Seam and RESTEasy:

*You haven't seen **REST** yet*

Dan Allen
Author of Seam in Action
Senior Software Engineer
JBoss, by Red Hat

Lincoln Baxter, III
Creator of PrettyFaces
Co-founder of OcpSoft

jboss.org

Agenda

- REST principles
- JAX-RS and RESTEasy
- Seam RESTEasy integration
- Demo

jboss.org

Abusing HTTP

jboss.org

REST to the rescue

Web Service

jboss.org

REST, spelled out

REpresentational State Transfer

jboss.org

Staying grounded with REST

- Simple
- Lightweight
- High performance

jboss.org

Core ingredients of the Web

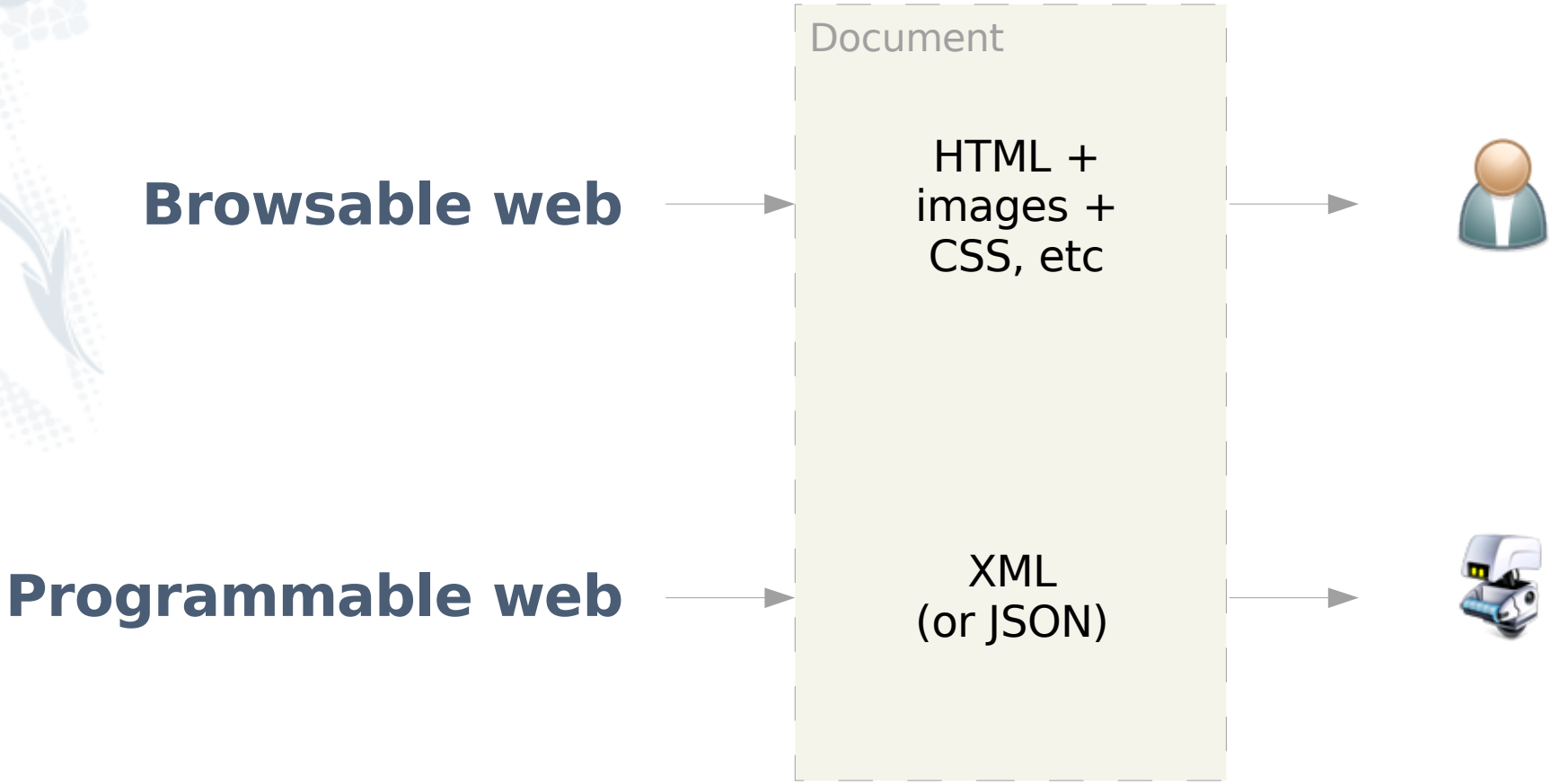
- **HTTP application protocol**
- **URI naming standard**
- **XML markup language**
  (and alternatives)

jboss.org

Every web site is a service

jboss.org

A tale of two webs



jboss.org

Enabling automation

RESTful architectural principles

- **Addressable resources**
- **Uniformed, constrained interface**
  - GET, PUT, DELETE, POST
- **Representation-oriented**
  - multiple formats
- **Stateless communication**

