JSR 311: JAX-RS: The Java™ API for RESTful Web Services

Marc Hadley, Paul Sandoz, Roderico Cruz

Sun Microsystems, Inc.
http://jsr311.dev.java.net/

TS-6411

Agenda

REST Primer

Why a Java™ Platform REST API

API Elements

Deployment Options

Issues

Demo

Agenda

REST Primer

Why a Java™ Platform REST API

API Elements

Deployment Options

Issues

Demo

REST

- **RE**presentational **S**tate **T**ransfer
- Name coined by Roy Fielding in his Ph.D thesis*
- Architectural Style of the Web

* http://ics.uci.edu/~fielding/pubs/dissertation/top.htm

REST Tenets

- Resources
  - Identified by a URI
  - Realized as representations
- Methods
  - Small, fixed set for all resources
  - Involve exchange of representations
- Representations
  - Embody state
  - Multiple alternate representations

REST vs. RPC

- RPC
  - Few endpoints, many custom methods
  - Few nouns, many verbs
  - `musicdb.getRecordings("magnum")`
- REST
  - Many resources, few fixed methods
  - Many nouns, few verbs
  - `GET /music/artists/magnum/recordings`

CRUD

The Atom Publishing Protocol

	Verb	Noun
Create	POST	Collection
Read	GET	Collection
Read	GET	Entry
Update	PUT	Entry
Delete	DELETE	Entry

HTTP

- HyperText Transfer Protocol
- The protocol of the WWW
- http://ietf.org/rfc/rfc2616.txt
- An application protocol
- HTTP and REST—Which came first: the chicken or the egg?

HTTP Example

Request

```
GET /music/artists/magnum/recordings HTTP/1.1
Host: media.example.com
Accept: application/xml
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 08 May 2007 16:41:58 GMT
Server: Apache/1.3.6
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0"?>
<recordings xmlns="…">
  <recording>…</recording>
  …
</recordings>
```

Agenda

REST Primer

Why a Java Platform REST API

API Elements

Deployment Options

Issues

Demo

REST APIs

- Lots of Web companies now offering REST APIs for their services
- Where both WS-* and REST API offered, REST API more widely used
  - REST APIs often easier to consume with scripting languages
  - Browser-based experimentation also easy
- Current platform APIs for building REST WS are rather low level
  - Many opportunities for simplifying development

Example

- Music Collection
  - `/music/artists`
  - `/music/artists/{id}`
  - `/music/recordings`
  - `/music/recordings/{id}`
  - `/music/artists/{id}/recordings`
  - `/music/genre/{id}`
  - `/music/format/{id}`
- XML and JSON support

Artist Resource Using Servlet API

```
public class Artist extends HttpServlet {

    public enum SupportedOutputFormat {XML, JSON};

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String accept = request.getHeader("accept").toLowerCase();
        String acceptableTypes[] = accept.split(",");
        SupportedOutputFormat outputType = null;
        for (String acceptableType: acceptableTypes) {
            if (acceptableType.contains("*/*") || acceptableType.contains("application/*") ||
                acceptableType.contains("application/xml")) {
                outputType=SupportedOutputFormat.XML;
                break;
            } else if (acceptableType.contains("application/json")) {
                outputType=SupportedOutputFormat.JSON;
                break;
            }
        }
        if (outputType==null)
            response.sendError(415);
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        if (pathSegments.length < 2 && pathSegments.length > 3)
            response.sendError(404);
        else if (pathSegments.length == 3 && pathSegments[2].equals("recordings")) {
            if (outputType == SupportedOutputFormat.XML)
                writeRecordingsForArtistAsXml(response, artist);
            else
                writeRecordingsForArtistAsJson(response, artist);
        } else {
            if (outputType == SupportedOutputFormat.XML)
                writeArtistAsXml(response, artist);
            else
                writeArtistAsJson(response, artist);
        }
    }
    private void writeRecordingsForArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeRecordingsForArtistAsJson(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsJson(HttpServletResponse response, String artist) { ... }
}
```

There Must Be a Better Way

Server-side API Wish List

- High level
- Declarative
- Clear mapping to REST concepts
- Takes care of the boilerplate code
- Make it easy to do the right thing
- Graceful fallback to low-level APIs when required

- **Disclaimer**: Early in the Java Specification Request (JSR) process, everything from here on in **liable to change**!

