



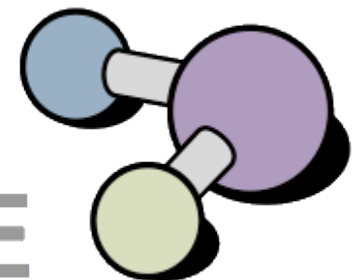
# ReSTful OSGi Web Applications Tutorial

Khawaja Shams & Jeff Norris

California Institute of Technology, Jet Propulsion Laboratory



ENSEMBLE



## AGENDA

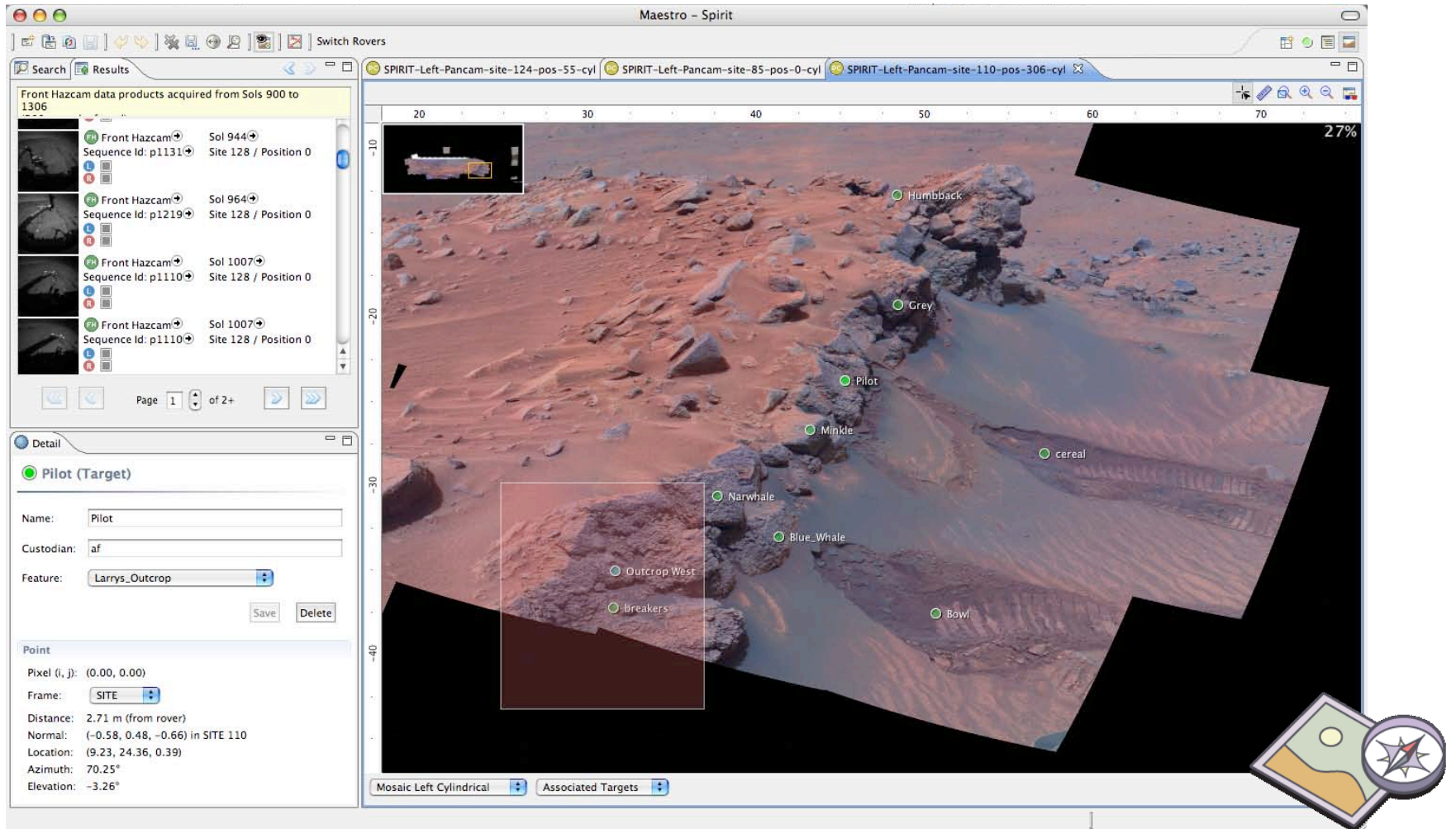
- Who are we and how did we get here?
- Overview of key technologies
- Brief demo of tutorial application (ReSTBots)
- Tutorial exercises
- Best practices
- Conclusion

