Supplement for OSLC Workshop

Bugzilla Tracked Resource Set Reference Application

Lab Exercises

Contents

Overview

Objectives: In this lab, you'll go through steps to enable Bugzilla as Tracked Resource Set Provider. This will make Bugzilla capable of sending Change Logs.

Pre-requisites:

- *Some Java, JavaScript, Java Servlet, Java Server Pages (JSP) and HTML programming experience within an Eclipse IDE. Experience with JAX-RS Web Services and Resource Description Framework (RDF) a plus.*

- *Open Services for Lifecycle Collaboration (OSLC) and its technologies such as OSLC Resource, Resource Shape, and OSLC REST APIs. OSLC (http://open-services.net/) is a community that is working to standardize the way that software lifecycle tools can share data. As Tracked Resource Set (TRS) refers OSLC, you should have understanding of them.*

- *OAuth (http://oauth.net/core/1.0/).You should have basic understanding of OAuth because TRS client utilizes OAuth protocol to access TRS providers. Lab 6 in the Lyo OSLC Workshop provides how to add OAuth support to your adapter.*

- *(Optional) Completion of the Lyo OSLC Workshop (http://wiki.eclipse.org/Lyo/OSLCWorkshop). This document refers to the workshop a lot for concepts around OSLC and Eclipse Lyo OSLC4J. The completion of the workshop will help your understanding.*

Length: 2-3 hours depending on which optional labs you include

Introduction

This workshop will focus on the implementation of a Tracked Resource Set (TRS) provider. It is intended to provide a foundation in the skills required for developing Tracked Resource Set Provider implementations.

TRS is a protocol designed for servers to expose a set of resources in a way that allows clients to discover all additions, removals and modifications to the resource. A Client utilizes this protocol to keep the information about the tracked resources (i.e. OSLC resources) updated. As the TRS is based on OSLC, this workshop is designed as the supplement for the Lyo OSLC Workshop and depends on its OSLC provider/consumer functions. So you should have understanding of OSLC.

OSLC (http://open-services.net/) is a community that is working to standardize the way that software lifecycle tools can share data and provides specifications for that. It includes OSLC resource definition, resource shape, and REST APIs for API-based programmatic resource access, and delegated UI and OSLC preview for Web-based UI integration. This workshop refers to the concept of RDF-based OSLC resource definition.

The example is based on a typical usage. Everyone's environment is unique, and the set of integration scenarios may need to be customized. Following these exercises will provide insight into how a common pattern can be used to implement a TRS provider for an existing tool. This has the advantage of providing additional capability to an existing tool without requesting vendor changes or locating tool source code. You will look at alternatives along the way but focus on this key pattern.

The objective of this workbook is to walk you an already-working implementation, so you can read and try this code by using a REST Client and Java debugger if you need. You will have access to the both source code of the Lyo OSLC Workshop project (org.eclipse.lyo.oslc4j.bugzilla) and this TRS provider workbook project (org.eclipse.lyo.oslc4j.bugzilla.trs) so that you can see exactly what changes are made and to where. Also each lab has the list of the Java and/or other type of files where the changes are made. These will help you to understand what needs to be implemented to enable an existing OSLC provider as a TRS provider.

The labs in this workshop make extensive use of OSLC4J and Tracked Resource Set (TRS) SDK from the Eclipse Lyo project and the projects in the Lyo OSLC Workshop. OSLC4J is a Java SDK for developing OSLC integrations. For more information, see:

http://wiki.eclipse.org/Lyo/LyoOSLC4J

http://wiki.eclipse.org/Lyo/OSLCWorkshop

http://wiki.eclipse.org/Lyo/TRSToolkit

Also, the TRS specification is available at:

http://open-services.net/wiki/core/TrackedResourceSet-2.0/

Scenario

In addition to the scenarios documented in the Lyo OSLC Workshop, Nina is looking at the way to track Bugs in her local replica, she will try to implement a TRS provider for Bugzilla so that she can keep her replica updated with the Bugs. Also by integrating this TRS provider with Rational Engineering Lifecycle Manager (RELM), she can visualize the relation between Bugs in Bugzilla and Workitems in Rational Team Concert (RTC) in order to keep the traceability within different Change Management systems.

Labs:

Completing all labs might require more than the time allotted for this workshop.

Lab 1: Enabling Tracked Resource Set Provider – this lab provides an introduction to defining OSLC resources for TRS and REST services using OSLC4J.

Lab 2: **Integrating TRS provider with Lifecycle Query Engine (LQE) (optional)** – this lab provides steps needed for the integration with LQE.

Lab 3: Integrating TRS provider with Rational Engineering Lifecycle Manager (RELM) (optional) – this lab provides steps needed for the integration with RELM.

Topics not covered:

There are a number of topics that will not be covered due to time constraints of this workshop. They are important concepts and are critical in supporting some scenarios. Some of the concepts **excluded** are: In addition to what is stated in the Lyo OSLC Workshop;

1 "Delete" Change Event:

Currently, Bugs in Bugzilla can be deleted when the container Product or Component is deleted. However, once being deleted, Bugzilla deletes the associated history too so that it can't tell us the deleted Bug as well as its history anymore.

