ReSTful OSGi Web Applications Tutorial

Khawaja Shams & Jeff Norris
California Institute of Technology, Jet Propulsion Laboratory
AGENDA

• Who are we and how did we get here?
• Overview of key technologies
• Brief demo of tutorial application (ReSTBots)
• Tutorial exercises
• Best practices
• Conclusion
Who are we?

- Developers of spacecraft operations tools from NASA
- Long-time users of the Eclipse Rich Client Platform
- Recent users of server-side Equinox
- New fans of ReSTful web application development
One of our rich client products: Maestro
What brought us to ReST & Equinox: Migrating capabilities to the server

- Data searcher
- Search results view
- Image view
- Time functions

Performance degraded by poor throughput
What brought us to ReST & Equinox:
Migrating capabilities to the server

Client

- plugins:
  - Data searcher
  - Search results view
  - Image view
  - Time functions

Internet

Server

- Data
- Data searcher
- Time functions
- Report generation
Eclipse RCP + Server-side Equinox = “Tierless Computing”

• Develop many plugins independent of deployment environment
• Share some capabilities between server and client
• Freely migrate capabilities back and forth as needed
• Use a consistent development environment (Eclipse) and component model (OSGi) throughout
• Debug clients and servers simultaneously!
Why not SOAP?

- SOAP and ReST are both viable ways to deploy web apps.
- *In our experience*, ReST services have been easier to develop and, more importantly, easier for others to use.
- Your mileage may vary!

Our perception of the difficulty of using web services

- ReST
- SOAP
Overview of Key Technologies
ReST and **Resource** Oriented Architectures

• Applications divided into resources (nouns), not services (verbs)
• Relies on HTTP methods to define operations on resources
• Communicate through exchanging representations of resources: **Re**presentational **State** **Transfer**
• Stateless and asynchronous (just like HTTP)

**Question:** How would you make these servlet URLs ReSTful?

http://foobar.com/viewUserDetails?userId=123
http://foobar.com/addNewUser?userId=456&name="Jeff"&team="blue"
http://foobar.com/findUsers?nameContains="Je"
ReST leverages http

• Standard, well-documented protocol for communication between a client and server
• URIs for everything
  ▸ Universal addressability
  ▸ Easy linking among resources
• Provides the essential CRUD operations on resources
  ▸ PUT, GET, POST, DELETE
• Cacheable
• Easy to use in every programming language (even scripting languages)
• Easy to test from a web browser (esp. Firefox with Poster plugin)

Question: When should you use POST instead of PUT?
Reporting status from ReSTful applications

- HTTP Status Codes provide a rich, standardized language for describing the server’s response to a request. Examples:
  
  - **Successful Codes (2XX):**
    - 200 OK
    - 201 Created
    - 202 Accepted
    - 204 No Content
  
  - **Redirection Codes (3XX):**
    - 301 Moved Permanently
    - 304 Not Modified
  
  - **Client Error (4xx):**
    - 400 Bad Request
    - 401 Unauthorized
    - 403 Forbidden
    - 404 Not Found
    - 405 Method Not Allowed
  
  - **Server Errors (5xx):**
    - 500 Internal Server Error

See [http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

Question: What code would you use to indicate that the client included invalid characters in the request? To report that the server ran out of memory? To reject a deletion request?
Equinox / OSGi

• The same plugin model as the RCP
• The same development environment as the RCP
• Inspection of running programs via an interactive console
• Dynamic extensibility - add new plugins without restarting server
• Scoping of modules
Restlet (http://www.restlet.org/)

- API developed by Noelios Consulting
- Streamlines the development of ReSTful webapps
- Addresses some limitations of traditional servlets
  - Provides powerful URI mapping and parsing capabilities
  - Decouples IO from web applications

```
http request → Restlet Router → Restlet → Restlet → Restlet
```
Ensemble ReST Framework

• Simply an integration of the Restlet API with Equinox, developed by NASA
• Allows easy definition of new Restlet Resources via extension points
• Provides some convenience utilities for handling requests
• Open source! All code provided in this tutorial will be available online via the Open Channel Foundation
**Tutorial**

**ENSEMBLE**
ReSTBots

- ReSTBots are simple robot simulations with the following attributes:
  - Name
  - Position (x,y) - the current location of the ReSTBot
  - Goal (x,y) - where the ReSTBot has been commanded to move to
  - Direction (x,y) - current drive direction (velocity vector)
- ReSTBots repel each other like identically charged particles
- ReSTBots can be represented in XML

```
<Restbot name="Foo"
  x="1" y="2"
  gx="3" gy="4"
  dx=".1" dy=".1"/>
```

```xml
x,y
dx,dy
```

```xml
gx,gy
```
ReSTBot Server

