JavaOne™

JavaOne™

Developing JAX-RS Web Application Utilizing SSE and WebSocket

Martin Matula
Sr. Dev. Manager, Oracle

Program Agenda

- About This Lab
- Quick Intro to the Used Technologies
- Lab Exercises
- Getting Started
- Resources

About This Lab

- Follow the lab guide
- Exercises are self-paced
- Raise your hand if you get stuck – we are here to help
- To get most of the lab try to understand the code, don't just blindly copy-paste

Technologies Used in this Lab

Quick Intro

- Jersey/JAX-RS 2.0
  - Server-sent events
- Tyrus/Java API for WebSocket
- JSON Processing

JAX-RS 2.0/Jersey

Description

- Java API for RESTful Web Services
  - Annotation-based API for exposing RESTful web services
- New in JAX-RS 2.0
  - Client API
  - Filters/intereptors
  - Server-side content negotiation
  - Asynchronous processing

JAX-RS 2.0/Jersey

Client API

```
// Get instance of Client
Client client = ClientFactory.newClient();

// Get account balance
String bal = client.target("http://.../atm/{cardId}/balance")
    .pathParam("cardId", "111122223333")
    .queryParam("pin", "9876")
    .request("text/plain").get(String.class);
```

JAX-RS 2.0/Jersey

Where to get more info

- JavaOne sessions:
  - Pimp My RESTful Java Applications
    - Marek Potociar
    - Parc 55 - Cyril Magnin I - 10/3/12, 1:00pm - 2:00pm
- On the web:
  - Specification project: http://jax-rs-spec.java.net
  - Implementation project: http://jersey.java.net
  - Twitter: @gf_jersey

Java API for Web Socket

Description

- Annotation-based API for utilizing Web Socket protocol in Java web applications
  - Planned to be part of JavaEE 7
- Allows defining web socket endpoints
  - Handling onOpen, onClose, onError and onMessage events
  - Bi-directional communication between peers
- Support for encoders/decoders to map message content to/from Java objects

Java API for Web Socket

Example – Simple Endpoint

```
@WebSocketEndpoint("/echo")
public class EchoBean {
    @WebSocketMessage
    public String echo(String message) {
        System.out.println("Message received: " + message);
        return message + " (from your server)";
    }
}
```

Java API for Web Socket

Example – Decoder/Encoder

```
@WebSocketEndpoint(”/drawing/”,
    decoders = ShapeCoding.class, encoders = ShapeCoding.class,
)
public class DrawingWebSocket {
    @WebSocketMessage
    public void shapeCreated(Shape shape, Session session) { … }
}

public class ShapeCoding implements Decoder.Text<Shape>, Encoder.Text<Shape> {
    public Shape decode(String s) throws DecodeException { … }
    public boolean willDecode(String s) { … }
    public String encode(Shape object) throws EncodeException { … }
}
```

Java API for Web Socket

Where to get more information

- JavaOne Sessions
  - HTML5 WebSocket and Java, Danny Coward, CON7001
    - Oct 3, 4:30pm – 5:30pm, Parc 55, Cyril Magnin I
- On The Web
  - Specification Project: http://websocket-spec.java.net
  - Implementation: http://tyrus.java.net

Standard JSON API

Contents

- Parsing/Processing JSON
- Data binding : JSON text <-> Java Objects
- Two JSRs (similar to JAXP and JAXB)
  - Processing/Parsing (JSON-P) – Java EE 7
  - Binding (JSON-B) – Java EE 8

Java API for Processing JSON

JSR-353

- Streaming API to produce/consume JSON
  - Similar to StAX API in XML world
- Object model API to represent JSON
  - Similar to DOM API in XML world
- Aligns with Java EE 7 schedules
  - EDR ends soon
- EG (Oracle, RedHat, Twitter, 3 individual members)
  - Also, user community !