In order to support the "Delete" Change event, Tracked Resource Set Provider needs to keep the history of the deleted Bugs. For example, there are various areas of Bugzilla that an extension can hook into, which allow the extension to run code during that point in Bugzilla's execution. If you hook "object_before_delete", you can keep the all history data of Bug just before it is going to be deleted, so that Tracked Resource Set Provider can record that event anyway. However the actual implementation is beyond the scope of this workshop.

2 Concurrent access to the Resources: A clever TRS client will spawn several threads to access TRS and the referred OSLC Resources. So that TRS provider needs to be "concurrent access safe". We achieve that in this workshop by serializing the concurrent requests using mutual exclusion (Java's "synchronized" block).

3 SPARQL and RELM View development : This workshop objective is to learn how to implement TRS provider, so SPARQL and RELM View development isn't covered.

Icons

The following symbols appear in this document at places where additional guidance is available.

Icon	Purpose	Explanation
⚠	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
ℹ	Information	This symbol indicates information that might not be necessary to complete a step, but is helpful or good to know.
🚧	Trouble-shooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

Lab 0 Getting additional setup

Objectives:

Perform the necessary steps to get the environment set up and ready for development and testing.

0.1 Set up the lab environment

Please see the Eclipse Lyo project Wiki for setup instructions:

http://wiki.eclipse.org/Lyo/BuildTRS4JBugzilla

Lab 1 Enabling Tracked Resource Set Provider

Objectives:

- Understand the concept of Tracked Resource Set (TRS) provider
- Explore OSLC resources for Tracked Resource Set (TRS) in Eclipse Lyo TRS SDK
- Explore the working implementation which uses JAX-RS Java API to provide OSLC REST services

Description:

TRS is a protocol designed for servers to expose a set of resources in a way that allows clients to discover all additions, removals and modifications to the resources. A TRS client utilizes this protocol to keep the information about the tracked resources (i.e. OSLC resources) updated. The specification, OSLC Tracked Resource Set Specification Version 2.0, can be found at:

http://open-services.net/wiki/core/TrackedResourceSet-2.0/

OSLC (http://open-services.net/) is a community that is working to standardize the way that software lifecycle tools can share data and provides specifications for that. It includes OSLC resource definition, resource shape, and REST APIs for API-based programmatic resource access, and delegated UI and OSLC preview for Web-based UI integration. This workshop refers to the concept of RDF-based OSLC resource definition. The TRS is based on the OSLC resource concepts and built on top of it.

TRS Provider provides a set of links to all resources available at a particular point of time (base resource) and a list of change log about OSLC resource CUD (creation, update, deletion) operation (change log resource).

A **Client** utilizes a Tracked Resource Set for such as keeping its own local replica of Resources in the Tracked Resource Set. The client retrieves that which OSLC resources in the replica are required to be updated, obtains the content of the resources using REST API (HTTP GET) for the OSLC resources, and then update the replica with the new contents. The following items are the brief summary:

- TRS provides the fact that resources are created, changed or modified
- TRS only provides references (links) to OSLC resources. Content to be replicated by a client are obtained via OSLC resource REST API
    - This implies that you need to implement OSLC resource (at least HTTP GET) before start implementing TRS
- Different from history or typical notification event, a TRS resource does not include what property was changed, what was the previous value, or the new value.

1.1 Understand and Explore the Lyo TRS Toolkit

TRS consists of several OSLC resources (TRS resources) to describe information for TRS:

1 Tracked Resource Set

2 Base

3 Change Log

4 Change Event

For the Bugzilla adapter, you have implemented OSLC4J-annotated Java classes for Bugzilla OSLC Change Request resource, which is an OSLC resource, in Lab 1 in the Lyo OSLC Workshop. OSLC4J framework gives RDF, JSON and Turtle serialization of the annotated Java classes.

Lyo TRS Toolkit provides OSLC4J-annoated Java beans for TRS resources. To learn the toolkit, see:

http://wiki.eclipse.org/Lyo/TRSToolkit

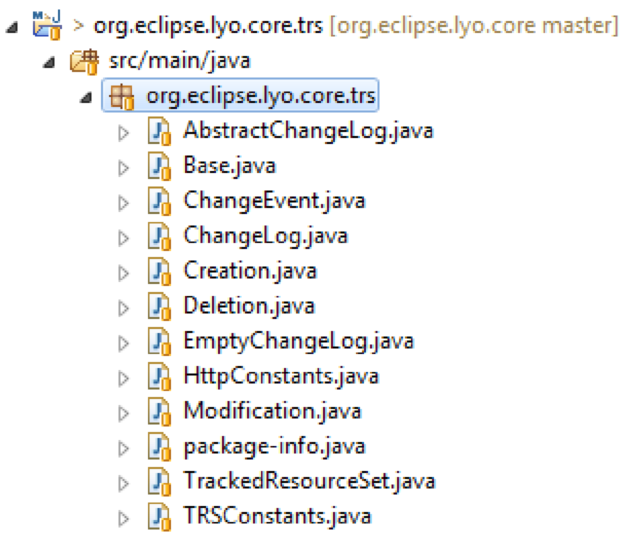
In this section, you will explore the TRS resource definitions provided by TRS toolkit so that you can get familiar with using them.

