Feel at ease and

REST-assured

Johan Haleby

Agenda

- HTTP Basics
- REST Assured Basics
- Demo
- Beyond the Basics
- Demo
- Spring MVC Support

HTTP Request

- Request line
  - GET /images/logo.png HTTP/1.1
- Headers
  - HTTP 1.1 requests must include the Host: header.
  - E.g. Accept-Language: en
- Empty line
- Optional message body

HTTP Request Example

```
POST /enlighten/calais.asmx HTTP/1.1
Host: api.opencalais.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://clearforest.com/Enlighten"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
  <Enlighten xmlns="http://clearforest.com/">
  <licenseID>string </licenseID>
  <content>string</content>
  <paramsXML>string </paramsXML>
  </Enlighten>
  </soap:Body>
</soap:Envelope>
```

JAYWAY

HTTP Response

- Status Line
  - HTTP/1.1 200 OK
  - HTTP/1.1 404 Not Found
- Headers
  - 0 or more
  - E.g. Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
- Empty line
- Optional message body

HTTP Response Example

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Happy New Millennium!</h1>
(more file contents)
   .
   .
   .
</body>
</html>
```

Parameters

```
http://stackoverflow.com/questions/203495/testing-rest-webservices
http://www.google.se/search?sourceid=chrome&ie=UTF-8&q=google
POST /enlighten/calais.asmx/Enlighten HTTP/1.1
Host: api.opencalais.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length

licenseID=someId&content=someContent
```

Path parameter

Query parameter

Form parameter

REST Assured

- Built on-top of (a forked) HTTP Builder
- Focus on testing with BDD-like specification syntax
- Groovy implementation
- Java interface
- No Groovy config
- Standard dependency
- Apache2 license

REST Assured - Requests

```
get("/lotto");
post("/lotto");
delete("/lotto");
put("/lotto");
head("/lotto");
options("/lotto");
patch("/lotto");
```

Defaults to localhost:8080

JAYWAY

Parameters

```
// URL encoding happens automatically
given().param("parameter1", "value1").
        param("parameter2", "value2").
when().
        post("/something");
```

Path Parameters

```
given().
        pathParam("hotelId", "My Hotel").
        pathParam("roomNumber", 23).
when().
        post("/reserve/{hotelId}/{roomNumber}").
then().
        statusCode(200);
```

JAYWAY

Path Parameters

```
when().
   post("/reserve/{hotelId}/{roomNumber}", "My Hotel", 23).
then().
        statusCode(200);
```

JSON Parsing in Groovy

```
{"lotto":{
    "lottoId":5,
    "winning-numbers":[2,45,34,23,7,5,3],
    "winners":[{
      "winnerId":23,
      "numbers":[2,45,34,23,3,5]
    },{
      "winnerId":54,
      "numbers":[52,3,12,11,18,22]
    }]
  }
}
```

```
json.lotto.lottoId // 5
json.lotto.'winning-numbers' // [2, 45, 34, ..]
json.lotto.winners.size() // 2
json.lotto.winners[0] // "First winner"
json.lotto.winners.winnerId // [23, 54]
```

REST Assured - JSON

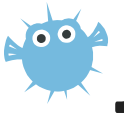
```
{"lotto":{
    "lottoId":5,
    "winning-numbers":[2,45,34,23,7,5,3],
    "winners":[{
      "winnerId":23,
      "numbers":[2,45,34,23,3,5]
    },{
      "winnerId":54,
      "numbers":[52,3,12,11,18,22]
    }]
  }
}
```

```
get("/lotto").then().body("lotto.lottoId", equalTo(5));
get("/lotto").then().body("lotto.winners.winnerId", hasItems(23,54));
get("/lotto").then().body("lotto.winners.size()", is(2));
get("/lotto").then().body("lotto.winners.winnerId[0]", is(23));
get("/lotto").then().body("lotto.winning-numbers", hasItems(2,45, ..));
```

JAYWAY

REST Assured - JSON