- ReSTful server containing the following resources:
  
  - http://localhost:8180/restbots \(\rightarrow\) RestbotListingResource
    - All ReSTBots on the server
  
  - http://localhost:8180/restbots/{name} \(\rightarrow\) RestbotResource
    - A single named ReSTBot
  
  - http://localhost:8180/restbots/{name}/pic \(\rightarrow\) RestbotPictureResource
    - The picture for a named ReSTBot

Question: What would you expect to happen if you performed a PUT on the second resource URL? How would you interpret a return code of 409 (CONFLICT)?
ReSTBot execution loops

• ReSTBots execute this loop:
   GET current position and goal
   Compute a new direction and POST it

• The ReSTBot server executes this loop:
   For each ReSTBot:
     Access its current position and direction
     Compute a new position based on the direction

• Other clients can inject new goals for ReSTBots

Yes, there is a synchronization problem here. Correcting it would have made this tutorial a lot more complicated.
Overview of tutorial application

- Restbot
- Restbot Installer
- Web Browser
- Restbot Server

- POST direction to server
- GET position, goal from server
- GET robot states from server
- DELETE restbot from server
- GET robot positions from server
- PUT restbot to server
- POST goal to server
Tutorial Plugins

- `gov.nasa.ensemble.core.restlet` — *Ensemble Equinox/ReSTlet System*

- `gov.nasa.ensemble.restbots.client`  
  `gov.nasa.ensemble.restbots.common`  
  `gov.nasa.ensemble.restbots.server`  
  *Example plugins developed specifically for this tutorial*

- `org.apache.commons.codec`  
  `org.apache.commons.httpclient`  
  `org.apache.log4j`  
  `org.apache.xalan-j`  
  *Third-party utilities for http, logging, and XML translation*

- `org.restlet` — *Noelios ReSTlet API*
Exercise 0: Launch the ReSTBot System

• Switch to “workspace_0”

• Goals:
  ♦ Create and start a runtime config for the ReSTBot Server
  ♦ Start RestbotClient to visualize the arena
  ♦ Start RestbotInstaller to add robots to the arena

• You’ll need this command-line argument:
3: Type Name “ReSTBot Server”

4: Scroll down, add org.eclipse...jetty
VM arguments:
-Declipse.ignoreApp=true -Dosgi.noShutdown=true
Startup Procedure

1. First, stop all running applications
2. Start First
3. Start Second
4. Start Third

Normal Startup Output

ReSTBot Server [OSGi Framework] /System/Library/Frameworks/javaVM.framework/Versions/1.app

INFO: Version Jetty/5.1.x
0 [Start Level Event Dispatcher] INFO gov.nasa.ensemble.core.restlet
INFO: Started org.mortbay.jetty.servlet.ServletHandler@1fa06
INFO: Started HttpContext[/,]
1 [Start Level Event Dispatcher] INFO gov.nasa.ensemble.core.restlet
1 [Start Level Event Dispatcher] INFO gov.nasa.ensemble.core.restlet
INFO: Started SocketListener on 0.0.0.0:8180
INFO: Started org.mortbay.http.HttpServer@472b3c
The OSGi Console

- Great tool for managing your OSGi environment
- Allows you to diagnose, install, uninstall, start, stop, update bundles
- This functionality can be turned on when deploying to a web application container (see web.xml)
- Similar functionality available through a web interface:
  - Knopflerfish http_console
Firefox Poster Plugin (Testing)
Exercise 1

• Switch to “workspace_1”

• Goals:
  • Learn how to accept updates to a resource
  • Write a client that updates a resource

• Classes to modify:
  • gov.nasa.ensemble.restbots.server.RestbotResource
    ▪ handlePost() - accept updates to ReSTBot direction and goal
  • gov.nasa.ensemble.restbots.common.Restbot
    ▪ updateGoal() - POST updated goal based on mouse click in RestbotClient view
@Override
public void handlePost() {
    String name = getRequestParameter("name");
    Restbot restbot = Restbot.getRestbot(name);

