JAX-RS 2.1 Reloaded

What you see is what you get (hopefully)!

Markus KARG (Head Crashing Informatics, JSR 339, JSR 370)

Java Forum Stuttgart, 2017-07-06
Legal Disclaimer

This presentation expresses solely my personal opinion and is not necessarily aligned with the official statement of any of my customers or employers, the JSR 339 and 370 Expert Group, Oracle Corp., or any other named company.

All trademarks belong to their particular owners, even if not declared explicitly.

The Cheeky™ comic character is used by courtesy of inviticon℠.
How To Become An EG Member

- Born 1973
- ZX Spectrum (~1985)
- State-Qualified Information Scientist (1997)
- Java Addict (1997)
- Jersey Contributor (Jersey 0.8)
- JAX-RS EG Member (JSR 339, 370)
- Real-World Head of Development (ISV)

https://headcrashing.wordpress.com
markus@headcrashing.eu
Today's Agenda

- JAX-RS – The key to Java Micro Services
- Source code demonstrating selected features
- Q & A
„JAX-RS is one of Java EE's most important APIs.“

(Ed. Burns, Oracle, paraphrased)
Java 4 Cloud

Mobile Things

- JSON
- Dep. Injection
- NoSQL
- REST
- Runtime
- Container

JavaFX

JSON-B

- CDI
- JPA++
- JAX-RS
- JRE
- Docker
WARNING

JAX-RS 2.1 API is not final yet!
https://github.com/jax-rs/jax-rs.github.io/tree/master/apidocs/2.1

The content of the image is a screenshot of a GitHub repository named `jax-rs/jax-rs.github.io`. The repository contains documentation for JAX-RS API, specifically version fixes for the 2.1 javadoc. The screenshot shows the structure of the repository's files, including directories for `META-INF`, `doc-files`, `javax/ws/rs`, `allclasses-frame.html`, `allclasses-noframe.html`, `constant-values.html`, and `deprecated-list.html`. Each file has a timestamp indicating the last commit, which is 9-10 days ago.
Changes JAX-RS 2.0 → 2.1

✔ Java 8: Lambdas, Streams & CompletableFuture
✔ SSE: Pushing Events To The Client
✘ Improved CDI Integration
✘ NIO in Providers / Filters / Interceptors
✘ Declarative Security
✘ WARNING: JAXB becomes conditional
✔ JSON-B becomes mandatory
✘ Improved HATEOAS
✔ Reactive Client API: Simplifying asynchronous chains
✔ Support for MVC (JSR 370)
Jumpstart
Chapter 1. Getting Started

This chapter provides a quick introduction on how to get started building RESTful services using Jersey. The example described here uses the lightweight Grizzly HTTP server. At the end of this chapter you will see how to implement equivalent functionality as a JavaEE web application you can deploy on any servlet container supporting Servlet 2.5 and higher.

1.1. Creating a New Project from Maven Archetype

Jersey project is built using Apache Maven software project build and management tool. All modules produced as part of jersey project build are pushed to the Central Maven Repository. Therefore it is very convenient to work with Jersey for any Maven-based project as all the released (non-SNAPSHOT) Jersey dependencies are readily available without a need to configure a special maven repository to consume the Jersey modules.

Note

In case you want to depend on the latest SNAPSHOT versions of Jersey modules, the following repository configuration needs to be added to your Maven project pom:

```xml
<repository>
  <id>snapshots-repository.java.net</id>
  <name>Java.net Snapshot Repository for Maven</name>
  <url>https://maven.java.net/content/repositories/snapshots</url>
  <layout>default</layout>
</repository>
```

Since starting from a Maven project is the most convenient way for working with Jersey, let's now have a look at this approach. We will now create a new Jersey project that runs on top of a Grizzly consumer. We will use a Jersey-provided maven archetype. To create the project, execute the following Maven command in the directory where the new project should reside:

```
mvn archetype:generate -DarchetypeArtifactId=jersey-quickstart-grizzly2
    -DarchetypeGroupId=org.glassfish.jersey.archetypes -DinteractiveMode=false
    -DgroupId=com.example -DartifactId=simple-service -Dpackaging=war
    -DarchetypeVersion=2.26-b87
```

Feel free to adjust the group, package and/or artifactId of your new project. Alternatively, you can change it by updating the new project pom.xml once it gets generated.

