Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

Developing RESTful Web Services with JAX-RS

Marc Hadley
Paul Sandoz
Sun Microsystems, Inc

Agenda

- REST and JAX-RS Primer
- Deployment Options
- Demonstration
- Status
- Q & A

Very Short REST Primer: Buy This Book

REST is an Architectural Style

Set of constraints you apply to the architecture of a distributed system to induce desirable properties

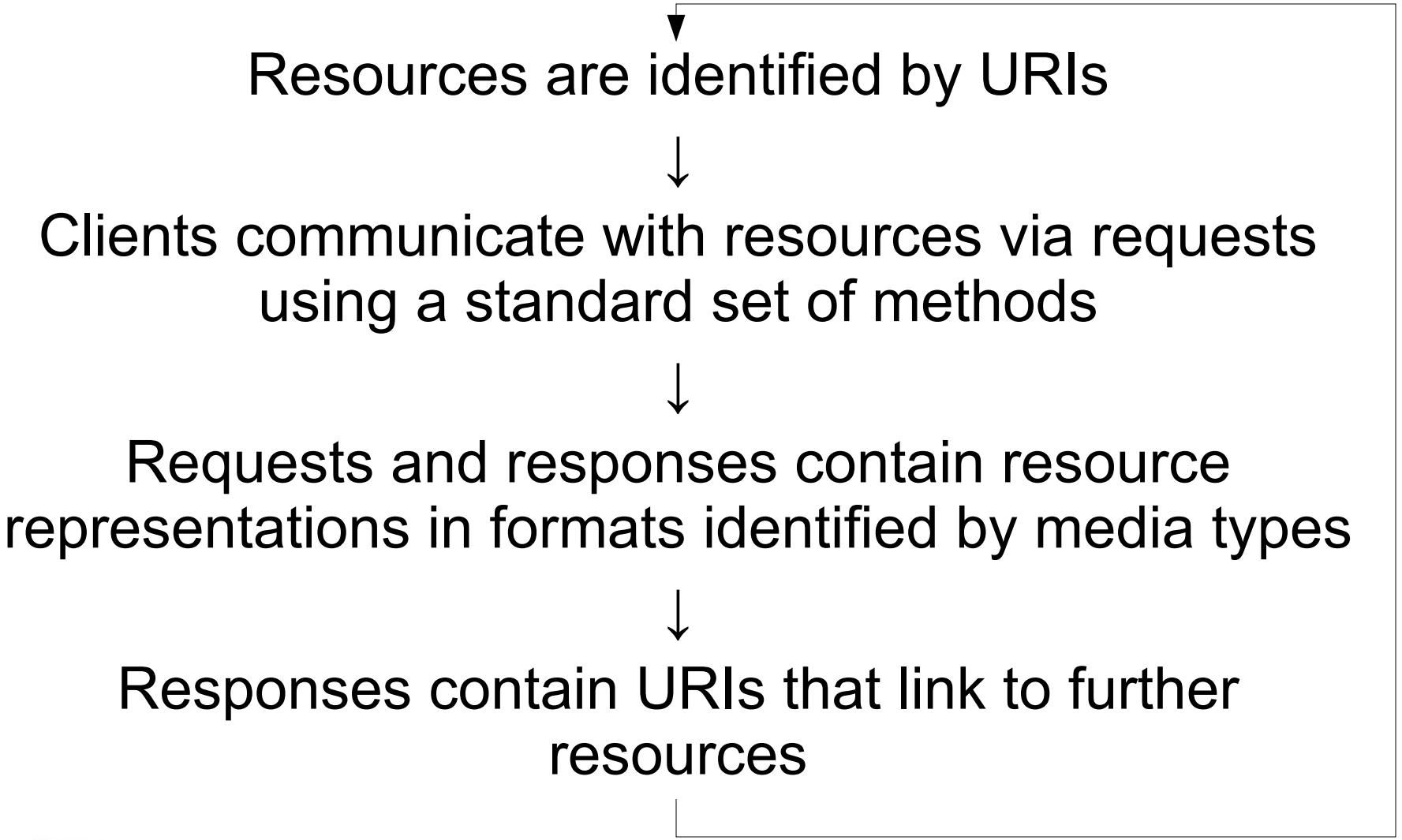
RESTful Web Services

Application of REST architectural style to services that utilize Web standards (URIs, HTTP, HTML, XML, Atom, RDF etc.)

Java API for RESTful Web Services (JAX-RS)

Standard annotation-driven API that aims to help developers build RESTful Web services in Java

RESTful Application Cycle



Resources are identified by URIs

http://example.com/widgets/foo

http://example.com/customers/bar

http://example.com/customers/bar/orders/2

http://example.com/orders/101230/customer

Resources are identified by URIs

- Resource == Java class
  - POJO
  - No required interfaces
- ID provided by `@Path` annotation
  - Value is relative URI, base URI is provided by deployment context or parent resource
  - Embedded parameters for non-fixed parts of the URI
  - Annotate class or “sub-resource locator” method

Resources are identified by URIs

```
@Path("properties")
public class SystemProperties {
   @GET
   List<SystemProperty> getProperties(...) {...}

   @Path("{name}")
   SystemProperty getProperty(...) {...}
}
```

