OSLC Consumer with Eclipse Lyo Project

Jean-Luc Johnson, AIRBUS Group Innovations
Gray Bachelor, IBM Rational
Samit Mehta, IBM Rational
Harry Reeder, IBM Rational
Your team today

- Jean-Luc Johnson (Airbus Group Innovations)
- Gray Bachelor (IBM Rational CTO Office)
- Samit Mehta (IBM Rational: ISV enablement and Ready for Rational)
- Harry Reeder (IBM Rational UK)
Crystal tool interoperability agenda

- 14:00-14:30 – Introduction and environment setup
- 14:30-15:00 – LAB 1
  - Bugzilla change request resource Java class
  - Discovery system to access the service
  - Summary
- 15:00-15:30 – LAB 2
  - Retrieve list of products in Bugzilla
- 15:30-17:00 – LAB 3
  - Deeper “Hands on” session with Lyo to continue
- 17:00-17:30 – Showcase
  - Java Swing GUI
OSLC Consumer workshop

Introduction and Overview
OSLC Consumer workshop

Initial setup
Initial setup

- **Pre-requisites**
    - Java JDK 1.6 + (Mandatory)
    - Include in the labs
  - Follow the instruction at [http://wiki.eclipse.org/Lyo/BuildOSLC4JBugzilla](http://wiki.eclipse.org/Lyo/BuildOSLC4JBugzilla)
    - Register yourself (Username/password) to [https://landfill.bugzilla.org/bugzilla-4.2-branch/createaccount.cgi](https://landfill.bugzilla.org/bugzilla-4.2-branch/createaccount.cgi)
    - Build the OSLC4JBugzilla service provider application

- **NOTES:**
  - We provide the code of the labs (Including the stable release of Eclipse Lyo)
  - We provide a bundle Tomcat + OSLC4JBugzilla war file
Check list

✓ Unzip the downloaded apache-tomcat-7.0.37.zip
  ✓ Run on Port number 8080
  ✓ Contains OSLC4JBugzilla.war

✓ Start Tomcat at [Tomcat-folder]\bin\startup.bat, startup.sh,...
  ✓ Please select the option that applies to your Operating System

✓ Register at https://landfill.bugzilla.org/bugzilla-4.2-branch/createaccount.cgi
  ✓ Username/password

✓ Start your Eclipse IDE

✓ Import the OSLC4JClientBugzilla.zip file into Eclipse
  ✓ File > Import > Existing projects into Workspace (under general)
  ✓ Option Select archive file
  ✓ Browse to file
  ✓ Finish

Note: OSLC4JClientBugzilla includes the stable release Eclipse Lyo 2.1.0
Test environment setup

- **Tomcat**

```xml
<rdf:Description rdf:about="http://10.80.223.28:8081/OSLC4JBugzilla/services/rootservices"
xmlns:oslc_cm="http://open-services.net/xmlns/cm/1.0/"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:jfs="http://jazz.net/xmlns/prod/jazz/jfs/1.0/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <dcterms:title>OSLC CM Adapter/Bugzilla Jazz Root Services</dcterms:title>
  <oslc_cm:cmServiceProviders rdf:resource="http://10.80.223.28:8081/OSLC4JBugzilla/services/catalog/singleton" />
  <jfs:authRealmName>Bugzilla</jfs:authRealmName>
  <jfs:authRequestConsumerKeyUrl rdf:resource="http://10.80.223.28:8081/OSLC4JBugzilla/services/oauth/requestKey" />
  <jfs:authApprovalModuleUrl rdf:resource="http://10.80.223.28:8081/OSLC4JBugzilla/services/oauth/approveKey" />
</rdf:Description>
```

11/6/14
Test environment setup

- Eclipse Package Explorer view
  - Labs
  - Swing client
  - Resource classes
Test Bugzilla Java Swing consumer Application

- Run Consumer GUI class
  - Expand Package view
  - Open ConsumerGUI.java
  - Set your username/password

Output:

Bugzilla Product tree

Selected Bug

Bug “Bugzilla tutorial” properties

Bugzilla Product

Raw bug “Bugzilla tutorial” RDF representation
OSLC Capabilities

- Discovery
- Standard Resource representation
- HTTP C.R.U.D for resources
- Query capabilities
- Creation factory
- **Not covered**
  - UI preview for resource Links
  - Delegated UIs for create and Select
OSLC Architecture

- Service provider catalog
  - Service providers
    - Services
    - Query capabilities
    - Creation Factory
    - Dialogs
An artefact within a tool is connected to another artefact based on the link data approach.
OSLC adapter concepts

- A tool may have both a service provider adapter and a consumer adapter
- But it is not mandatory
OSLC adapter based on the SOA approach

- /Rootservice document as entry point
  - We don’t want the consumer to guess the URIs
- Service providers
  - Advertise themselves to the catalog
- Consumer applications
  - Discover service providers
  - Discover services available
- Consume applications
  - invoke services (CRUD)
What examples do we see for OSLC interoperability?

