ReSTful OSGi Web Applications Tutorial

Khawaja Shams & Jeff Norris
California Institute of Technology – Jet Propulsion Lab
AGENDA

• Background and discussion on technology
• The architecture that works for us
• Brief Demo of Application
• Tutorial and Exercises
• Best Practices
• Conclusion
Background

Frameworks, Technologies, and Protocol
HTTP

• Standard protocol for communication between a client and server
• URI
  ♦ addressability
  ♦ Hypermedia links
• CRUD operations
  ♦ PUT
  ♦ GET
  ♦ POST
  ♦ DELETE
• Stateless
• Cacheable
ReST and ROA

- Applications divided into resources
- Communicate through exchanging representations of resources: Representational State Transfer
- Statelessness
- Uniform interface
- Addressability
OSGi

- Revolutionary level of modularity
- Dynamic extensibility
- Scoping of modules
Technologies

- Restlet
- Equinox
- Jetty and Apache Tomcat
Motivation

Why we went this route?
Requirements

• Serve a diverse Set of consumers
  ♦ Standalone java, C, C++ applications
  ♦ RCP Applications
  ♦ Perl
  ♦ Shell Scripts
• Collaborative development from three NASA centers (Ensemble)
• Rapid prototyping, development, and deployment of services and clients
• Decoupling Services (Untangling the Web)
• Security
• High performance
Ensemble ReST leverages…
Eclipse and OSGi

• Eclipse
  • Rapid development within Eclipse
  • Eclipse Debugger
  • Test application from within Eclipse
  • 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
Tutorial
Scenario

- Restbots
  - Goal and Direction
  - Charges
- Restbot Arena
- Server Application
- RCP Application
- Java Restbot installer
Exercise 1

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

• Tasks:
  - Modify ReSTlet code to accept updates in Restbot goals
  - Modify RCP application to update Restbot goals on click
What happens to my request after launch?

• HTTP Status Codes
  • 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
What is a resource?

- Addressability
- CRUD
- Statelessness
Exercise 2: Your own resource

• Goals:
  - Learn:
    - How to create, register, and develop a new resource
    - How to leverage ReSTlet API for operating on the resource
    - How to leverage 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
Best Practices
HTTP Methods

• Safe Methods
  • GET
  • HEAD

• Idempotent methods
  • GET
  • HEAD
  • PUT
  • DELETE

• Unsafe and non-idempotent method:
  • POST

• Why this is important?
Apache HTTP Client Performance

• Use only one client for your entire application
• Application multithreaded?
  ♦ Use MultiThreadedHttpConnectionManager
• Release a connection after you are done with the request; eg: get.releaseConnection()
• Request and Response Streaming
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
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
  - [http://www.onlamp.com/pub/a/onlamp/2008/02/19/developing-restful-web-services-in-perl.html](http://www.onlamp.com/pub/a/onlamp/2008/02/19/developing-restful-web-services-in-perl.html)

• OSGi on the Server Side

• RESTful Web Services