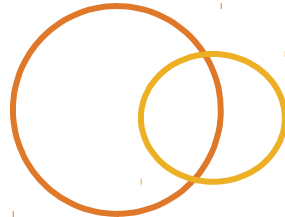
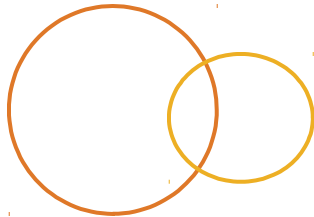
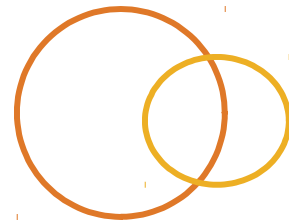
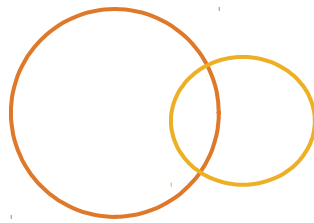


End to End Testing with RESTAssured

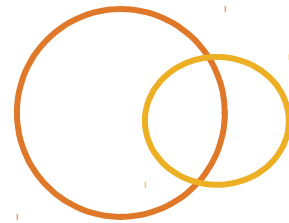
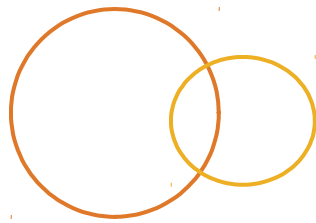


Objectives

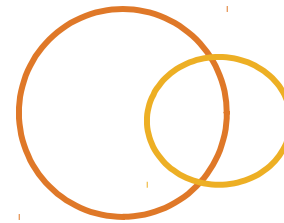


- ◉ Create a maven-based project for end-to-end tests using REST Assured and TestNG
- ◉ Locate REST Assured documentation
- ◉ Send GET, POST, PUT, DELETE, and PATCH requests on URLs and with entity bodies where appropriate
- ◉ Send requests with path parameters, query parameters, and headers
- ◉ Validate simple assertions about data in JSON entity responses
- ◉ Validate simple assertions about headers in responses
- ◉ Validate simple assertions about response status code

Objectives



- Give an overview of the key difference between REST Assured testing and options for “containerless” testing in the context of mocking and unit testing



- ◉ Project for testing / assertions on REST type services
- ◉ Works “over the wire”
 - ◉ Does not support “in-memory” testing, and therefore does not support mocking of service elements
 - ◉ Project hosted in Google code at:
- ◉ <https://code.google.com/p/rest-assured/>
- ◉ Getting started docs at:
- ◉ [.../rest-assured/wiki/GettingStarted](https://code.google.com/p/rest-assured/wiki/GettingStarted)
- ◉ Usage guide with lots of examples:
- ◉ [.../wiki/Usage](https://code.google.com/p/rest-assured/wiki/Usage)

Building Tests With REST Assured



- Build a maven project with dependency:

- groupId: com.jayway.restassured
- artifactId: rest-assured

- Make static imports

`com.jayway.restassured.RestAssured.*`

`com.jayway.restassured.matcher.RestAssuredMatchers.*`

`org.hamcrest.Matchers.*`

REST Assured Programming



- REST Assured emphasizes a fluent programming style and X-Path like field expressions for verification

```
get("http://localhost:8080/data/data/0")  
    .then().assertThat()  
    .statusCode(200)  
    .and()  
    .body("data0.value", equalTo("1"));
```

General Flow Of REST Assured



◉ REST Assured tests typically start with preparing the request:

◉ `given()`

- ◉ `formParam(key, value)`
- ◉ `queryParam(key, value)`
- ◉ `pathParam(key, value)` □ path params are named
- ◉ `cookie(key, value)`
- ◉ `header(key, value)`
- ◉ `request().body(entity)` or simply `body(entity)`
- ◉ `contentType(mimeType)`

General Flow Of REST Assured



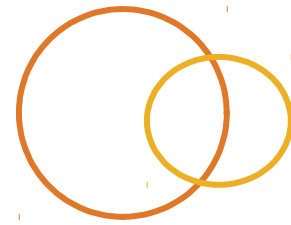
- ◉ After preparing the request, issue it:
- ◉ `given()...when()`
 - ◉ `post(url)`
 - ◉ `get(url)`
 - ◉ `put(url)`
 - ◉ `delete(url)`
- ◉ Use named path parameters already configured in the `given()` clause:
 - ◉ `get("http://blah.com/customers/{pathParam}")`

General Flow Of REST Assured



- ⦿ Then perform validations or extractions on the result
- ⦿ `given()...when()...then().assertThat()`
 - ⦿ `statusCode(value)`
 - ⦿ `header(key, value)`
 - ⦿ `body(equalTo(stringBody))`
 - ⦿ `body("path.spec.value", equalTo(5))`

Lab Exercise



- Create a Java test using that exercises one of the service endpoints in your existing code. Verify that the service returns the data you expect.