1.1.1 Changed Resources

This section doesn't involve any code changes. Just explore the Lyo TRS toolkit.

1.1.2 Steps

__1. Find **org.eclipse.lyo.core.trs** package in org.eclipse.lyo.core.trs project. The package contains Java beans for TRS resources. Notice that each OSLC resource in the TRS specification has a corresponding Java bean in the package. Each bean class has Javadoc document in it and you can use it for the detail.



```
> org.eclipse.lyo.core.trs [org.eclipse.lyo.core master]
  src/main/java
    org.eclipse.lyo.core.trs
      AbstractChangeLog.java
      Base.java
      ChangeEvent.java
      ChangeLog.java
      Creation.java
      Deletion.java
      EmptyChangeLog.java
      HttpConstants.java
      Modification.java
      package-info.java
      TrackedResourceSet.java
      TRSConstants.java
```

__2. Find **TrackedResouceSet.java** class in the org.eclipse.lyo.core.trs package and browse it and the javadoc to get familiar with the contents. This class represents Track Resource Set Resource. You can find there are properties (pairs of getter and setter methods) which are correspondent to Tracked Resource Set resource defined in the TRS specification. The significant properties and the annotations are:

```
/**
 * @return the base
 */
@OslcName(TRS_TERM_BASE)
@OslcDescription("An enumeration of the Resources in the Resource Set.")
@OslcPropertyDefinition(TRS_BASE)
@OslcTitle("Base")
public URI getBase() {
    return base;
}

/**
 * @return the changeLog
 */
@OslcName(TRS_TERM_CHANGE_LOG)
@OslcDescription("A Change Log providing a time series of incremental adjustments to the
Resource Set.")
@OslcPropertyDefinition(TRS_CHANGE_LOG)
@OslcTitle("Change Log")
public AbstractChangeLog getChangeLog() {
    return changeLog;
}
```

__3. Find **Base.java** class in org.eclipse.lyo.core.trs package and browse it and the javadoc to get familiar with the contents. This class represents Base Resource. There are properties: `members`, `cutoffEvent`, and `nextPage`.

__4. Find **ChangeLog.java** class in org.eclipse.lyo.oslc4j.core.trs package and browse it and the javadoc to get familiar with the contents. There are `change` and `previous` properties.

1.2 Create a primary TRS Service

You will start with implementing an outline of required elements for TRS. You will create a new JAX-RS service class.