Agenda

REST Primer

Why a Java Platform REST API

API Elements

Deployment Options

Issues

Demo

Artist Resource Using Servlet API

```
public class Artist extends HttpServlet {

    public enum SupportedOutputFormat {XML, JSON};

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String accept = request.getHeader("accept").toLowerCase();
        String acceptableTypes[] = accept.split(",");
        SupportedOutputFormat outputType = null;
        for (String acceptableType: acceptableTypes) {
            if (acceptableType.contains("*/*") || acceptableType.contains("application/*") ||
                acceptableType.contains("application/xml")) {
                outputType=SupportedOutputFormat.XML;
                break;
            } else if (acceptableType.contains("application/json")) {
                outputType=SupportedOutputFormat.JSON;
                break;
            }
        }
        if (outputType==null)
            response.sendError(415);
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        if (pathSegments.length < 2 && pathSegments.length > 3)
            response.sendError(404);
        else if (pathSegments.length == 3 && pathSegments[2].equals("recordings")) {
            if (outputType == SupportedOutputFormat.XML)
                writeRecordingsForArtistAsXml(response, artist);
            else
                writeRecordingsForArtistAsJson(response, artist);
        } else {
            if (outputType == SupportedOutputFormat.XML)
                writeArtistAsXml(response, artist);
            else
                writeArtistAsJson(response, artist);
        }
    }
    private void writeRecordingsForArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeRecordingsForArtistAsJson(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsJson(HttpServletResponse response, String artist) { ... }
}
```

Automated Content Negotiation

```
public class Artist extends HttpServlet {

    @ProduceMime("application/xml")
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        if (pathSegments.length < 2 || pathSegments.length > 3)
            response.sendError(404);
        else if (pathSegments.length == 3 &&
                pathSegments[2].equals("recordings"))
            writeRecordingsForArtistAsXml(response, artist);
        else
            writeArtistAsXml(response, artist);
        }
    }

    ...
}
```

URI Templates

```
@UriTemplate("/artists/{id}")
public class Artist extends HttpServlet {

    @ProduceMime("application/xml")
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
        throws ServletException, IOException {
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        ...
    }

    ...
}
```

POJO

```
@UriTemplate("/artists/{id}")
public class Artist {

    @ProduceMime("application/xml")
    @HttpMethod
    InputStream getXml(
      @UriParam("id") String artist) {
        ...
    }

    ...
}
```

RESTful Web Services (JAX-RS) POJO

```
@UriTemplate("/artists/{id}")
@ProduceMime("application/xml")
public class Artist {

    @HttpMethod
    InputStream getXml(@UriParam("id") String artist) { ... }

    @HttpMethod
    @ProduceMime("application/json")
    InputStream getJson(@UriParam("id") String artist) { ... }

    @HttpMethod
    @UriTemplate("recordings")
    InputStream getRecordingsXml(@UriParam("id") String artist) { ... }

    @HttpMethod
    @ProduceMime("application/json")
    @UriTemplate("recordings")
    InputStream getRecordingsJson(@UriParam("id") String artist) { ... }
}
```