## Who are we?

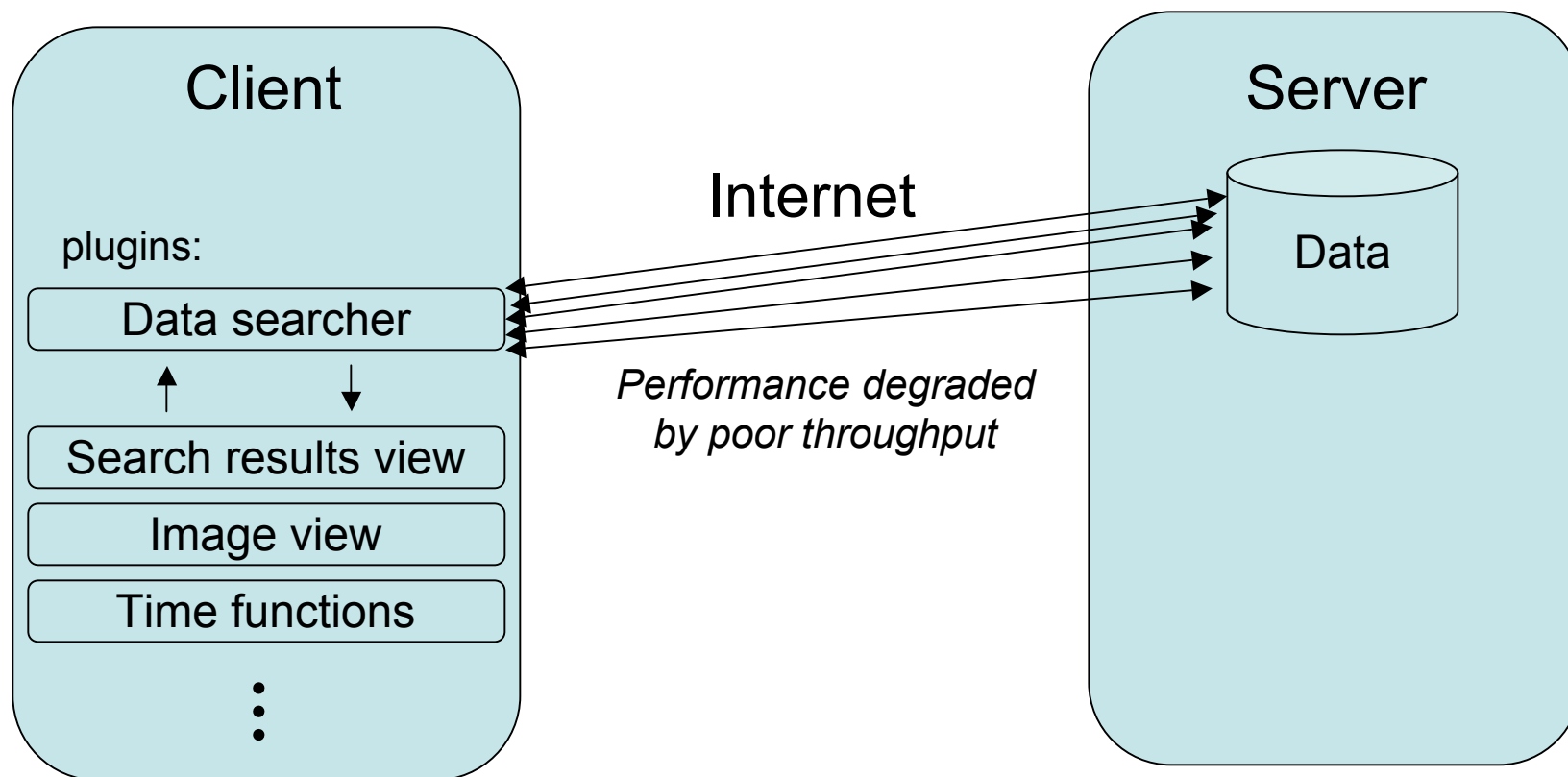
- Developers of spacecraft operations tools from NASA
- Long-time users of the Eclipse Rich Client Platform
- Recent users of server-side Equinox
- New fans of ReSTful web application development



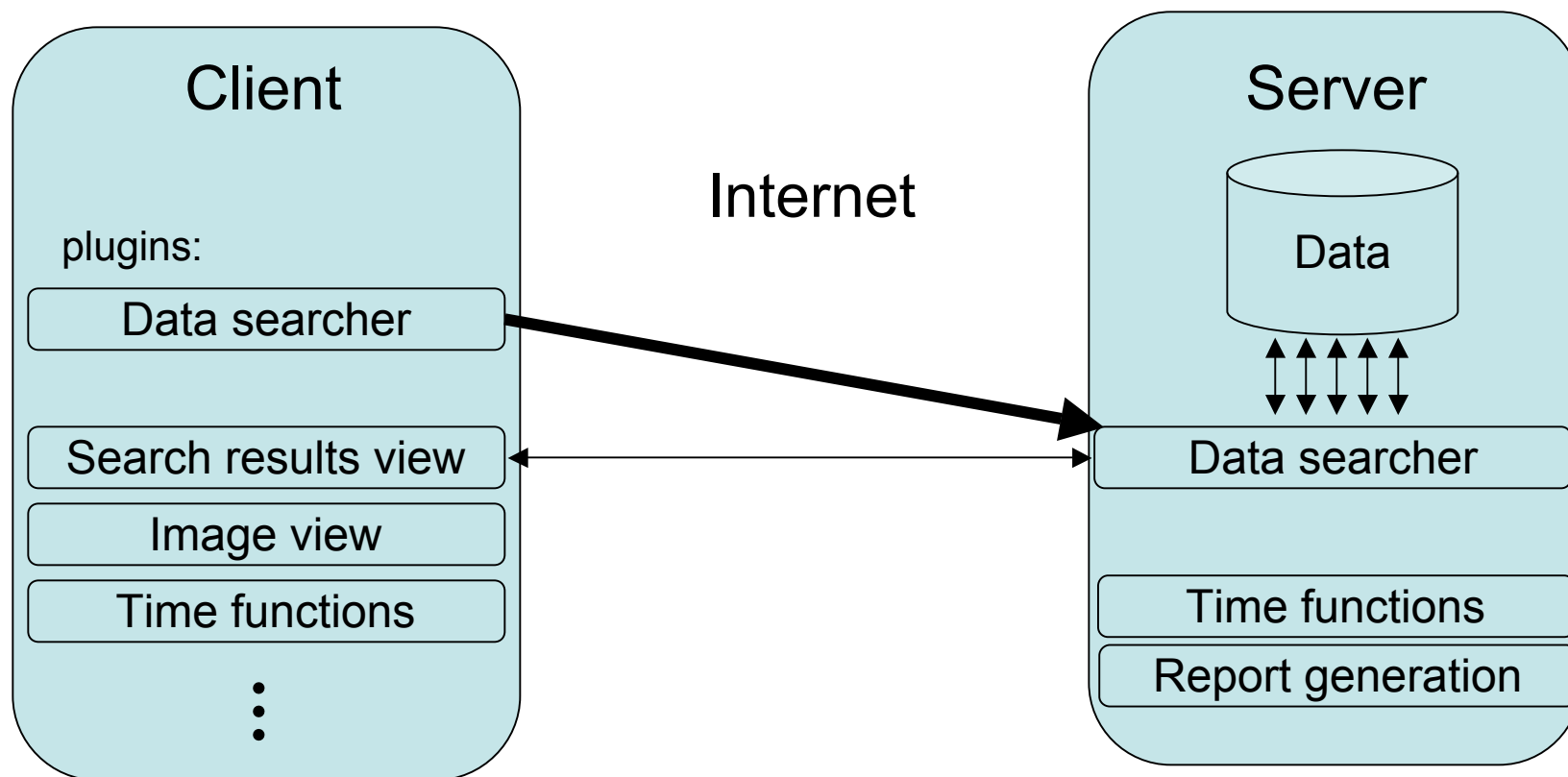
# One of our rich client products: Maestro