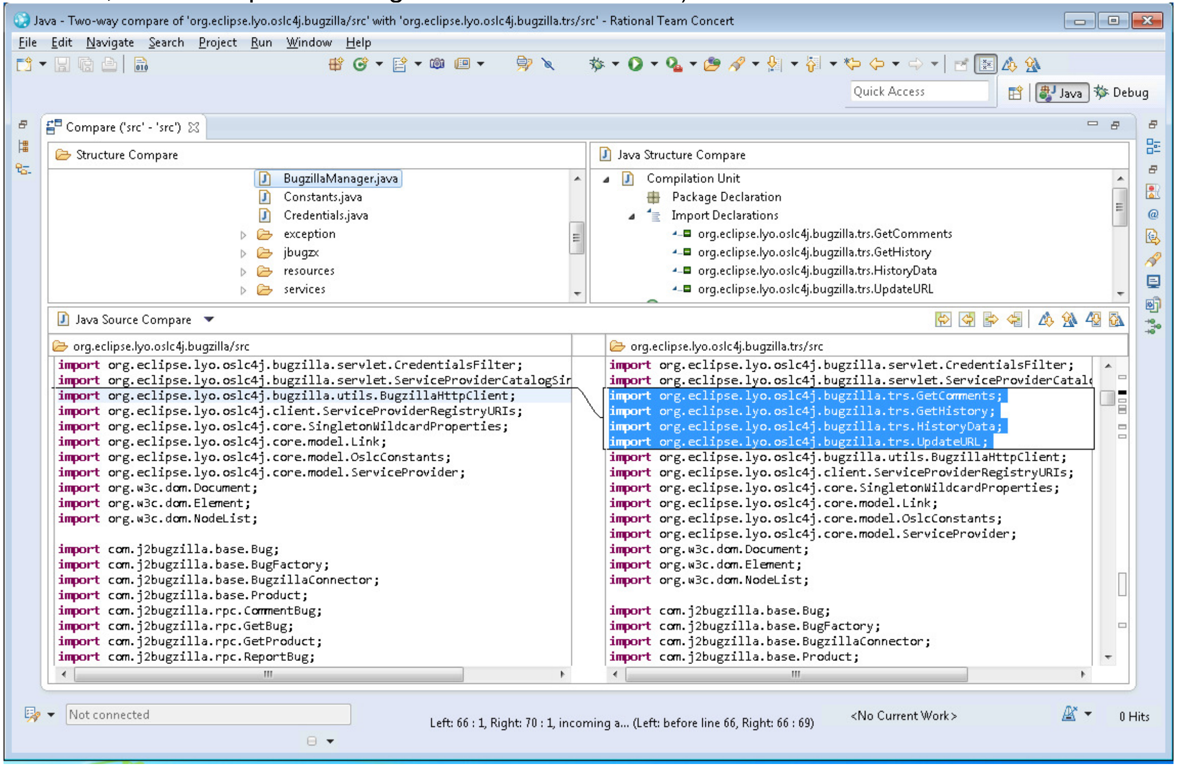
1.2.1 Changed Resources

Changed resources can be found by comparing the original org.eclipse.lyo.oslc4j.bugzilla project and the org.eclipse.lyo.oslc4j.bugzilla.trs project using Eclipse.

__a. Select the “src” folder in both projects in Project Explorer view

__b. Select “Compare With” > “Each Other” from the context menu

__c. In “Structure Compare” pane, files that have changes are listed and you can show file difference by double-clicking on a file name. You may ignore whitespace by clicking on “Ignore White Space Where Applicable” button. (In the given VM image, this button is already enabled, so white spaces are ignored as a difference.)


Java - Two-way compare of 'org.eclipse.lyo.oslc4j.bugzilla/src' with 'org.eclipse.lyo.oslc4j.bugzilla.trs/src' - Rational Team Concert
File Edit Navigate Search Project Run Window Help
Quick Access
Java
Debug
Compare ('src' - 'src')
Structure Compare
Java Structure Compare
Java Source Compare
Not connected
<No Current Work>
0 Hits

Structure Compare:

```
BugzillaManager.java
Constants.java
Credentials.java
exception
jbugzx
resources
services
```

Java Structure Compare:

```
Compilation Unit
    Package Declaration
    Import Declarations
        org.eclipse.lyo.oslc4j.bugzilla.trs.GetComments
        org.eclipse.lyo.oslc4j.bugzilla.trs.GetHistory
        org.eclipse.lyo.oslc4j.bugzilla.trs.HistoryData
        org.eclipse.lyo.oslc4j.bugzilla.trs.UpdateURL
```

org.eclipse.lyo.oslc4j.bugzilla/src

```
import org.eclipse.lyo.oslc4j.bugzilla.servlet.CredentialsFilter;
import org.eclipse.lyo.oslc4j.bugzilla.servlet.ServiceProviderCatalogSir
import org.eclipse.lyo.oslc4j.bugzilla.utils.BugzillaHttpClient;
import org.eclipse.lyo.oslc4j.client.ServiceProviderRegistryURIs;
import org.eclipse.lyo.oslc4j.core.SingletonWildcardProperties;
import org.eclipse.lyo.oslc4j.core.model.Link;
import org.eclipse.lyo.oslc4j.core.model.OslcConstants;
import org.eclipse.lyo.oslc4j.core.model.ServiceProvider;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.j2bugzilla.base.Bug;
import com.j2bugzilla.base.BugFactory;
import com.j2bugzilla.base.BugzillaConnector;
import com.j2bugzilla.base.Product;
import com.j2bugzilla.rpc.CommentBug;
import com.j2bugzilla.rpc.GetBug;
import com.j2bugzilla.rpc.GetProduct;
import com.j2bugzilla.rpc.ReportBug;
```

org.eclipse.lyo.oslc4j.bugzilla.trs/src

```
import org.eclipse.lyo.oslc4j.bugzilla.servlet.CredentialsFilter;
import org.eclipse.lyo.oslc4j.bugzilla.servlet.ServiceProviderCatal
import org.eclipse.lyo.oslc4j.bugzilla.trs.GetComments;
import org.eclipse.lyo.oslc4j.bugzilla.trs.GetHistory;
import org.eclipse.lyo.oslc4j.bugzilla.trs.HistoryData;
import org.eclipse.lyo.oslc4j.bugzilla.trs.UpdateURL;
import org.eclipse.lyo.oslc4j.bugzilla.utils.BugzillaHttpClient;
import org.eclipse.lyo.oslc4j.client.ServiceProviderRegistryURIs;
import org.eclipse.lyo.oslc4j.core.SingletonWildcardProperties;
import org.eclipse.lyo.oslc4j.core.model.Link;
import org.eclipse.lyo.oslc4j.core.model.OslcConstants;
import org.eclipse.lyo.oslc4j.core.model.ServiceProvider;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.j2bugzilla.base.Bug;
import com.j2bugzilla.base.BugFactory;
import com.j2bugzilla.base.BugzillaConnector;
import com.j2bugzilla.base.Product;
```

Left: 66 : 1, Right: 70 : 1, incoming a... (Left: before line 66, Right: 66 : 69)

__d. You can use the following buttons to navigate differences and changes.

The comparison contains all the changes to enable TRS in OSLC4JBugzilla. The resources changed particularly for this section are the following:

- ✓ **org.eclipse.lyo.oslc4j.bugzilla.services.TrackedResourceSetService.java**: New (only getTrackedResourceSetText() method)

- ✓ **org.eclipse.lyo.oslc4j.bugzilla.services.BugzillaApplication.java**: Modified as explained below. (added `//TRS resource` and `RESOURCE_CLASSES.add(TrackedResourceSetService.class);` lines).