```
when().
        get("/lotto").
then().
        body("lotto.lottoId", equalTo(5)).
        body("lotto.winners.winnerId", hasItems(23,54)).
        body("lotto.winners.size()", is(2)).
        body("lotto.winners.winnerId[0]", is(23)).
        body("lotto.winners.winner-numbers", hasItems(2,45, ..));
```



JAYWAY

REST Assured - XML

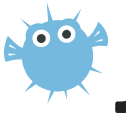
```
<records>
      <car name='HSV Maloo' make='Holden' year='2006'>
        <country>Australia</country>
        <record type='speed'>Pickup Truck with speed of 271kph</record>
      </car>
      <car name='P50' make='Peel' year='1962'>
        <country>Isle of Man</country>
        <record type='size'>Street-Legal Car at 99cm wide, 59kg</record>
      </car>
      <car name='Royale' make='Bugatti' year='1931'>
        <country>France</country>
        <record type='price'>Most Valuable Car at $15 million</record>
      </car>
</records>
```

```
when().
          get("/xml").
then().
          body("records.car[0].@name", equalTo("HSV Maloo")).
          body("records.car[-1].country", equalTo("France")).
          body("records.car[0..1].@year", hasItems("2006","1962")).
          body("records.car.country",
                              hasItems("Australia","Isle of Man", "France"));
```

JAYWAY

REST Assured – Simple Validation

DEMO!



JAYWAY

REST Assured – XML (X-Path)

```
<records>
      <car name='HSV Maloo' make='Holden' year='2006'>
        <country>Australia</country>
        <record type='speed'>Pickup Truck with speed of 271kph</record>
      </car>
      <car name='P50' make='Peel' year='1962'>
        <country>Isle of Man</country>
        <record type='size'>Street-Legal Car at 99cm wide, 59kg</record>
      </car>
      <car name='Royale' make='Bugatti' year='1931'>
        <country>France</country>
        <record type='price'>Most Valuable Car at $15 million</record>
      </car>
</records>
```

```
when().
          get("/xml").
then().
          body(hasXPath("/records/car[1]/@name"), equalTo("HSV Maloo")).
          body(hasXPath("/records/car[1]/country[text()='Australia'"));
```

JAYWAY

REST Assured – XML Schema

```
<records>
      <car name='HSV Maloo' make='Holden' year='2006'>
        <country>Australia</country>
        <record type='speed'>Pickup Truck with speed of 271kph</record>
      </car>
      <car name='P50' make='Peel' year='1962'>
        <country>Isle of Man</country>
        <record type='size'>Street-Legal Car at 99cm wide, 59kg</record>
      </car>
      <car name='Royale' make='Bugatti' year='1931'>
        <country>France</country>
        <record type='price'>Most Valuable Car at $15 million</record>
      </car>
</records>
```

```
get("/xml").then().body(matchesDtd(dtd));
get("/xml").then().body(matchesXsd(xsd));
```

REST Assured – JSON Schema

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "title": "Greeting",
  "description": "Will greet you",
  "id": "http://jsonschema.net",
  "required": true,
  "properties": {
    "greeting": {
      "type": "object",
      "id": "http://jsonschema.net/greeting",
      "required": true,
      "properties": {
        "firstName": {
          "type": "string",
          "id": "http://jsonschema.net/greeting/firstName",
          "required": true
        },
        "lastName": {
          "type": "string",
          "id": "http://jsonschema.net/greeting/lastName",
          "required": true
        }
      }
    }
  }
}
```

```
{
    "greeting":{
        "firstName":"John",
        "lastName":"Doe"
    }
}
```

JAYWAY

REST Assured – JSON Schema

```
given().
        param("firstName", "John").
        param("lastName", "Doe").
when().
        get("/greetJSON").
then().
        body(matchesJsonSchemaInClasspath(
                                    "greeting-schema.json"));
```

REST Assured - Request Body

```
// Binary serialization
given().body("{ "message" : "hello" }").when().post("/jsonBody");
given().body(message).when().post("/binaryBody");

// Object serialization
given().body(message, ObjectMapper.GSON).when().post("/beautiful-message");
given().
    param("something", "something").
    body(message, ObjectMapper.GSON).
when().
    post("/serializedJsonParameter")
```