## What brought us to ReST & Equinox: Migrating capabilities to the server



## What brought us to ReST & Equinox: Migrating capabilities to the server



## Eclipse RCP + Server-side Equinox = “Tierless Computing”

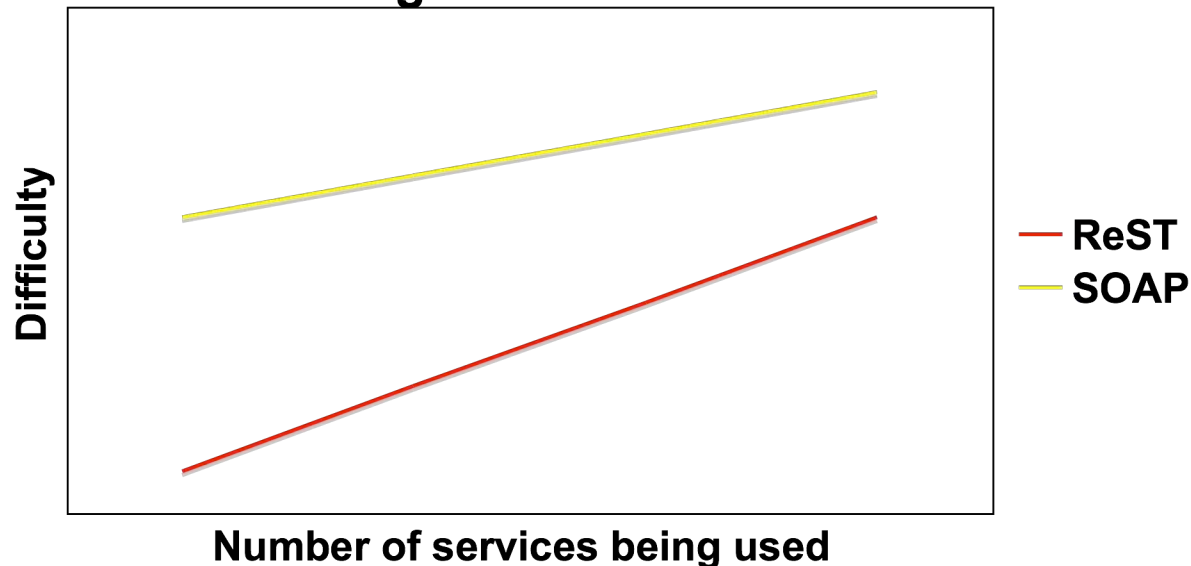
- Develop many plugins independent of deployment environment
- Share some capabilities between server and client
- Freely migrate capabilities back and forth as needed
- Use a consistent development environment (Eclipse) and component model (OSGi) throughout
- Debug clients and servers simultaneously!



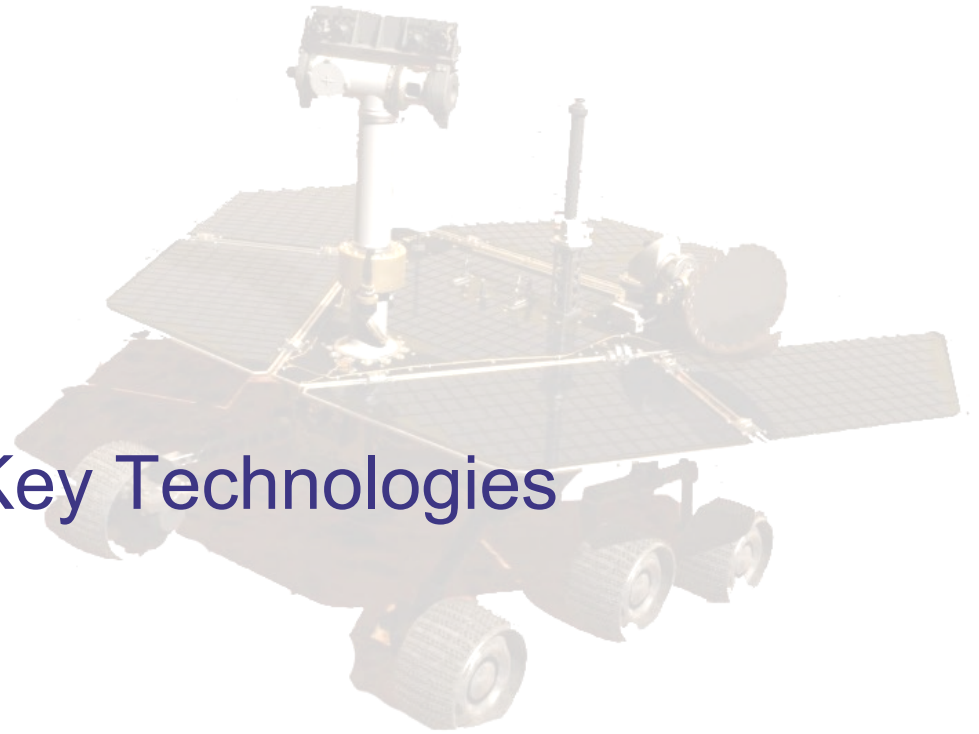
## Why not SOAP?