Artist Resource Using Servlet API

```
public class Artist extends HttpServlet {

    public enum SupportedOutputFormat {XML, JSON};

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String accept = request.getHeader("accept").toLowerCase();
        String acceptableTypes[] = accept.split(",");
        SupportedOutputFormat outputType = null;
        for (String acceptableType: acceptableTypes) {
            if (acceptableType.contains("*/*") || acceptableType.contains("application/*") ||
                acceptableType.contains("application/xml")) {
                outputType=SupportedOutputFormat.XML;
                break;
            } else if (acceptableType.contains("application/json")) {
                outputType=SupportedOutputFormat.JSON;
                break;
            }
        }
        if (outputType==null)
            response.sendError(415);
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        if (pathSegments.length < 2 && pathSegments.length > 3)
            response.sendError(404);
        else if (pathSegments.length == 3 && pathSegments[2].equals("recordings")) {
            if (outputType == SupportedOutputFormat.XML)
                writeRecordingsForArtistAsXml(response, artist);
            else
                writeRecordingsForArtistAsJson(response, artist);
        } else {
            if (outputType == SupportedOutputFormat.XML)
                writeArtistAsXml(response, artist);
            else
                writeArtistAsJson(response, artist);
        }
    }
    private void writeRecordingsForArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeRecordingsForArtistAsJson(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsJson(HttpServletResponse response, String artist) { ... }
}
```

Declaring Resource Class Methods

- Annotate with **`@HttpMethod`**
  - Java method name can be anything
  - Default HTTP method taken from Java method name

```
@HttpMethod
public XXX getXXX()

@HttpMethod("GET")
public XXX find()
```

Method Return Types

- **`HttpResponse`**—specialized subclasses for common types of response: redirection, resource creation, etc.
- **`Representation<T>`**—control over entity related headers: media type, language, etc.
- **`T`**—when a default media type can be determined
  - `T` extensible via SPI

```
@HttpMethod
public Recording getRecording()
```

Method Parameters

- Multiple annotated with **`@UriParam`**, **`@QueryParam`**, **`@MatrixParam`**, **`@HeaderParam`**
  - Flexible typing
- Zero or one of type **`Entity<T>` or `T`**
  - **`T`** extensible via SPI

```
@HttpMethod
public Recording updateRecording(
    @UriParam("id") String id,
    Recording updatedRecording)
```

Agenda

REST Primer

Why a Java Platform REST API

API Elements

Deployment Options

Issues

Demo

Deployment Options

- Servlet and Java APIs for XML Web Services (JAX-WS) Provider required by JSR
- Sun RI adds support for Grizzly and Java Platform, Standard Edition 6 (Java SE 6 platform) Lightweight HTTP Server
- Restlet support expected
- SPI for supporting other containers

Agenda

REST Primer

Why a Java Platform REST API

API Elements

Deployment Options

Issues

Demo

Open Issues

- Annotations vs. Classes/Interfaces
- Resource lifecycle
  - Per request, per session, singleton
- Container vs. API support
  - Security
  - Filters/Guards
- Other language support
  - Annotation use difficult in JavaScript™ programming language, JRuby, etc.
  - Lower level APIs start to lose the benefits