REST Assured – Multi-Part

JAYWAY

REST Assured – Multi-Part

```
<html>
      <body>
            <div>Select file to upload:</div>
            <form id="feedUploadForm"
                  action="http://localhost:8080/multipart/file"
                  method="post" enctype="multipart/form-data">
               <input type="file" name="file" size="40">
               <input type=submit value="Upload!">
            </form>
      </body>
</html>
```

JAYWAY

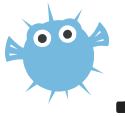
REST Assured – Multi-Part

```
given().
        multiPart(new File("/tmp/large_file.bin")).
when().
        post("/multipart/file").
then().
        statusCode(200);
```

JAYWAY

REST Assured – Response

```
Response response = get("/lotto");

// Get all headers
Headers allHeaders = response.getHeaders();

// Get a single header value:
String headerName = response.getHeader("headerName");

// Get all cookies
Map<String, String> allCookies = response.getCookies();

// Get a single cookie value:
String cookieValue = response.getCookie("cookieName");

// Get status line
String statusLine = response.getStatusLine();

// Get status code
int statusCode = response.getStatusCode();
```



JAYWAY

REST Assured – Response

```
Response response = get("/lotto");
String responseBody = response.getBody().asString();

String body = get("/lotto").asString();

byte[] byteArray = get("/lotto").asByteArray();

InputStream inputStream = get("/lotto").asInputStream();

Lotto lotto = get("/lotto").as(Lotto.class);

int lottoId  = get("/lotto").path("lotto.lottoId");
```

REST Assured – JSON Path

```
{"lotto":{
    "lottoId":5,
    "winning-numbers":[2,45,34,23,7,5,3],
    "winners":[{
      "winnerId":23,
      "numbers":[2,45,34,23,3,5]
    },{
      "winnerId":54,
      "numbers":[52,3,12,11,18,22]
    }]
  }
}
```

```
String json = get("/lotto").asString();

// Or a bit more efficiently
JsonPath jsonPath = new JsonPath(json).setRoot("lotto");
int lottoId = jsonPath.getInt("lottoId");
List<Integer> winnerIds = jsonPath.get("winners.winnerId");
```

JAYWAY

REST Assured – XML Path

```
<records>
      <car name='HSV Maloo' make='Holden' year='2006'>
        <country>Australia</country>
        <record type='speed'>Pickup Truck with speed of 271kph</record>
      </car>
      <car name='P50' make='Peel' year='1962'>
        <country>Isle of Man</country>
        <record type='size'>Street-Legal Car at 99cm wide, 59kg</record>
      </car>
</records>
```

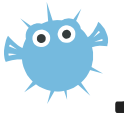
```
String xml = get("/xml").asString();

// Or a bit more efficiently
// Static import of "fromString(xml).setRoot("records");
XmlPath xmlPath = new XmlPath(xml);
String country = xmlPath.get("car[0].country");
List<String> carNames = xmlPath.get("car.@name");
```

JAYWAY

REST Assured - Authentication

- Basic
- Preemptive Basic
- Form
- Digest
- OAuth 1 and 2
- Certificate
- Custom

```
given().auth().basic("jetty","jetty").when().get("/secured/basic");
given().auth().certificate(certURL,"password”).when().get("/secured/cert");
```



REST Assured - Defaults

```
RestAssured.baseURI = "http://myhost.org";
RestAssured.port = 80;
RestAssured.basePath = "/resource";
RestAssured.authentication = basic("username", "password");
```

```
get("/xml")
```

http://myhost.org:80/resource/xml with basic auth

```
RestAssured.rootPath = "x.y.z";
RestAssured.filters(..);
RestAssured.requestContentType(..);
RestAssured.responseContentType(..);
RestAssured.requestSpecification = .. // Default request specification
RestAssured.responseSpecification = .. // Default response specification
RestAssured.urlEncodingEnabled = ..
```

```
RestAssured.reset();
```

JAYWAY

REST Assured – Specifications

```
given().
        queryParam("category", "books").
        cookie("user", "admin").
when().
        get("/book/22").
then().
        body(equalTo("Catch 22”));
```