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

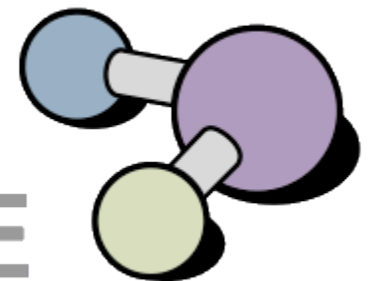
**Our perception of the difficulty of using web services**



## Overview of Key Technologies



**ENSEMBLE**



## 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 : **Representational 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/addNewUser?userId=456&name='Jeff'&team='blue')*

*[http://foobar.com/findUsers?nameContains="Je"](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>



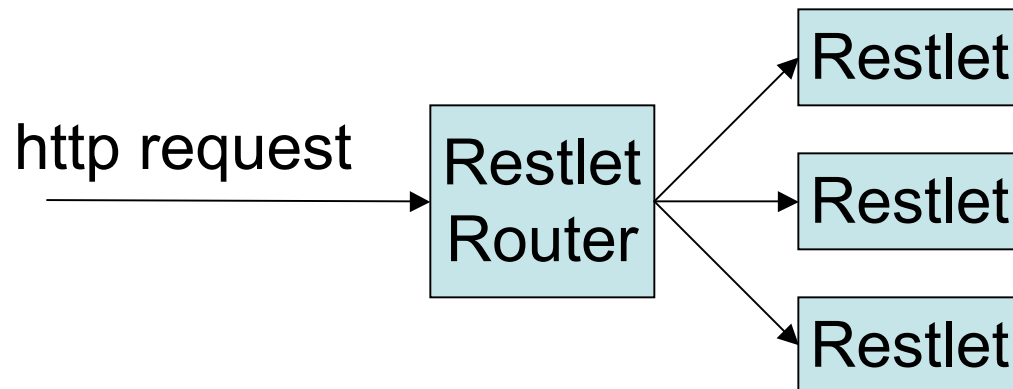
*Question: What code would you use to indicate that the client included invalid characters in the request? To report that the server ran out of memory? To reject a deletion request?*

## 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/>)

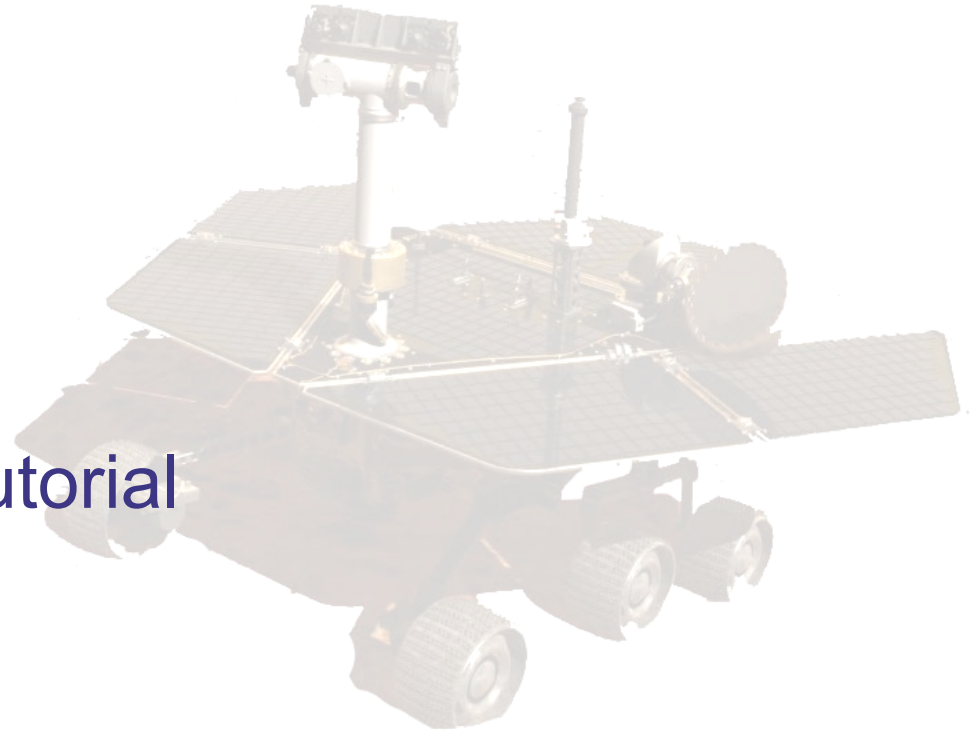
- API developed by Noelios Consulting
- Streamlines the development of ReSTful webapps
- Addresses some limitations of traditional servlets
  - ◆ Provides powerful URI mapping and parsing capabilities
  - ◆ Decouples IO from web applications