Standard Set of Methods

Method	Purpose
GET	Read, possibly cached
POST	Update or create without a known ID
PUT	Update or create with a known ID
DELETE	Remove

Standard Set of Methods

- Annotate resource class methods with standard method
  - `@GET`, `@PUT`, `@POST`, `@DELETE`, `@HEAD`
  - `@HttpMethod` meta-annotation allows extensions, e.g. WebDAV
- JAX-RS routes request to appropriate resource class and method
- Flexible method signatures, annotations on parameters specify mapping from request
- Return value mapped to response

Standard Set of Methods

```
@Path("properties/{name}")
public class SystemProperty {

   @GET
   Property get(@PathParam("name") String name)
     {...}

   @PUT
   Property set(@PathParam("name") String name,
     String value) {...}

}
```

Resource Representations

- Representation format identified by media type. E.g.:
  - XML - application/properties+xml
  - JSON - application/properties+json
  - (X)HTML+microformats - application/xhtml+xml
- JAX-RS automates content negotiation
  - **`GET /foo`**
    `Accept: application/properties+json`

Resource Representations
Static and dynamic content negotiation

- Annotate methods or classes with static capabilities
  - **`@Produces`**, **`@Consumes`**
- Use **`Variant`**, **`VariantListBuilder`** and **`Request.selectVariant`** for dynamic capabilities
  - Also supports language and encoding

Resource Representations

```
@GET
@Produces("application/properties+xml")
Property getXml(@PathParam("name") String name) {
   ...
}

@GET
@Produces("text/plain")
String getText(@PathParam("name") String name) {
   ...
}
```

Responses Contain Links

```
HTTP/1.1 201 Created
Date: Wed, 03 Jun 2009 16:41:58 GMT
Server: Apache/1.3.6
Location: http://example.com/properties/foo
Content-Type: application/order+xml
Content-Length: 184

<property self="http://example.com/properties/foo">
  <parent ref="http://example.com/properties/bar"/>
  <name>Foo</name>
  <value>1</value>
</order>
```

Responses Contain Links

- **`UriInfo`** provides information about deployment context, the request URI and the route to the resource
- **`UriBuilder`** provides facilities to easily construct URIs for resources

Responses Contain Links

```
@Context UriInfo i;

SystemProperty p = ...
UriBuilder b = i.getBaseUriBuilder();
URI u = b.path(SystemProperties.class)
    .path(p.getName()).build();

List<URI> ancestors = i.getMatchedURIs();
URI parent = ancestors.get(1);
```

Agenda

- REST and JAX-RS Primer
- Deployment Options
- Demonstration
- Status
- Q & A

Java SE Deployment

- **`RuntimeDelegate`** is used to create instances of a desired endpoint class
- Application supplies configuration information
  - List of resource classes and providers as subclass of **`Application`**
- Implementations can support any Java type
  - Jersey supports Grizzly (see below) and the LW HTTP server in Sun's JDK.

Example Java SE Deployment

```
Application app = ...
RuntimeDelegate rd = RuntimeDelegate.getInstance();
Adapter a = rd.createEndpoint(app, Adapter.class);

SelectorThread st = GrizzlyServerFactory.create(
    "http://127.0.0.1:8084/", a);
```

Servlet

- JAX-RS application packaged in **`WAR`** like a servlet
- For JAX-RS aware containers
  - **`web.xml`** can point to **`Application`** subclass
- For non-JAX-RS aware containers
  - **`web.xml`** points to implementation-specific **`Servlet`**; and
  - an **`init-param`** identifies the **`Application`** subclass
- Resource classes and providers can access **`Servlet`** request, context, config and response via injection

Java EE

- Resource class can be an EJB session or singleton bean
- Providers can be an EJB stateless session or singleton bean
- JAX-RS annotations on local interface or no-interface bean
- If JCDI (JSR 299) also supported then
  - Resource classes can be JCDI beans
  - Providers can be JCDI beans with application scope
- Full access to facilities of native component model, e.g. resource injection

Implementations

- Jersey
- Restlet
- JBoss RESTEasy
- Apache CXF
- Triaxrs
- Apache Wink

Agenda

- REST and JAX-RS Primer
- Deployment Options
- Demonstration
- Status
- Q & A

Agenda

- REST and JAX-RS Primer
- Deployment Options
- Demonstration
- Status
- Q & A

Status

- 1.0 Finalized in October 2008
- 1.1 maintenance release underway
  - Spec completed in March 2009
  - RI and TCK ready soon
- 2.0 planning underway
  - Client API
  - Quality of source
  - Form data and MIME multipart
  - Declarative hyperlinking
  - Representation templating

Agenda

- REST and JAX-RS Primer
- Deployment Options
- Demonstration
- Status
- Q & A

For More Information

- Official JSR Page
  - http://jcp.org/en/jsr/detail?id=311
- JSR Project
  - http://jsr311.dev.java.net/
- Reference Implementation
  - http://jersey.dev.java.net/
- Blogs
  - http://weblogs.java.net/blog/mhadley/
  - http://blogs.sun.com/sandoz/
  - http://blogs.sun.com/japod/

Thank You

Marc Hadley
marc.hadley@sun.com

Paul Sandoz
paul.sandoz@sun.com