REST Assured – Specifications

```
given().
        queryParam("category", "books").
        cookie("user", "admin").
when().
        get(“/book/42”).
then().
        body(equalTo("The Hitchhiker's Guide to the Galaxy”));
```

JAYWAY

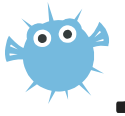
REST Assured – Specifications

Create a reusable specification ..

```
RequestSpecification books = new RequestSpecBuilder().
                addQueryParam("category", "books").
                addCookie("user", "admin").
                        build();
```

.. and use it

```
given().
        spec(books).
when().
        get("/book/22").
then().
        body(equalTo("Catch 22"));
```



Advanced validation

```
<records>
      <car name='HSV Maloo' make='Holden' year='2006'>
        <country>Australia</country>
        <record type='speed'>Pickup Truck with speed of 271kph</record>
      </car>
      <car name='P50' make='Peel' year='1962'>
        <country>Isle of Man</country>
        <record type='size'>Street-Legal Car at 99cm wide, 59kg</record>
      </car>
      <car name='Royale' make='Bugatti' year='1931'>
        <country>France</country>
        <record type='price'>Most Valuable Car at $15 million</record>
      </car>
</records>
```

Groovy Collections

- Has all Java methods
- And more:
  - collect
  - each
  - find
  - findAll
  - sum
  - ..

Groovy Find All

```groovy
def myList = ["Johan","Mattias","Per","Eric"]

// Find all names with length greater than or equal to 5
myList.findAll { it.length() >= 5 }
```

Result: ["Johan","Mattias"]

Groovy Find

```
def myList = ["Johan","Mattias","Per","Eric"]

// Find all names with length greater than or equal to 5
myList.find { it.length() >= 5 }
```

Result: "Johan"

Groovy Collect

```groovy
def myList = ["Johan","Mattias","Per","Eric"]

// Return a list with the length of each name
myList.collect { it.length()}

Result: [5, 7, 3, 4]
```

JAYWAY

Groovy Sum

```
def myList = [1,2,3]

myList.sum()

Result: 6
```

REST Assured – Advanced validation

DEMO!

Spring MVC Module

Spring MVC

```
@Controller
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    @RequestMapping(value = "/greeting", method = GET)
    public @ResponseBody Greeting greeting(
                @RequestParam(value="name", required=false,
                                    defaultValue="World") String name) {
        return new Greeting(counter.incrementAndGet(), format(template, name));
    }
}
```

Spring MVC

```
@Test public void
greeting_resource_returns_an_id_and_expected_greeting_in_body() {
    given().
            standaloneSetup(new GreetingController()).
            param("name", "Johan").
    when().
            get("/greeting").
    then().
            statusCode(200).
            body("id", equalTo(1)).
            body("content", equalTo("Hello, Johan!"));
}
```

Spring MVC

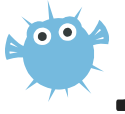
```
@Before public void
given_rest_assured_is_configured_with_greeting_controller() {
    RestAssuredMockMvc.standaloneSetup(new GreetingController());
}
```

Spring MVC

```
@Test public void
greeting_resource_returns_an_id_in_the_response_body() {
    given().
            param("name", "Johan").
    when().
            get("/greeting").
    then().
            body("id", equalTo(1)));
}

@Test public void
greeting_resource_returns_expected_greeting_in_body() {
    given().
            param("name", "Johan").
    when().
            get("/greeting").
    then().
            body("content", equalTo("Hello, Johan!"));
}
```

Spring MVC Module Benefits

- Simpler to start environment
- Simplifies mocking
- Tests are executed faster than having to go through a servlet container
- Test a controller in isolation
- Authentication support
- Don’t need to learn a new API
- REST Assured building blocks (including XmlPath)
- Easy to refactor if changing from Spring



JAYWAY

Spring MVC “Drawbacks”

- Doesn’t support all MockMvc features
- Larger dependency
- Possible feature lag
- Caution: Filters, Authentication etc may not applied!

Web Page

- Search “rest assured java” in Google
- http://code.google.com/p/rest-assured/
- http://blog.jayway.com/


JAYWAY