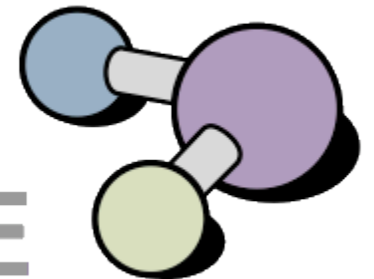
## Ensemble ReST Framework

- 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



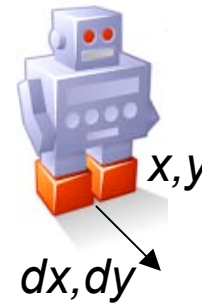
# ENSEMBLE



## 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" />
```



~~X~~ gx,gy

## ReSTBot Server

- ReSTful server containing the following resources:
- `http://localhost:8180/restbots` → *RestbotListingResource*
  - ◆ All ReSTBots on the server
- `http://localhost:8180/restbots/{name}` → *RestbotResource*
  - ◆ A single named ReSTBot
- `http://localhost:8180/restbots/{name}/pic` → *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)?*

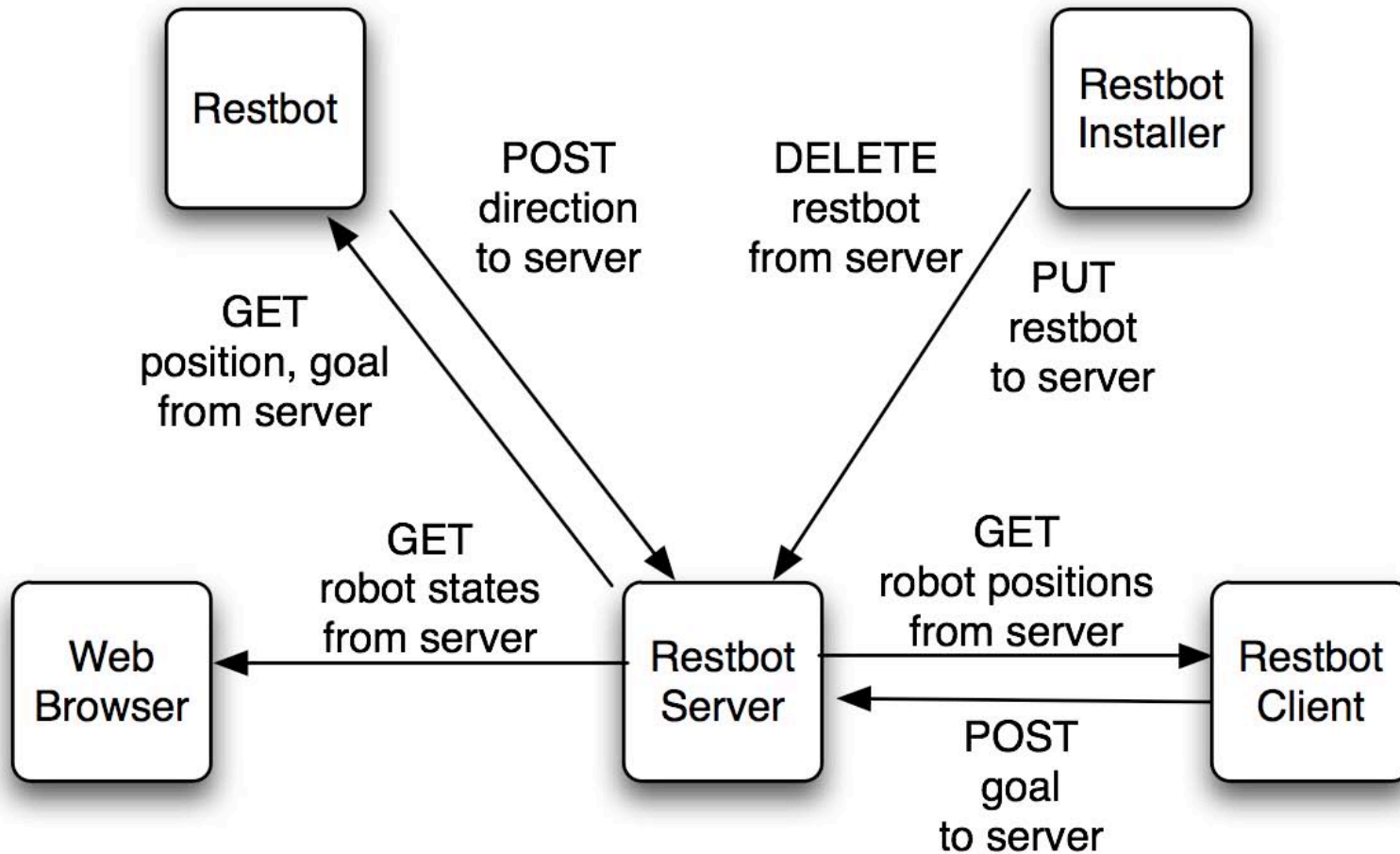
## ReSTBot execution loops

- 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



## 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:
  - ◆ `-Dorg.eclipse.equinox.http.jetty.http.port=8180`

**Run**  
Create, manage, and run configurations  
Create a configuration to launch the OSGi framework.

**2** [Icons]

**1** [Tree View]

**3: Type Name "ReSTBot Server"** [Name: New\_configuration]

**8** [Bundles Arguments Settings Tracing Environment Common]

**5** [Bundles Table]

Bundles	Start Level	Auto-Start
Workspace		
<input checked="" type="checkbox"/> gov.nasa.ensemble.core.restlet (1.0.0)	default	default
<input checked="" type="checkbox"/> gov.nasa.ensemble.restbots.client (1.0.0)	default	default
<input checked="" type="checkbox"/> gov.nasa.ensemble.restbots.common (1.0.0)	default	default
<input checked="" type="checkbox"/> gov.nasa.ensemble.restbots.server (1.0.0)	default	default
<input checked="" type="checkbox"/> org.apache.commons.codec (1.3.0)	default	default
<input checked="" type="checkbox"/> org.apache.commons.httpclient (3.1.0)	default	default
<input checked="" type="checkbox"/> org.apache.log4j (1.2.11)	default	default
<input checked="" type="checkbox"/> org.apache.xalan_j (1.0.0)	default	default
<input checked="" type="checkbox"/> org.eclipse.equinox.http.servletbridge (1.0.0.qualifier)	default	default
<input checked="" type="checkbox"/> org.eclipse.equinox.servlet.bridge.http (3.2.0.qualifier)	default	default
<input checked="" type="checkbox"/> org.eclipse.equinox.servlet.bridge.launcher (3.2.0.qualifier)	default	default
<input checked="" type="checkbox"/> org.eclipse.equinox.servletbridge (1.0.0.qualifier)	default	default
<input checked="" type="checkbox"/> org.restlet (1.0.0)	default	default

**4** [Select All, Deselect All, Add Working Set..., Add Required Bundles, Restore Defaults]

**7** [Add Required Bundles]

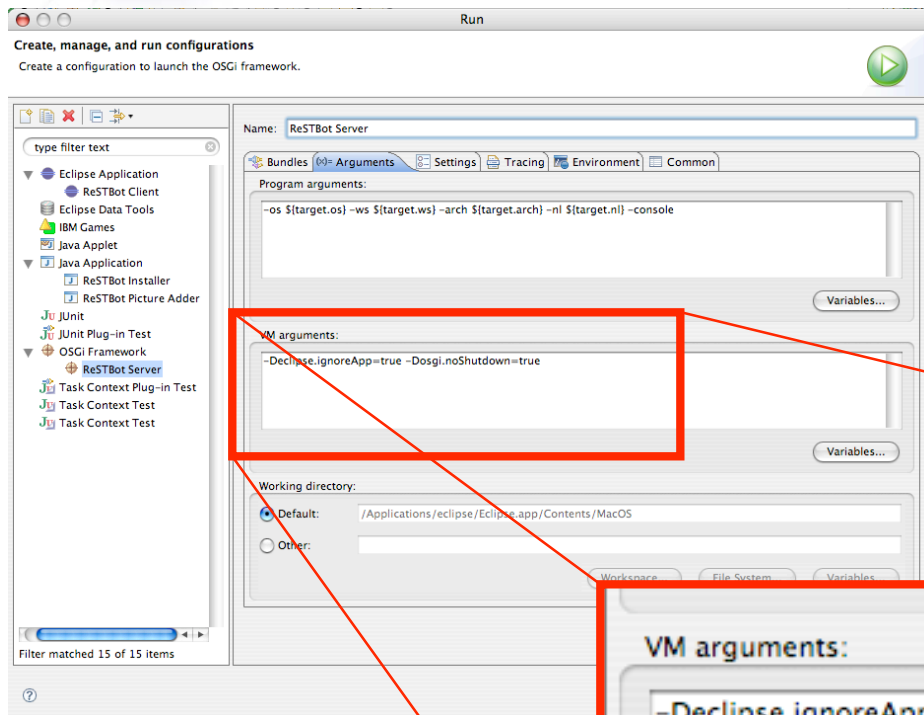
**6: Scroll down, add org.eclipse...jetty** [Target Platform]

Include optional dependencies when computing required bundles  
 Add new workspace bundles to this launch configuration automatically  
 Validate bundles automatically prior to launching

[Validate Bundles] [Apply] [Revert] [Close] [Run]

Filter matched 15 of 15 items

# eclipseCON™ 2008



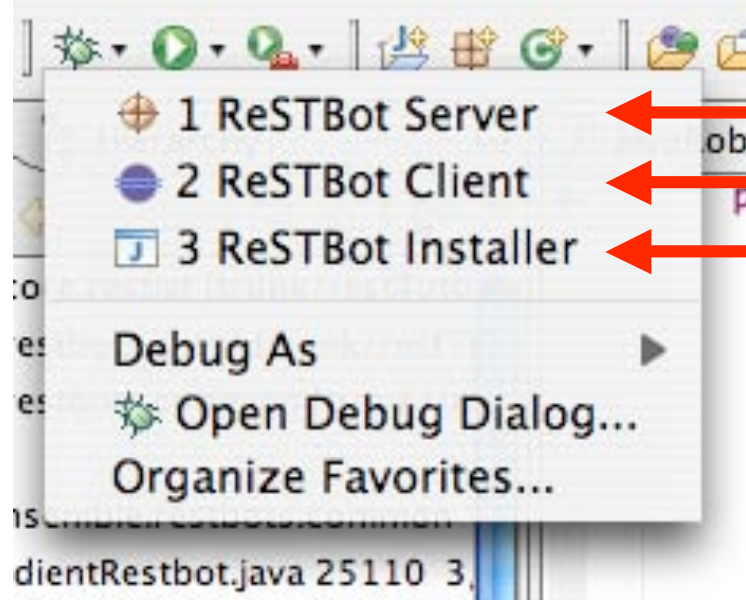
**VM arguments:**