1.2.2 Steps

__1. You will take a look at a TRS service code which is implemented as a JAX-RS service class. The class is an entry point to TRS REST service. In this step, you will see the test code, but you will see the actual code in the next section. The child steps show the class and required configuration to register the class to JAX-RS.

__a. Open **org.eclipse.lyo.oslc4j.bugzilla.trs** project, open **org.eclipse.lyo.oslc4j.bugzilla.services** package and open **TrackedResourceSetService.java**. The class is the JAX-RS service class created for TRS in the org.eclipse.lyo.oslc4j.bugzilla.trs. You'll see the added **@Path("/trs")** annotation to the class. The value "/trs" is the TRS URI.

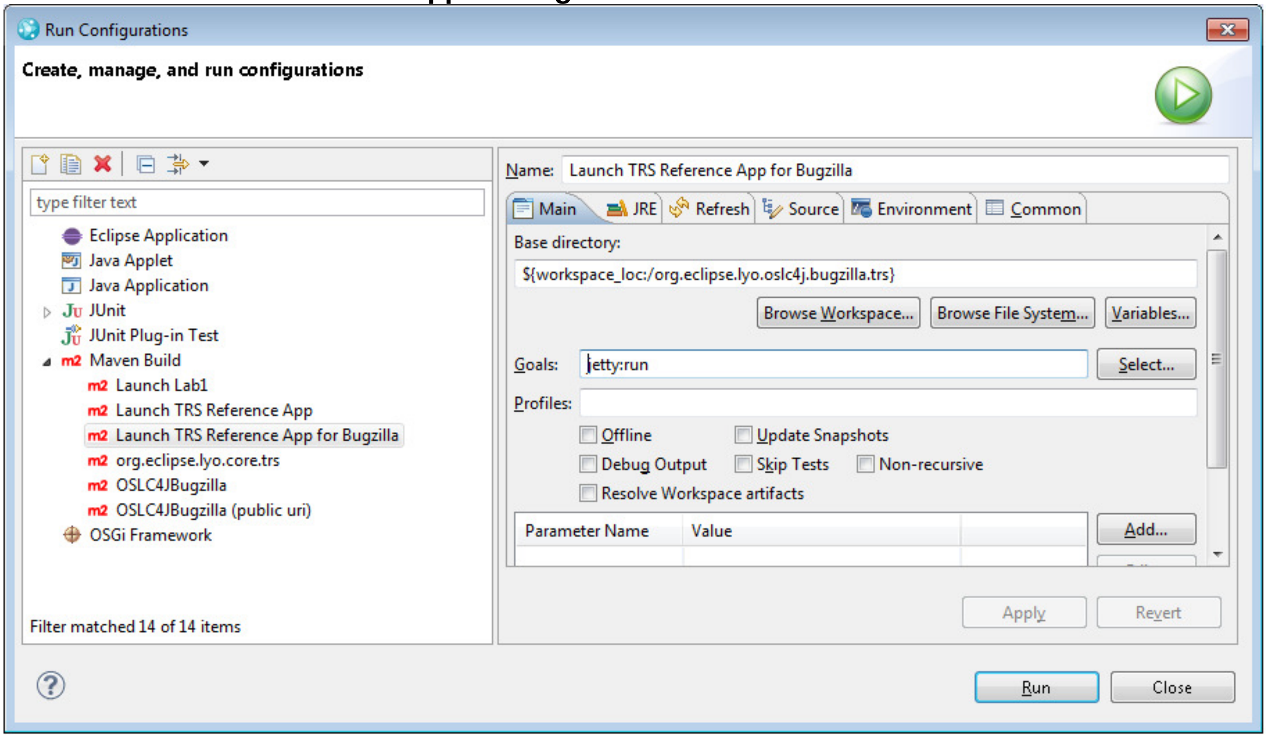
__b. Also, you'll see a service method that responds to REST requests for the TRS resources. Take a look at the **getTrackedResourceSetText()** function. The function is mapped to the path **/trs** and invoked by HTTP GET method due to **@GET** annotation. The method has **@Produces** annotation, too. It indicates that when HTTP request is invoked with **Accept** header whose value is "text/html" or "text/plain", this method is selected.