    try {
        if (restbot == null) {
            // TODO send proper response if restbot is not found
            getResponse().setStatus(Status.CLIENT_ERROR_NOT_FOUND, "No restbot with this name exists");
            return;
        }
        InputStream inputStream = getRequest().getEntity().getStream();
        if (restbot.updateFromXML(inputStream)) {
            getResponse().setStatus(Status.SUCCESS_OK, "Restbot updated");
        } else {
            getResponse().setStatus(Status.CLIENT_ERROR_BAD_REQUEST, "Check the XML in the body");
        }
    } catch (ParserConfigurationException e) {
        trace.error(e);
        getResponse().setStatus(Status.CLIENT_ERROR_BAD_REQUEST, e.getMessage());
    } catch (NumberFormatException nfe) {
        trace.error(nfe);
        getResponse().setStatus(Status.CLIENT_ERROR_BAD_REQUEST, nfe.getMessage());
    } catch (IllegalStateException ise) {
        trace.info(ise);
        getResponse().setStatus(Status.CLIENT_ERROR_BAD_REQUEST, "The restbot name does in the body does not match the URI");
    } catch (IOException e) {
        trace.error(e);
        getResponse().setStatus(Status.SERVER_ERROR_INTERNAL, "IOException while reading from input stream");
    }
}
/**
 * This method is used to update the server about the direction of the
 * rover. The server may give consideration to the direction of the rover
 * when calculating the new position of the rover. Before updating the
 * server, the restbot will calculate its direction based on its current
 * position and the goal position.
 */

public void updateGoal() {
    Document doc;
    HttpClient client = HttpClientUtils.getClient();
    PostMethod post = new PostMethod(resbotURI);
    try {
        doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
        Element restbotElement = doc.createElement("restbot");
        doc.appendChild(restbotElement);
        restbotElement.setAttribute(Restbot.NAME_ATTR, getName());
        restbotElement.setAttribute(Restbot.GOAL_X_ATTR, String.valueOf(getGoalX()));
        restbotElement.setAttribute(Restbot.GOAL_Y_ATTR, String.valueOf(getGoalY()));
        String docAsStr = Restbot.serializeDocument(doc);
        StringRequestEntity requestEntity = new StringRequestEntity(docAsStr, "text/xml", "UTF-8);
        post.setRequestEntity(requestEntity);
        int resp = client.executeMethod(post);
        trace.info("Updated the server with my goal and received response:" + resp);
    } catch (ParserConfigurationException e) {
        trace.error(e);
    } catch (IOException e) {
        trace.error(e);
    } finally {
        post.releaseConnection();
    }
}
Exercise 2: Your own resource

• Switch to “workspace_2”
• Goals:
  ◦ Create, register, and develop a new resource
  ◦ Leverage the ReSTlet API for operating on the resource
  ◦ Use HTTP status codes
• Tasks:
  ◦ Create a new resource
  ◦ Register the resource through extension point
  ◦ Implement required methods
  ◦ Modify client to access the resource
  ◦ Add status codes to your resource
  ◦ Modify client to interpret and handle status codes
Adding a new Resource

• Open plugin.xml from gov.nasa.ensemble.restbots.server
• Switch to the Extensions tab
• Right-click on Extension point and add new Resource

- Fill out details
- Recommended class name: gov.nasa.jpl.maestro.restbots.server.RestbotPictureResource
• Move new extension point to top
RestbotPictureResource.handleGet()

```java
@Override
public void handleGet() {
    String restbotName = getRequestParameter("name");
    Restbot restbot = Restbot.getRestbot(restbotName);
    if (restbot == null) {
        getResponse().setStatus(Status.CLIENT_ERROR_NOT_FOUND, "No restbot with this name exists");
        return;
    }
    byte[] image = restbot.getImageData(false);
    if (image == null) {
        getResponse().setStatus(Status.CLIENT_ERROR_NOT_FOUND, "This restbot does not have a picture");
        return;
    }
    getResponse().setEntity(new ByteArrayRepresentation(image));
}
```
RestbotPictureResource.handlePut()

@Override
public void handlePut() {
    String restbotName = getRequestParameter("name");
    Restbot restbot = Restbot.getRestbot(restbotName);
    if (restbot == null) {
        getResponse().setStatus(Status.CLIENT_ERROR_NOT_FOUND, "No restbot with this name exists");
        return;
    }
    try {
        InputStream stream = getRequest().getEntity().getStream();
        ByteArrayOutputStream bout = new ByteArrayOutputStream(stream.available());
        byte[] buf = new byte[stream.available()];
        int len = 0;
        while ((len = stream.read(buf))>= 0){
            bout.write(buf, 0, len);
        }
        restbot.setImageData(bout.toByteArray());
    } catch (IOException e) {
        trace.error(e);
        getResponse().setStatus(Status.SERVER_ERROR_INTERNAL, e.getMessage());
    }
}
RestbotPictureResource.handleDelete()