Addressable resources

- **URI for every resource in system**
- **Resource reachable by unique ID**

http://socialize.com/services/mojavelinux/status/311

- **Provides scoping information**
  - Query string used to narrow result set
- **Stepping stones**
  - Makes it possible to link (linkability)
  - Allows disparate applications to interact

Uniformed, constrained interface

- **Protocol method == operation**
  - 4 HTTP methods: *GET, PUT, DELETE, POST*
- **An architecture based on 4 methods?**
  - SQL (SELECT, INSERT, UPDATE, DELETE)
  - JMS (send, receive)

HTTP methods

- **GET - read only, idempotent and safe**
- **PUT - insert or update, idempotent**
- **DELETE - remove services, idempotent**
- **POST - *NOT* idempotent *NOR* unsafe**
  - constraints are relaxed for flexibility

Representation-oriented

- **Data has representation**
  - Negotiated between client and server
- **HTTP was designed for this purpose**
  - Content-Type header (MIME type)
  - Accept* headers

Stateless communication

- **More scalable**
  - GET lends itself well to caching
- **Client maintains state**
- **Takes burden off server**

Respect the medium

- **Request**
  - HTTP method
  - URI
  - Request headers
  - Entity body
- **Response**
  - HTTP response code
  - Response headers
  - Entity body

What do you need to REST?

- **HTTP client** (browser, bot, smart phone)
- **HTTP server that speaks REST**

JAX-RS

jboss.org

JSR-311: JAX-RS

- **Java APIs for developing Web Services following the REST architectural style**
- **Goals:**
  - POJO-based (annotations)
  - HTTP-centric
  - Format independent (MIME type)
  - Container independent
  - Inclusion in Java EE 5+

Our first REST resource

http://socialize.com/services/timeline

Our first JAX-RS resource

```
@Path("/timeline")
public class TimelineService {

  @GET
  public String getUpdates() {
    return "<updates><update>...</update></updates>";
  }

}
```

jboss.org

How it works

- REST servlet handles GET request
- Instance of TimelineService is created
- @GET method called
- Return value sent as response
- Resource instance thrown away

JAX-RS component model is intentionally simple!

Throttling the response

http://socialize.com/services/timeline?count=25

Accepting a query parameter

```
@Path("/timeline")
public class TimelineService {

  @GET
  public String getUpdates(@QueryParam("count")
      @DefaultValue("50") int count) {
    ...
  }

}
```

http://socialize.com/services/timeline ↴

http://socialize.com/services/timeline?count=50

jboss.org

Parameter types

- @QueryParam – Query string
- @HeaderParam – HTTP header
- @CookieParam – HTTP cookie
- @FormParam – Form input
- @PathParam – URI path

Stepping into a sub-resource

http://socialize.com/services/timeline/mojavelinux

Mapping a path parameter

```
@Path("/timeline")
public class TimelineService {

  @GET
  @Path("/{user}")
  public String getUpdates(@PathParam("user") String u) {
    ...
  }

}
```

Name defined in path expression; segment injected into method

Negotiating a response

- **So what's in the response?**
  - Plain text?
  - HTML?
  - XML?
  - JSON?
- **The client needs to tell us**
  - Lists formats, weighted by preference
- **We have to decide what we support**
  - Respond with best match

Producing explicitly

```
@Path("/timeline")
public class TimelineService {

  @GET
  @Path("/{user}")
  @Produces("application/xml")
  public String getUpdatesXml(@PathParam("user") String u) {
    ...
  }

}
```

Specify which formats are supported using @Produces

jboss.org

Producing explicitly

```
@Path("/timeline")
public class TimelineService {

  @GET
  @Path("/{user}")
  @Produces("application/json")
  public String getUpdatesJson(@PathParam("user") String u) {
    ...
  }

}
```

Specify which formats are supported using @Produces

Simplifying response production

- **Creating XML and JSON is laborious :(**
- **JAX-RS supports converters**
  - HTTP entity body readers/writers
- **Better yet, built-in JAXB provider!**
  - Object ⇔ XML
- **RESTEasy provides more built-ins**