```
-Declipse.ignoreApp=true -Dosgi.noShutdown=true  
-Dorg.eclipse.equinox.http.jetty.http.port=8180
```

# Startup Procedure



*First, stop all running applications*

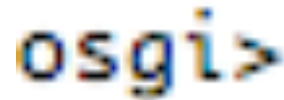


- ← *Start First*
- ← *Start Second*
- ← *Start Third*

*Normal Startup Output* →

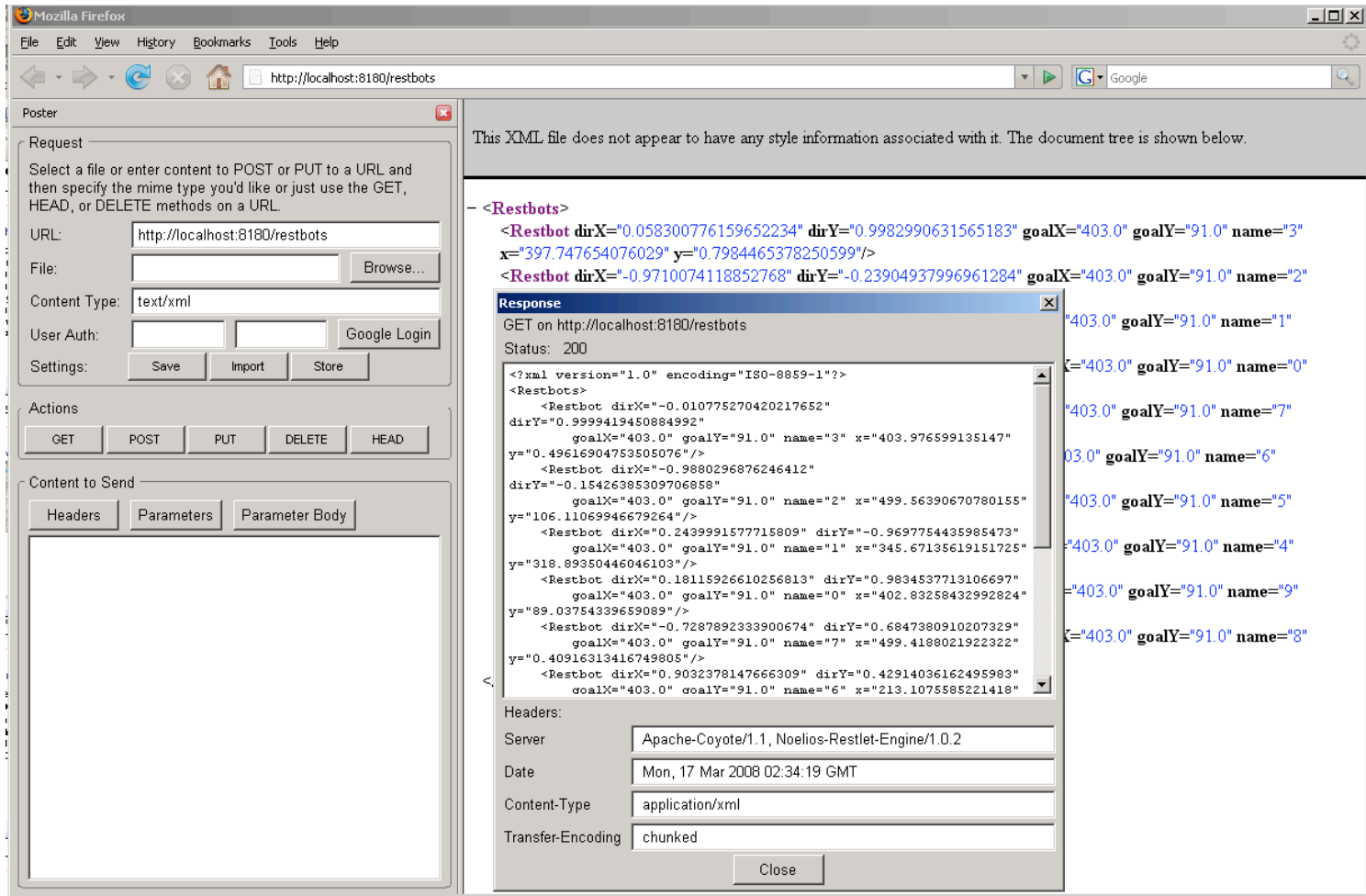
```
REStBot Server [OSGi Framework] /System/Library/Frameworks/JavaVM.framework/Versio
osgi> Mar 16, 2008 11:57:22 PM org.mortbay.http.HttpServer doStart
INFO: Version Jetty/5.1.x
0 [Start Level Event Dispatcher] INFO gov.nasa.ensemble.core.restlet
Mar 16, 2008 11:57:22 PM org.mortbay.util.Container start
INFO: Started org.mortbay.jetty.servlet.ServletHandler@1fa06
Mar 16, 2008 11:57:22 PM org.mortbay.util.Container start
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
Mar 16, 2008 11:57:22 PM org.mortbay.http.SocketListener start
INFO: Started SocketListener on 0.0.0.0:8180
Mar 16, 2008 11:57:22 PM org.mortbay.util.Container start
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)