__c. The service class needs to be registered to a JAX-RS registry. Open **BugzillaApplication.java** in the same package. The class is a JAX-RS services registry in org.eclipse.lyo.oslc4j.bugzilla.trs. There is a line for registering the TrackedResourceSetService to JAX-RS.

```
//TRS resource
RESOURCE_CLASSES.add(TrackedResourceSetService.class);
```

__2. Test the current changes.

__a. Select Run->Run Configurations menu item and then expand the Maven Build section and select **Launch TRS Reference App for Bugzilla** and click the **Run** button


Run Configurations

Create, manage, and run configurations

type filter text

- Eclipse Application
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Maven Build
  - Launch Lab1
  - Launch TRS Reference App
  - Launch TRS Reference App for Bugzilla
  - org.eclipse.lyo.core.trs
  - OSLC4JBugzilla
  - OSLC4JBugzilla (public uri)
- OSGi Framework

Filter matched 14 of 14 items

Name: Launch TRS Reference App for Bugzilla

Main | JRE | Refresh | Source | Environment | Common

Base directory:
${workspace_loc:/org.eclipse.lyo.oslc4j.bugzilla.trs}

Browse Workspace... | Browse File System... | Variables...

Goals: jetty:run

Profiles:

- [ ] Offline
- [ ] Update Snapshots
- [ ] Debug Output
- [ ] Skip Tests
- [ ] Non-recursive
- [ ] Resolve Workspace artifacts

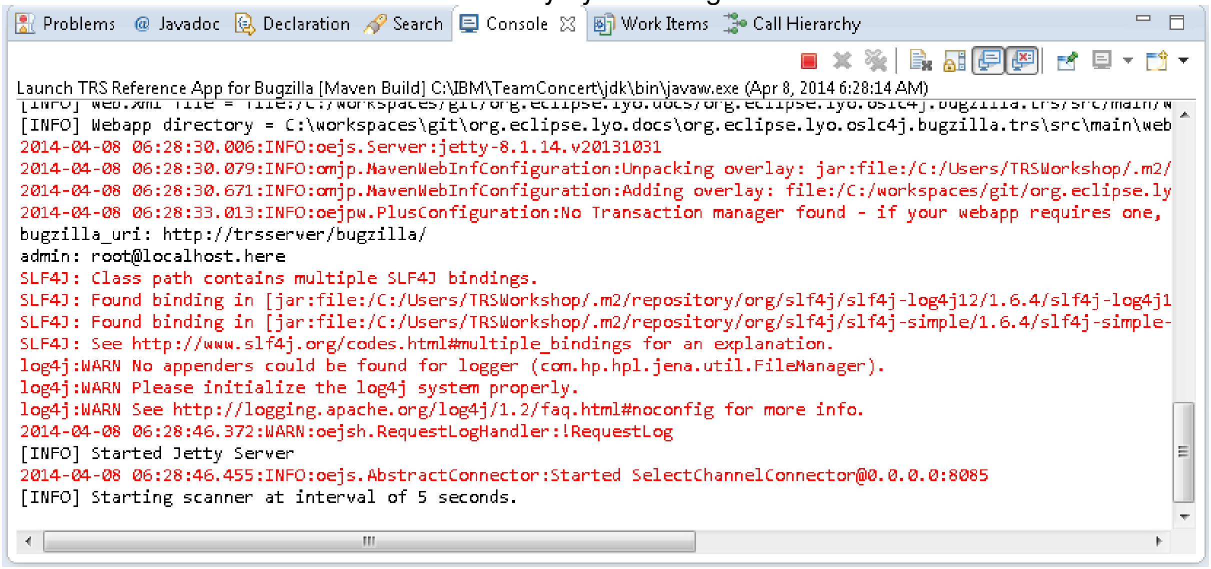
Parameter Name	Value

Add... | Apply | Revert | Run | Close

Java Debugger

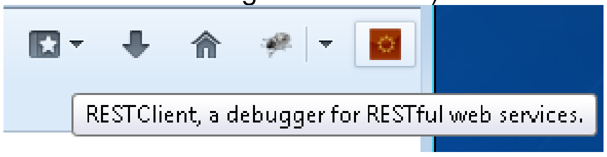
If you would like to see how TRS Provider works, you can use Java Debugger. When you will select "Run -> Debug Configurations", select "**Launch TRS Reference App for Bugzilla**" and click "**Debug**" , TRS Provider will start as a debug mode so that you can put the break point..