1.2. Exploring the Newly Created Project

Once the project generation from a Jersey Maven archetype is successfully finished, you should see the new simple-service project directory created in your current location. The directory contains a standard Maven project structure:

Project build and management configuration is described in the pom.xml located in the project root directory.

Project sources are located under src/main/java.
Live Demo
<table>
<thead>
<tr>
<th>Branch</th>
<th>Protected</th>
<th>Project</th>
<th>Description</th>
<th>Days Ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>java8</td>
<td>protected</td>
<td>project</td>
<td>Java 8</td>
<td>6</td>
</tr>
<tr>
<td>jaxrs20</td>
<td>protected</td>
<td>project</td>
<td>Optional/Instant</td>
<td>4</td>
</tr>
<tr>
<td>microservice</td>
<td>protected</td>
<td>project</td>
<td>microservice</td>
<td>6</td>
</tr>
<tr>
<td>rx</td>
<td>protected</td>
<td>project</td>
<td>rx</td>
<td>6</td>
</tr>
<tr>
<td>slides</td>
<td>default</td>
<td>protected</td>
<td>Slides</td>
<td>6</td>
</tr>
<tr>
<td>sse</td>
<td>protected</td>
<td>project</td>
<td>SSE</td>
<td>6</td>
</tr>
</tbody>
</table>
Java 8
$ mvn clean package exec:java

INFO Scanning for projects...

INFO BUILDING DEMO 1.0-SNAPSHOT

INFO Deleting C:\Users\Markus\git\jaxrs-2.1-reloaded\target

INFO -- maven-clean-plugin:2.3.2:clean (default-clean) @ demo ---
INFO Deleting C:\Users\Markus\git\jaxrs-2.1-reloaded\target

INFO -- maven-resources-plugin:2.6:resources (default-resources) @ demo ---
INFO Using UTF-8 encoding to copy filtered resources.
INFO skip non existing resourceDirectory C:\Users\Markus\git\jaxrs-2.1-reloaded\src\main\resources

INFO -- maven-compiler-plugin:3.1:compile (default-compile) @ demo ---
INFO Compiling 7 source files to C:\Users\Markus\git\jaxrs-2.1-reloaded\target\classes

INFO -- maven-resources-plugin:2.6:testResources (default-testResources) @ demo ---
INFO Using UTF-8 encoding to copy filtered resources.
INFO skip non existing resourceDirectory C:\Users\Markus\git\jaxrs-2.1-reloaded\src\test\resources

INFO -- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ demo ---
INFO No sources to compile.

INFO -- maven-surefire-plugin:2.12.4:test (default-test) @ demo ---
INFO No tests to run.

INFO -- maven-jar-plugin:2.4:jar (default-jar) @ demo ---
INFO Building jar: C:\Users\Markus\git\jaxrs-2.1-reloaded\target\demo-1.0-SNAPSHOT.jar

INFO -- exec-maven-plugin:1.2.1:java (default-cli) @ validate @ demo ---
INFO exec-maven-plugin:1.2.1:java (default-cli) @ validate @ demo <<<

INFO -- exec-maven-plugin:1.2.1:java (default-cli) @ demo ---
INFO Started listener bound to [localhost:8080]
INFO [HttpServer] Started.