A model with XML marshaling hints

```
@XmlRootElement(name = "updates")
public class Timeline implements Serializable {

   private List<Update> updates = new ArrayList<Update>();

   @XmlElement(name = "update")
   public List<Update> getUpdates() {
      return updates;
   }

   public void setUpdates(List<Update> updates) {
      this.updates = updates;
   }

}
```

jboss.org

A model with XML marshaling hints

```
@XmlRootElement(name = "update")
public class Update implements Serializable {

   private Long id;
   private User user;
   private Date created;
   private String text;

   // getters and setters

}
```

jboss.org

Turning production over to JAXB

```
@Path("/timeline")
public class TimelineService {

  @GET
  @Path("/{user}")
  @Produces("application/xml")
  public Timeline getUpdates(@PathParam("user") String u) {
    ...
  }

}
```

jboss.org

RESTEasy

JBoss Community

- **Fully certified JAX-RS implementation**
- **Portable to any container**
- **Embedded server for testing**
- **Client-side framework for JAX-RS**
- **Response caching and compression**
- **Rich set of providers**
  - XML, JSON, Atom, YAML, etc.
- **Asynchronous support**

jboss.org

What does Seam provide?

- RESTEasy bootstrap and configuration
- Automatic resource/provider discovery
- Resource/provider as Seam component
- Seam security (role, rule and ACLs)
- HTTP authentication
- Exception to HTTP response mapping
- REST CRUD framework
- HTTP session control
- Facelets templates

jboss.org

Typical JAX-RS setup

```
public abstract class Application {
  public abstract Class<?>> getClasses
  public abstract Object>getSingletons()
}

public SocializeConfig extends Application {
  ...
}

<context-param>
  <param-name>javax.ws.rs.core.Application</param-name>
  <param-value>com.socialize.SocializeConfig</param-value>
</context-param>
```

jboss.org

REST as a Seam resource

http://socialize.com/seam/resources/rest/timeline/mojavelinux

Seam-infused REST

```
@Name("timelineService")
@Path("/timeline")
public class TimelineService {
   @In TimelineDao timelineDao;

   @GET
   @Path("/{user}")
   @Produces("application/xml")
   public Timeline getUpdates(@PathParam("user") String user) {
      return timelineDao.fetchAll(user);
   }

}
```

jboss.org

Trapping exceptions

```
<exception class="com.socialize.NoSuchUserException">
  <http-error error-code="404">
     <message>No such user</message>
  </http-error>
</exception>
```

REST in a few fragments

```
<framework:entity-home name="userHome"
  entity-class="com.socialize.model.User"/>

<resteasy:resource-home name="userResourceHome"
  path="/users"
  entity-home="#{userHome}"
  entity-id-class="java.lang.Long"
  media-types="application/xml application/json"
  readonly="false"/>

<resteasy:resource-query name="userResourceQuery"
  path="/users"
  entity-class="com.socialize.model.User"
  media-types="application/xml application/json"/>
```

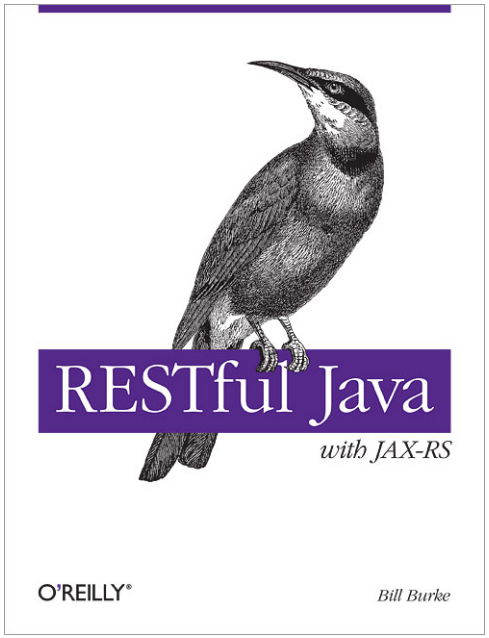
Socialize demo

Summary and call to action

- **REST is why the web works**
  - Addressable resources
  - Uniformed-interface
  - Representation-oriented
  - Stateless communication
- **Expose your data in a standard way**
  - Don’t ball it up inside web pages!
- **Embrace simplicity → Seam+RESTEasy**

Must reads

RESTful Web Services
Richardson & Ruby
O'Reilly



RESTful Java with JAX-RS
Bill Burke
O'Reilly

jboss.org

GET /questions? HTTP/1.1

jboss.org

Presentation resources

- **JSR-311**
  - http://jcp.org/en/jsr/detail?id=311
- **RESTEasy**
  - http://jboss.org/resteasy
- **Seam RESTEasy integration**
  - Web Services chapter of Seam reference guide
- **Bill Burke's REST series on DZone**
  - http://java.dzone.com/articles/putting-java-rest
- **Code samples**
  - http://seaminaction.googlecode.com/svn/demos

jboss.org