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<Restbots>
  <Restbot dirX="0.058300776159652234" dirY="0.9982990631565183" goalX="403.0" goalY="91.0" name="3"
  x="397.747654076029" y="0.7984465378250599"/>
  <Restbot dirX="-0.9710074118852768" dirY="-0.23904937996961284" goalX="403.0" goalY="91.0" name="2"
  x="403.0" goalY="91.0" name="1"
  x="403.0" goalY="91.0" name="0"
  x="403.0" goalY="91.0" name="7"
  x="403.0" goalY="91.0" name="6"
  x="403.0" goalY="91.0" name="5"
  x="403.0" goalY="91.0" name="4"
  x="403.0" goalY="91.0" name="9"
  x="403.0" goalY="91.0" name="8"
  </Restbots>
  
```

Response

```

GET on http://localhost:8180/restbots
Status: 200

<?xml version="1.0" encoding="ISO-8859-1"?>
<Restbots>
  <Restbot dirX="-0.010775270420217652"
  dirY="0.9999419450884992"
  goalX="403.0" goalY="91.0" name="3" x="403.976599135147"
  y="0.49616904753505076"/>
  <Restbot dirX="-0.9880296876246412"
  dirY="-0.15426385309706858"
  goalX="403.0" goalY="91.0" name="2" x="499.56390670780155"
  y="106.11069946679264"/>
  <Restbot dirX="0.2439991577715809" dirY="-0.9697754435985473"
  goalX="403.0" goalY="91.0" name="1" x="345.67135619151725"
  y="318.89350446046103"/>
  <Restbot dirX="0.18115926610256813" dirY="0.9834537713106697"
  goalX="403.0" goalY="91.0" name="0" x="402.83258432992824"
  y="89.03754339659089"/>
  <Restbot dirX="-0.7287892333900674" dirY="0.6847380910207329"
  goalX="403.0" goalY="91.0" name="7" x="499.4188021922322"
  y="0.40916313416749805"/>
  <Restbot dirX="0.9032378147666309" dirY="0.42914036162495983"
  goalX="403.0" goalY="91.0" name="6" x="213.1075585221418"
  
```

Headers:

Server	Apache-Coyote/1.1, Noelios-Restlet-Engine/1.0.2
Date	Mon, 17 Mar 2008 02:34:19 GMT
Content-Type	application/xml
Transfer-Encoding	chunked

## 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

## RestbotResource.handlePost()