Observe that the server started successfully by switching to **Console** view



```
Launch TRS Reference App for Bugzilla [Maven Build] C:\IBM\TeamConcert\jdk\bin\javaw.exe (Apr 8, 2014 6:28:14 AM)
[INFO] web.xml file = file:/C:/workspaces/git/org.eclipse.lyo.docs/org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/w
[INFO] Webapp directory = C:\workspaces\git\org.eclipse.lyo.docs\org.eclipse.lyo.oslc4j.bugzilla.trs\src\main\web
2014-04-08 06:28:30.006:INFO:oejs.Server:jetty-8.1.14.v20131031
2014-04-08 06:28:30.079:INFO:omjp.MavenWebInfConfiguration:Unpacking overlay: jar:file:/C:/Users/TRSWorkshop/.m2/
2014-04-08 06:28:30.671:INFO:omjp.MavenWebInfConfiguration:Adding overlay: file:/C:/workspaces/git/org.eclipse.ly
2014-04-08 06:28:33.013:INFO:oejpw.PlusConfiguration:No Transaction manager found - if your webapp requires one,
bugzilla_uri: http://trsserver/bugzilla/
admin: root@localhost.here
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Users/TRSWorkshop/.m2/repository/org/slf4j/slf4j-log4j12/1.6.4/slf4j-log4j1
SLF4J: Found binding in [jar:file:/C:/Users/TRSWorkshop/.m2/repository/org/slf4j/slf4j-simple/1.6.4/slf4j-simple-
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2014-04-08 06:28:46.372:WARN:oejsh.RequestLogHandler:!RequestLog
[INFO] Started Jetty Server
2014-04-08 06:28:46.455:INFO:oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:8085
[INFO] Starting scanner at interval of 5 seconds.
```

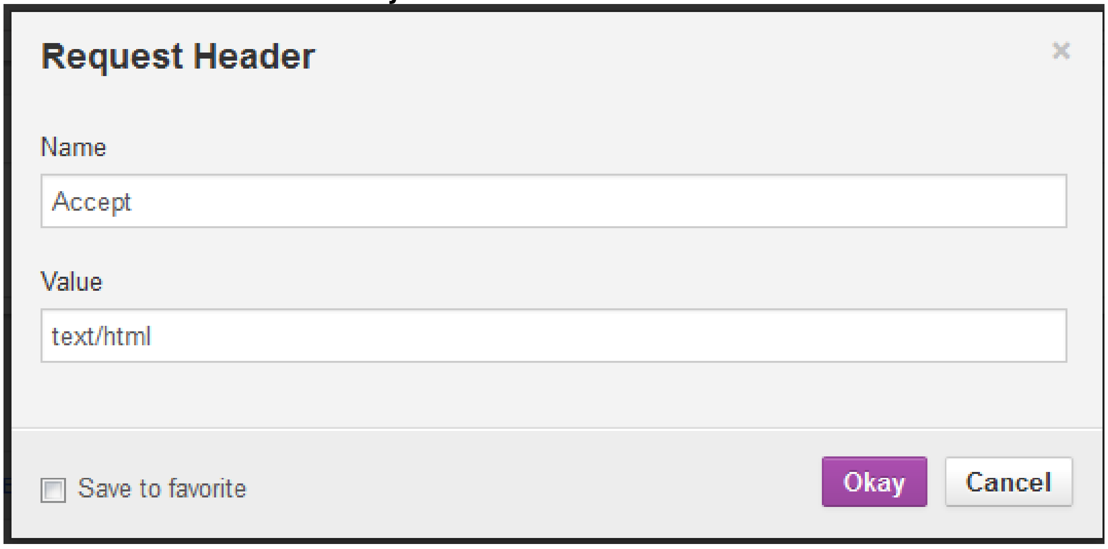
__b. Launch REST Client from Firefox by selecting Web Developer ->REST Client (You can use the icon in the navigation toolbar.)



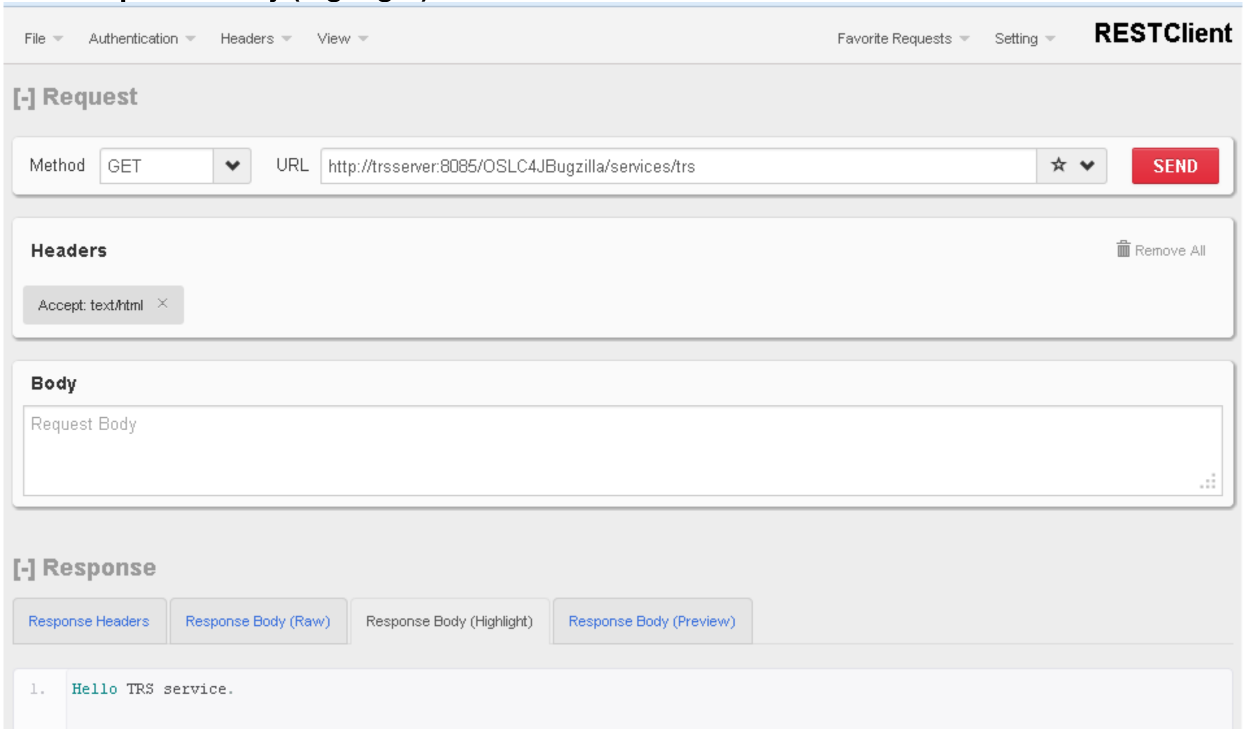
For this you'll use an add-in of Firefox called **REST Client** (https://addons.mozilla.org/en-us/firefox/addon/restclient/) to give you control over the requests being sent to your adapter

