oneWholeCustomer = base.path("1");
```

```
Invocation.Builder ib = oneWholeCustomer.request();  
ib.accept(MediaType.APPLICATION_JSON);
```

```
Response resp = ib.get();
```

```
System.out.println("As object response is "  
    + resp.readEntity(CustomerTO.class));
```

Example POST Sending JSON



Client c1 = ... [] as before

WebTarget base = ... [] as before

Invocation.Builder ib = ... [] as before

```
ib.accept(MediaType.APPLICATION_JSON);  
Response resp = ib.buildPost(Entity.json(  
    new CustomerTO("Tony", "Dunroamin")  
)).invoke();
```

CustomerTO returned =
`resp.readEntity(CustomerTO.class);`

Other Response Elements



- ◉ Response is the same class as used in the server
- ◉ Status and headers can be read:
 - ◉ `int status = resp.getStatus();`
 - ◉ `String aHeader = resp.getHeaderString("x-my-header");`
 - ◉ `MultivaluedMap<String, Object> hv = r.getHeaders();`
- ◉ Other response data such as length, date, cookies, media type, and allowed methods, can be read from this object too

Sending PATCH Requests



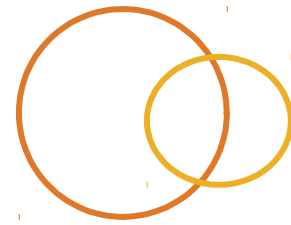
- ◉ In principle, a PATCH (or other non-standard request) may be sent like this:

```
...  
Invocation.Builder ib = ...  
  
Response r = ib.build("PATCH",  
    Entity.json(new CustomerTO("Phoenix", null))  
).invoke();
```

- ◉ In practice in Jersey this must be enabled first:

```
myClient.property(  
    HttpURLConnectionProvider.SET_METHOD_WORKAROUND,  
    true);
```

Lab Exercise



- Create a stand-alone Java program that connects to your TCRS service and invokes an operation on one of your existing service endpoints, so that it fetches a JSON structure over the wire, and JAX-RS client converts it into a Java object in memory