```
@Override
public void handlePost() {
    String name = getRequestParamer("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");
    }
}
```

## Restbot.updateGoal()

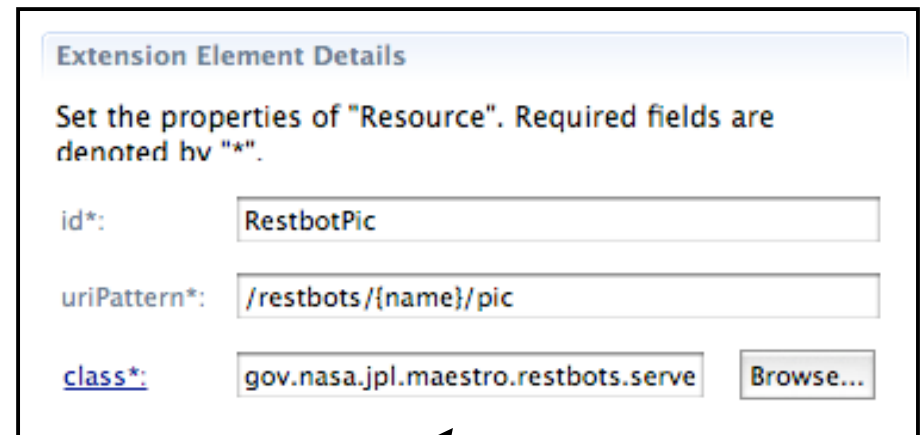
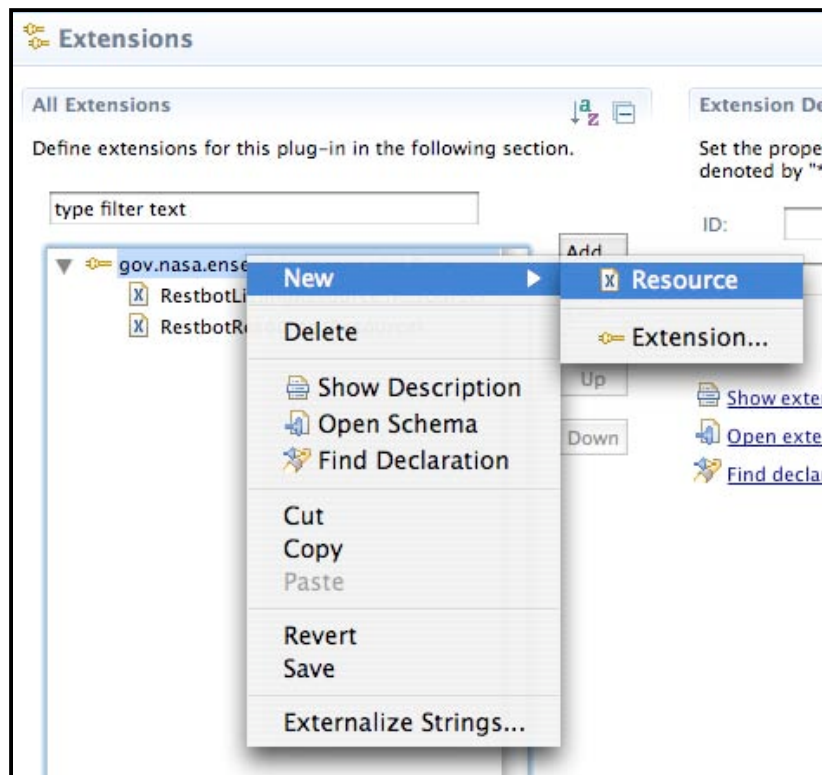
```
/**
 * This method is used to update the server about the direction of the
 * rover. The server may give consideration to the direction of the rover
 * when calculating the new position of the rover. Before updating the
 * server, the restbot will calculate its direction based on its current
 * position 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()

@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()

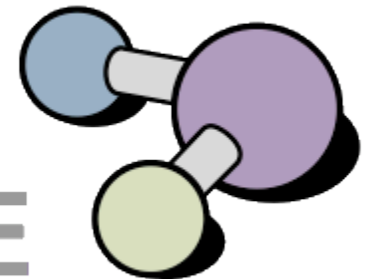
@Override

```
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



**ENSEMBLE**



## 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

```
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 MultiThreadedHttpClientConnectionManager
- Release a connection after you are done with EACH request; eg: `get.releaseConnection()`
- Request and Response Streaming
  - ◆ `InputStreamRequestEntity`
  - ◆ `getResponseBodyAsStream`

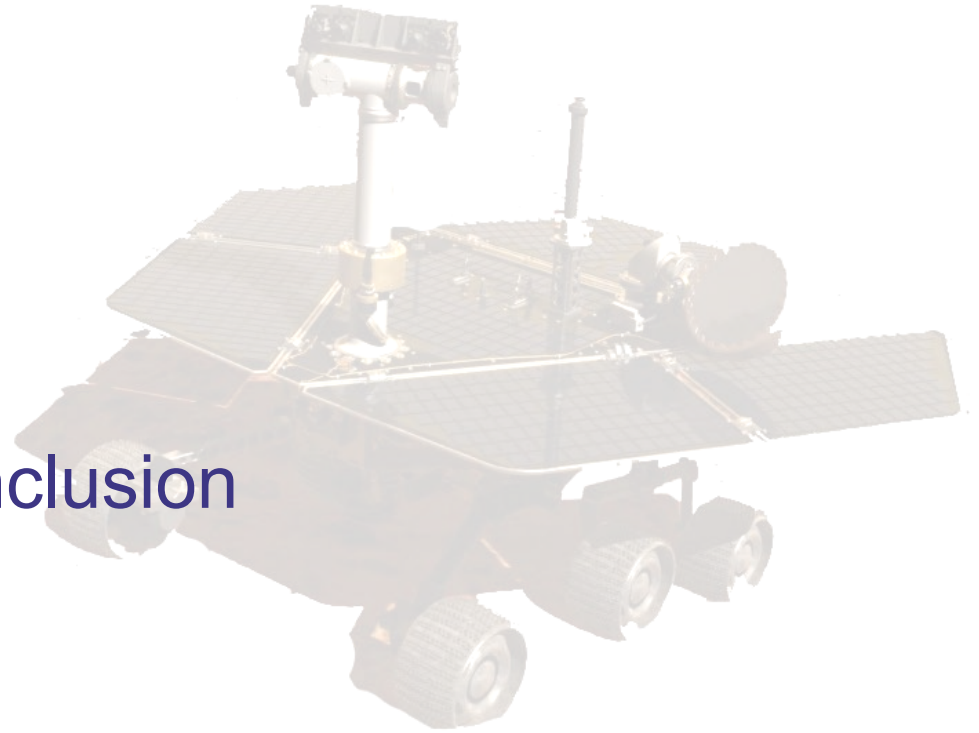


*Question: What if you need numerous simultaneous connections?*

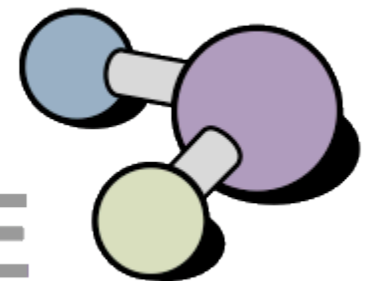
## Recipe for ReSTful Web Applications

- 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**



## 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 you 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
  - ◆ <http://www.onlamp.com/pub/a/onlamp/2008/02/19/developing-restful-web-services-in-perl.html>
- OSGi on the Server Side
  - ◆ <http://dev2dev.bea.com/pub/a/2007/12/osgi-introduction.html>
- Poster Plugin:
  - ◆ <https://addons.mozilla.org/en-US/firefox/addon/2691>
- RESTful Web Services