Agenda

REST Primer

Why a Java Platform REST API

API Elements

Deployment Options

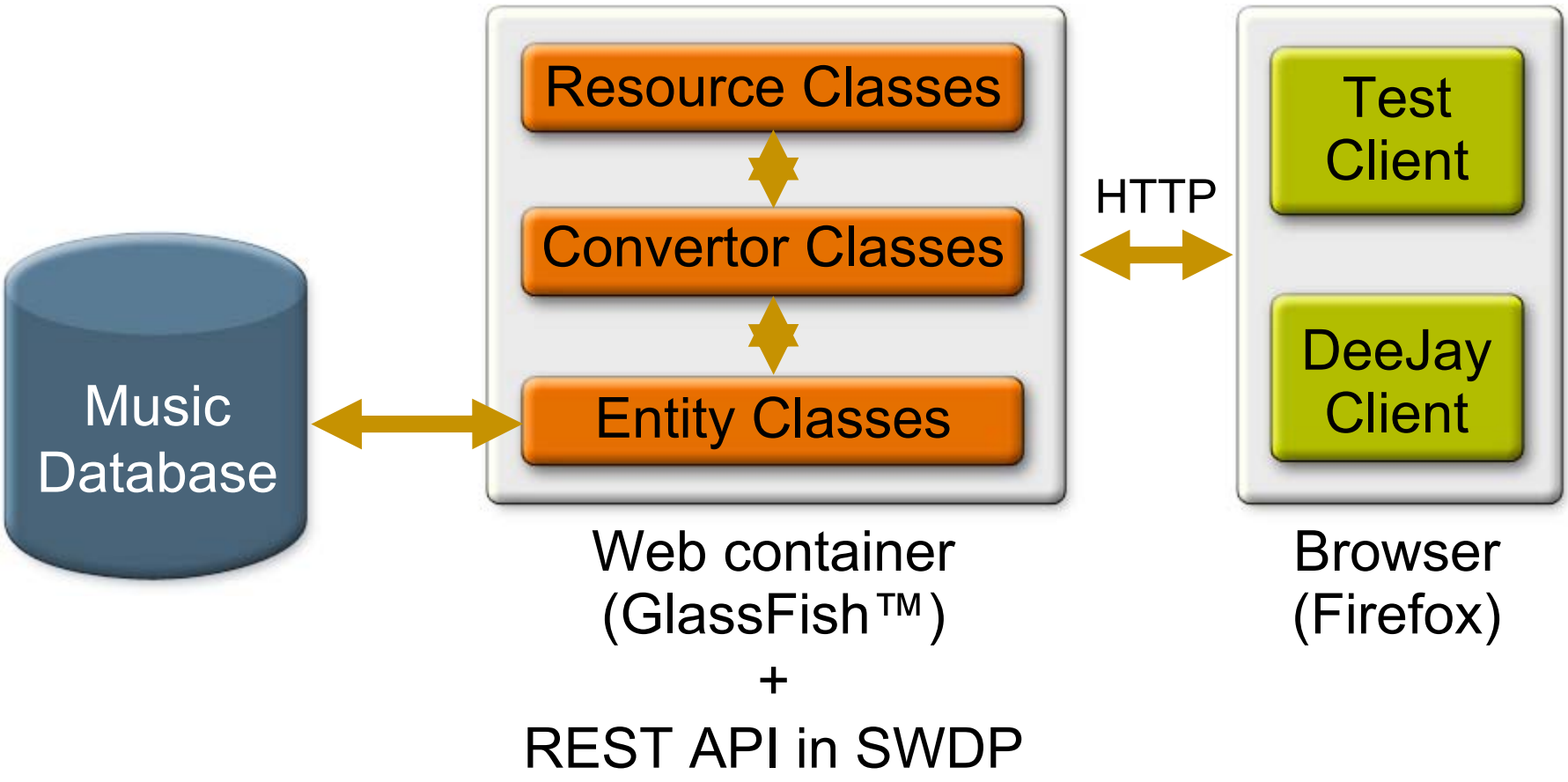
Issues

Demo

RESTful Tooling

- Rapid RESTful application development
  - Generate, deploy, test, edit, deploy, test, edit, ...
- Generate Web service from database
  - Expose existing data onto the Web in a few clicks
  - Container/item resource pattern maps to DB tables
- Generate Client artifacts from Web service
  - For client applications or for testing

Music Service Overview



DEMO

API and Tools

Summary

- High-level declarative programming model
- REST concepts reflected in the API
- Flexible typing, runtime takes care of common conversions
- SPIs to extend capabilities
  - Pluggable support for types, containers, and resolvers
- Flexible deployment options

For More Information

- TS-6029: “Beyond Blogging: Feeds in Action”
- BOF-6412: “Describing RESTful Applications: WADLing with Java”
- **http://jcp.org/en/jsr/detail?id=311**
- **http://jsr311.dev.java.net/**
- **http://developers.sun.com/web/swdp**
- Atom Publishing Protocol

Q&A

JSR 311: JAX-RS: The Java API for RESTful Web Services

Marc Hadley, Paul Sandoz, Roderico Cruz

Sun Microsystems, Inc.
http://jsr311.dev.java.net/

TS-6411