- **Native integration examples**
  - Jazz platform,
  - HP Quality centre,
  - Microsoft SharePoint, ...
  - Salesforce

- **Plug-in/Wrapper examples**
  - Open Modelica

- **Gateway examples**
  - Bugzilla approach

More examples at [http://openservices.net/software/](http://openservices.net/software/)
Our journey

- Java class Bugzilla
  Change request
- Discover
  capability
- Retrieve
  Service providers
- Retrieve 5 first
  bugs
- Automated bug
  creation

- Java SWING
  Application
  (consume/post
  RDF document)
- Update a bug
LAB 1: Java class bugzilla Change request.
Java classes ChangeRequest and BugzillaChangeRequest

- Look at the OSLC CM specification
- Open package
  org.eclipse.lyo.oslc4j.bugzilla.resources
  - Open ChangeRequest.java
- Open BugzillaChangeRequest.java
  - Extend ChangeRequest.java with
    - Product
    - Version
    - priority
    - component
Identify the class OSLCRestClient
  ○ Open Lyo wink project
    • Src/org/eclipse/lyo/oslc4j/OsclRestClient.java
  ○ OslcRestClient, key java class of the labs

```java
public <T> T[] getOsloResources(Class<T> resourceClass)
{
    try {
        final ClientResponse response = clientResource.accept(MediaType.APPLICATION_JSON_TYPE);
        final int statusCode = response.getStatusInfo().getStatusCode();
        if (statusCode == HttpServletResponse.SC_OK) {
            return response.getEntity(resourceClass);
        }
        throw new ClientWebException(null, response);
    }
    catch (final ClientWebException exception) {
        final ClientResponse response = exception.getResponse();
        if (response != null) {
            final int statusCode = response.getStatusInfo().getStatusCode();
            if ((statusCode == HttpServletResponse.SC_NO_CONTENT) ||
                (statusCode == HttpServletResponse.SC_NOT_FOUND) ||
                (statusCode == HttpServletResponse.SC_CONFLICT)) {
                return null;
            }
        }
        throw exception;
    }
}
```
LAB 1 cont’d: Discovery capability - Rootservices as entry point

- Java class Bugzilla
- Change request

- Discover capability

- Retrieve Service providers

- Retrieve 5 first bugs

- Automated bug creation

- Java SWING Application (consume/post RDF document)

- Update a bug
Lab 1: resource class overview - Rootservices document

- **Actions:**
  - Locate Rootservices URL (entry point)
    - [http://localhost:8080/OSLC4JBugzilla/rootservices](http://localhost:8080/OSLC4JBugzilla/rootservices)
  - Write a java code to parse Rootservices document to extract catalog URI
  - Use Java class JazzRootServicesHelper

- **Output:** URL of the Service Provider Catalog
  - [http://localhost:8080/OSLC4JBugzilla/services/catalog/singleton](http://localhost:8080/OSLC4JBugzilla/services/catalog/singleton)

**Key info:**
- Lyo enables discovery capability
- Consumers don’t have to guess Resources URIs
LAB 2: Retrieve service providers (Bugzilla products)

Java class Bugzilla
Change request

Discover
capability

Retrieve
Service providers

Retrieve 5 first bugs

Automated bug creation

Java SWING Application (consume/post RDF document)

Update a bug
LAB2: Discovery capability – Service provider and service list

- Retrieve list of service providers (SP)
  - SP mapped to Bugzilla product
  - Other applications may map SP to:
    - Project area, systems or subsystem

- Actions
  - Consume Service provider catalog document
    - [http://localhost:8080/OSLC4JBugzilla/services/catalog singleton](http://localhost:8080/OSLC4JBugzilla/services/catalog singleton)
  - List Services for Service provider 2.

- Outputs:
  - List of Service providers
  - List of Services for Service provider 2
LAB2: outputs

- Outputs:
  - List of Service providers
    
    Service Providers:
    - Spider Sections, http://10.80.223.38:8081/OSLC4JBugzilla/services/serviceProviders/4
    - Sam's Widget, http://10.80.223.38:8081/OSLC4JBugzilla/services/serviceProviders/19
    - FoodReplicator, http://10.80.223.38:8081/OSLC4JBugzilla/services/serviceProviders/2
    - MyOwnBadSelf, http://10.80.223.38:8081/OSLC4JBugzilla/services/serviceProviders/3
    - WorldControl, http://10.80.223.38:8081/OSLC4JBugzilla/services/serviceProviders/1