@Override
public void handleDelete() {
    String restbotName = getRequestParameter("name");
    Restbot restbot = Restbot.getRestbot(restbotName);
    if (restbot == null) {
        getResponse().setStatus(Status.CLIENT_ERROR_NOT_FOUND, "No restbot with this name exists");
        return;
    }
    restbot.setImageData(null);
}
Tips, Tricks, and Best Practices
HTTP Methods

- Safe Methods
  - GET
  - HEAD

- Idempotent methods
  - GET
  - HEAD
  - PUT
  - DELETE

- Unsafe and non-idempotent method:
  - POST

Question: Why is this important?
More on Restlets

• Reslet vs. Resource
• Representations
• Security
  ◆ Guards
  ◆ Built in authentication schemes: HTTP BASIC, AWS Authentication, HTTP DIGEST support coming in Restlet 1.1
  ◆ Extensibility
Securing Your Restlets

- Container managed security vs. Restlet Security
- Guards
  - Available authentication schemes
  - Extending for custom authentication/authorization schemes
- Adding security to Ensemble ReST
- SSL
// How to add security:
Guard guard = new Guard (getContext(), ChallengeScheme.HTTP_BASIC, "EnsembleRealm");
    @Override
    public int authenticate(Request request) {
        // TODO add your own authenticate method unless you know all passwords in advance
        return super.authenticate(request);
    }
    @Override
    protected char[] findSecret(String identifier) {
        // TODO return password for a particular user (lookup in a database, etc)
        return super.findSecret(identifier);
    }
};
guard.setNext(router);
return guard;
Accessing a secured Restlet with HttpClient

• Authentications scheme supported:
  ◦ HTTP BASIC
  ◦ HTTP DIGEST
  ◦ NTLM

• Extensibility
  ◦ Custom authentication schemes allowed
  ◦ Implement the AuthScheme interface
  ◦ Register by invoking AuthPolicy.registerScheme()

• For more information:
  ◦ http://hc.apache.org/httpclient-3.x/authentication.html

⚠️ Alert: Take precautions with custom authentication schemes!!!
Authentication with HttpClient

```java
public void secureClient(HttpClient client) {
    AuthScope authScope = new AuthScope("EnsembleRealm", 8180, "foo.com");
    UsernamePasswordCredentials creds = new UsernamePasswordCredentials("username", "password");
    client.getState().setCredentials(authScope, creds);
}
```

- Create an AuthScope
- Create and set credentials

**Question:** Why is the AuthScope Important?
Apache HttpClient Performance

• Use only one client for your entire application
• Application multithreaded?
  ♦ Use MultiThreadedHttpConnectionManager
• Release a connection after you are done with EACH request; eg: get.releaseConnection()
• Request and Response Streaming
  ♦ InputStreamRequestEntity
  ♦ getResponseBodyAsStream

Question: What if you need numerous simultaneous connections?
Recipe for ReSTful Web Applications

• Identify Resources that you would like to expose
• Address addressability
• Decide which operations should be allowed
• Develop the resources (make extensive use of the status codes)
• Test and Deploy
Conclusion
Ensemble ReST leverages…
Eclipse and OSGi

- **Eclipse**
  - Rapid development with Eclipse (Launch Vehicle)
    - Eclipse Debugger
    - Test application from within IDE
  - Easy export process to production servers

- **OSGi**
  - Modularity in code
  - Runtime extensibility
  - Changes can be limited to specific modules
    - Rapid deployment of modifications
    - Minimizes risks when redeploying
Ensemble ReST leverages ...

HTTP and ReST

• HTTP Protocol
  • Widely supported
    ▪ Programming Languages
    ▪ Web Browsers
  • Resources are completely decoupled
  • Fast performance, especially for binary transfers
  • Standardized authentication and encryption schemes

• ReST
  • Uniform interface to do operations on resources
  • Hierarchical URIs makes writing and consuming sources more intuitive
  • Addressability
  • Statelessness is great for performance
Key Development Considerations

• How modular is your code base? (OSGi)
• How easy is it to access your application? (ReST)
• How hard is it to debug the application (Eclipse)
• What impact does adding a resource have on:
  ◦ Existing clients
  ◦ Existing applications
• How can you test your application?
  ◦ JUnit
  ◦ Firefox Poster Plugin
• How to secure the application and still make it accessible? (HTTP, SSL)
For more information…

• ReSTful Web Services in Perl
  ✷ [link](http://www.onlamp.com/pub/a/onlamp/2008/02/19/developing-restful-web-services-in-perl.html)

• OSGi on the Server Side
  ✷ [link](http://dev2dev.bea.com/pub/a/2007/12/osgi-introduction.html)

• Poster Plugin:
  ✷ [link](https://addons.mozilla.org/en-US/firefox/addon/2691)

• RESTful Web Services