INFO Jersey app started with WADL available at http://localhost:8080/myapp/application.wadl
INFO Hit enter to stop it...
Java 8: Lambdas, Streams & CompletableFuture

- Code runs faster and is more concise
  - Concise code with Lambda Expressions
  - Map-reduce solutions with Stream
    - Example: MessageBodyReader could parse entity with parallel threads
  - Reactive programming with CompletableFuture
  - JAX-RS API simplified
    - CompletableFutureStage is a valid JAX-RS 2.1 return type
Live Demo
SSE
Client \[\xrightarrow{\text{Request}}\] Server

\[\xleftarrow{\text{Response}}\]

JAX-RS 2.0
Client -> Request -> Server

Request <-> Response

JAX <-> RS 2.1
SSE (Server Sent Events)

- SSE here is **literally** that particular technology, but *not* a paradigm!
  - WebSockets are *not* supported!
- RESTful SSE is REpresentational State-Changes Transfer
- JAX-RS originally was about REST
- REST *typically* is interpreted as Request-Response
- REST *does not mandate* Request-Response

Imagine the combination of SSE with an reactive API!

„Whenever event of type X is received, process it just like a request or response.“

(See Wikipedia on SSE)
Live Demo
@Deprecated\(\text{since}="9", \text{forRemoval}=true\)

Module java.xml.bind

Deprecated, for removal: This API element is subject to removal in a future version.

Defines the Java Architecture for XML Binding (JAXB) API.

Since: 9
jre9/bin/java --add-modules java.xml.bind -jar MyRestApplication.jar

Only while stocks last!
Browse Central For  org.glassfish.jaxb : jxb-bom : 2.3.0-b170127.1453

Click on a link above to browse the repository.

Project Information

GroupId:  org.glassfish.jaxb
ArtifactId:  jxb-bom
Version:  2.3.0-b170127.1453

Project Object Model (POM)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--

DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.

Copyright (c) 2013-2017 Oracle and/or its affiliates. All rights reserved.

The contents of this file are subject to the terms of either the GNU General Public License Version 2 only ("GPL") or the Common Development and Distribution License("CDDL") (collectively, the "License"). You may not use this file except in compliance with the License. You can obtain a copy of the License at  
http://glassfish.java.net/public/CDDL+GPL_1_1.html or packager/legal/LICENSE.txt. See the License for the specific language governing permissions and limitations under the License.

When distributing the software, include this License Header Notice in each file and include the License file at packager/legal/LICENSE.txt.

GPL Classpath Exception:
Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the GPL Version 2 section of the License file that accompanied this code.

-->
```
JSON-B
Java API for JSON Binding (JSON-B)

- JSON is the de-facto standard for RESTful web services.
- JSON-B API is to JSON what JAXB is to XML.
- It’s simply straightforward to declare JSON-B support as MANDATORY.

Best Practice

Implement a Gateway Service providing JSON and XML using two Entity Providers.

- Never use @Produces at methods but only at Message Body Writers.
- JAX-RS will select the right Entity Provider with respect to Accept: header.
  - All kinds of clients will work, it is extensible, and provides good SoC.
Live Demo
Reactive Programming
Reactive Programming

- Java 8 provides core technology: CompletableFuture

- No standard for reactive Java so far
  - Java 9 will bring Flow API, JAX-RS 2.1 already supports that
  - Several third party frameworks already supported
  - CompletionStage is a valid return type in JAX-RS 2.1
Live Demo
Ivar Grimstad
ivargrimstad
Java Champion, JUG Leader, JCP EC and EG Member @mvc-spec, @javaee-security-spec, Speaker, Java EE Advocate, , NetBeans Dream Team

Christian Kaltepoth
chkal
Senior Developer @ ingenit in Germany, open source guy, speaker, JSR 371 expert group, ...
https://github.com/mvc-spec

mvc-spec
The public repository for JSR 371.
https://java.net/projects/mvc-spec
How The Expert Group Works

- „Benevolent Dictatorship”
- Oracle develops Jersey, i.e. JAX-RS RI.
- Pavel changed modus operandi in a good way: Open discussion!
- Oracle *still* decides, we have to live with the result.
- No democratic vote.