  - List of Services for Service provider 2
    
    talogUrl: http://10.80.223.38:8081/OSLC4JBugzilla/services/catalog/singleton

    Service:
LAB 3: Consume the services

Java class Bugzilla Change request
Discover capability
Retrieve Service providers
Retrieve 5 first bugs
Automated bug creation

Java SWT application (consume OSLC UIs)
Java SWING Application (consume/post RDF document)
Update a bug
LAB 3: query capability

- Consume services available
  - Query capability
    - OSLC query properties:
      - OSLC.where, oslc.paging, oslc.pagesize, oslc.prefix, ...
  - Actions:
    - Use query properties to retrieve bugs from Bugzilla
      - queryCapability.getQueryBase().toString() + "?oslc.paging=true&oslc.pageSize=3";
      - queryCapability.getQueryBase().toString() + "?oslc.where=dcterms:identifier=" + resourceId;
  - Output:
    - RDF document of bugs that match the criteria

Find out more at http://open-services.net/bin/view/Main/OSLCCoreSpecQuery

Note:
Assign "resourceId" to a bug id in the selected Bugzilla product
LAB 3: creation factory

Java class Bugzilla
Change request

Discover capability

Retrieve Service providers

Retrieve 5 first bugs

Automated bug creation

Java SWING Application
(consume/post RDF document)

Update a bug

11/6/14
LAB 3 : Creation factory

- Consume services available
  - Creation factory
    - Post raw RDF resources to the service provider URL

- Actions
  - Identify the service URL
  - Create a bug

```java
BugzillaChangeRequest bug = new BugzillaChangeRequest();
// update the bug title and description. the title can not be empty,
bug.setTitle("Bug from java client");
bug.setDescription("This bug has been ");
bug.setComponent("renamed component");
bug.setVersion("1.0");
bug.setStatus("NEW");
bug.setSeverity("Unclassified");
bug.setOperatingSystem("All");
bug.setPlatform("All");
```

- Output
  - Display the bug Id
  - See the bug on Bugzilla website

Note:
Make sure the property Component matches with a value in the product selected.

Key message:
OSLC4J handles automatically Serialisation from Java to RDF
LAB 3: Update a bug info

Java class Bugzilla
Change request

Discover capability

Retrieve Service providers

Retrieve 5 first bugs

Automated bug creation

Java SWING
Application (consume/post RDF document)

Update a bug
LAB 3: Update a bug info

- **Objective**
  - Update an existing bug

- **Actions**
  - Use OSLCRestClient class to retrieve a bug
  - Add a new comment
  - Send an Update request to the service provider

- **Output**
  - Check the bug on the bugzilla website
  - A new comment has been added on Internet

Limitations:
OSLC4JBugzilla does not provide the list of comments in the Bug structure

Key Message:
At the server side, there is a programmatic lock system to prevent resource override.
Check header Etag
Final Demo: Java Swing GUI consumer of Bugzilla bugs

- Java class Bugzilla Change request
- Discover capability
- Retrieve Service providers
- Retrieve 5 first bugs
- Automated bug creation
- Java SWING Application (consume/post RDF document)
- Update a bug
Objective
◦ List products/List bugs
◦ Post a new bug/Update a bug

Actions
◦ Configure the GUI with your credentials
  • Click Menu Item File > Options
◦ Click on the root node in navigation panel
  • Expand a product node
  • Select a bug to display RDF document
◦ Post a new bug
  • Click Menu Item Creation > New bug
  • Click Menu Item Update > Bug update

Client Swing demo based on the OSLC Eclipse Lyo SDK
Client Swing demo based on the OSLC Eclipse Lyo SDK

- **Actions**
  - Post a new bug
    - Click Menu Item Creation > New bug
Client Swing demo based on the OSLC Eclipse Lyo SDK

- **Actions**
  - Update the selected bug
    - Click Menu Item Update > Bug update
### Let’s recap

<table>
<thead>
<tr>
<th>We described:</th>
<th>We used OSLCRestClient:</th>
<th>We showed a java GUI:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- the OSLC capabilities</td>
<td>- To process the rootservice</td>
<td>- to navigate the products</td>
</tr>
<tr>
<td>- the discovery mechanism</td>
<td>- to get the list of products</td>
<td>- to display a bug</td>
</tr>
<tr>
<td>- how to extend the ChangeRequest resource</td>
<td>- To get the list of services</td>
<td>- To post a new bug</td>
</tr>
<tr>
<td></td>
<td>- To consume the services</td>
<td>- To update a bug</td>
</tr>
</tbody>
</table>

You should be able now to build your own OSLC consumer application based on the Eclipse lyo project.
Questions?

- **Contacts**
  - Jean-Luc Johnson (AGI): jean-luc.johnson@eads.com
  - Gray Bachelor (IBM): gray.bachelor@uk.ibm.com
  - Samit Mehta (IBM): samit.mehta@us.ibm.com
  - Harry Reeder (IBM)