JSR-353: Java API for Processing JSON

JsonReader/JsonWriter

- JsonReader – reads JsonObject/JsonArray from i/o

```
try(JsonReader reader = new JsonReader(io)) {
    JsonObject jsonObj = reader.readObject();
}
```

- JsonWriter – writes JsonObject/JsonArray to i/o

```
try(JsonWriter writer = new JsonWriter(io)) {
    writer.writeObject(jsonObj);
}
```

Resources

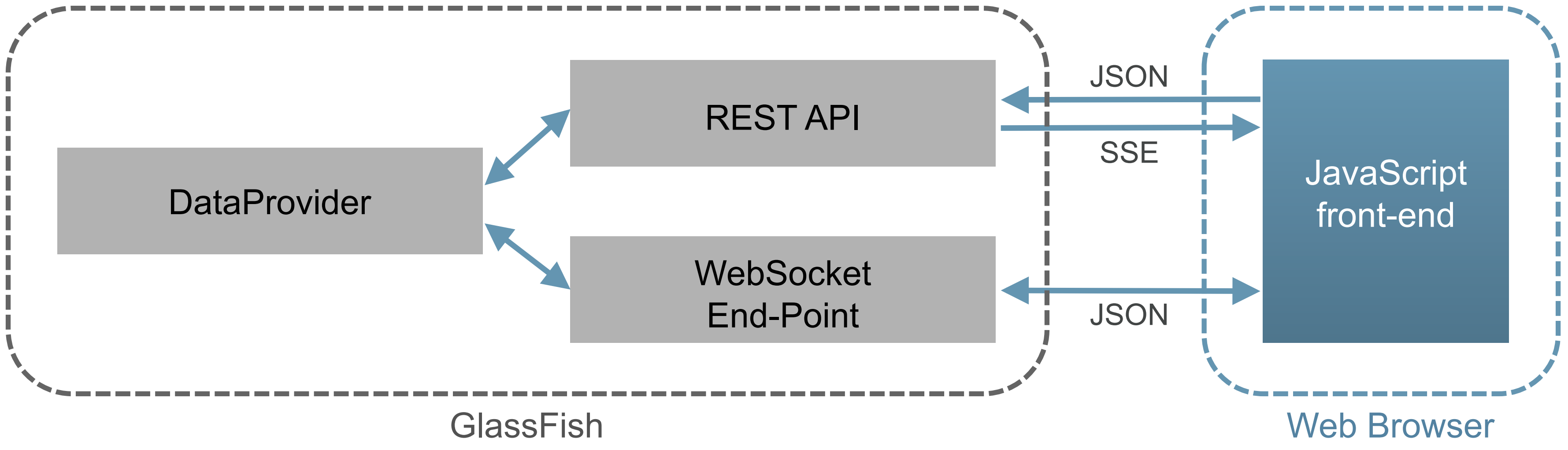
- JavaOne Session
  - CON3566 - JSR 353: Java API for JSON Processing - Jitendra Kotamraju
    - Wed, Oct 3rd, 10-11 am, Parc 55, Mission
- Projects
  - Specification Project - http://json-processing-spec.java.net
  - RI Project - http://jsonp.java.net
- Latest Javadoc
  - http://json-processing-spec.java.net/nonav/releases/1.0/edr/javadocs/index.html

Lab Exercises

- Drawing Board web application:
  - Exercise 1: Exposing RESTful API
  - Exercise 2: Adding Server-Sent Events
  - Exercise 3: Adding Web Sockets
- Simple Drawing Board client:
  - Exercise 4: Implementing a Simple Java Client

Drawing Board Application

High-Level Overview



Getting Started

- Launch HOL4461 virtual machine in VirtualBox (if not already started)
- Open lab-guide.pdf that’s on the desktop
- Follow the instructions
- Lab files installed under:
  - C:\Users\Lab\My Documents\hol

Additional Resources

- Follow **@gf_jersey** on Twitter (will post a link to the GitHub project with this lab there)
- Jersey – http://jersey.java.net
  - Mailing list: users@jersey.java.net
  - Fork Jersey on GitHub: http://github.com/jersey
- Tyrus – http://tyrus.java.net
- JSON Processing – http://jsonp.java.net

MAKE THE FUTURE JAVA

JavaOne™

ORACLE®

JavaOne™