__c. Enter the **Request URL** to be http://trsserver:8085/OSLC4JBugzilla/services/trs (Note: "trsserver" is the host name where Tracked Resource Set Provider runs. This host name is also specified in org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/resources/bugz.properties.)

__d. Select **Headers** > **Custom Header** menu and add a new header with the name **Accept** and the value **text/html.** Click **Okay** button.



__e. Execute the HTTP GET method by clicking the **SEND** action. You should see a page that only contains “Hello TRS service.” in text. If you are prompted for a password, the user id is [[Bugzilla user id (**trsuser@localhost.here**)]] and the password is [[Bugzilla user password (**passw0rd**)]] . Then click **Response Body (highlight)** tab



```
Hello TRS service.
```

REST Client
The look-and-feel of the REST Client in the screenshot above is different from the one in "OSLC Enable Your Tool in a Day" Workshop. It's due to the version and you can continue to use the existing REST Client.

1.3 Building Tracked Resource Set services

In the section 1.1, you have learned that the OSLC4J TRS toolkit provides Java beans for the TRS resources and OSLC4J provides serialization of the resources. In the section 1.2, you have learned JAX-RS for the service entry point. In this section, you'll see a code of TRS service for the Bugzilla adapter which combines them.

1.3.1 Changed Resources

All the changed resources for this section are the following:

- ✓ **org.eclipse.lyo.oslc4j.bugzilla.services.TrackedResourceSetService.java**: Modified (added five methods `getTrackedResourceSet(), getBase(), getBasePage(), getChangeLog(), getChangeLogPage(),` and import statements.)
- ✓ **org.eclipse.lyo.orls4j.bugzilla.trs** package: Added.
- ✓ **org.eclipse.lyo.oslc4j.bugzilla.BugzillaManager.java**: Modified (added two methods named `getBugHistoryById()` and an import statement.)
- ✓ **org.eclipse.lyo.oslc4j.bugzilla.resources.BugzillaChangeRequest.java**: Modified (modified `fromBug()` method and import statements for "see also" and "URL" support.)
- ✓ **org.eclipse.lyo.oslc4j.bugzilla.services.BugzillaChangeRequestService.java**: Modified (adapted to `BugzillaChangeRequest#fromBug()` method signature change.)
- ✓ **org.eclipse.lyo.oslc4j.bugzilla.servlet.BugzillaServiceProviderFactory.java**: Modified. (added line `new PrefixDefinition(Constants.TRS_NAMESPACE_PREFIX, new URI(Constants.TRS_NAMESPACE))`).

1.3.2 Steps

__1. Find **TrackedResouceSetService.java** class in org.eclipse.lyo.oslc4j.bugzilla.services package. Please find the **getTrackedResourceSet()** method. The method has **@GET** annotation so when TRS client invokes the path “/trs” this method is invoked. In the class, there is a static method invoked in the middle:

```
ChangeBugzillaHistories.buildBaseResourcesAndChangeLogs(httpServletRequest).
AbstractChangeLog changeLog =
(AbstractChangeLog)ChangeBugzillaHistories.getChangeLog("1", httpServletRequest);
if (changeLog == null) {
    changeLog = new EmptyChangeLog();
}
result.setChangeLog(changeLog);
return result;
```

The invocation of the static method builds the TRS Base resource and TRS Change Log at the moment. The built resources are kept in the **CreateBugzillaHistories** class.The built resources will be referred to later. See the **getBasePage()** method:

```
public Page getBasePage(
    @PathParam("page") String pagenum ) {
    Base base;
    try {
            base = ChangeBugzillaHistories.getBaseResource(pagenum, httpServletRequest);
    } catch (URISyntaxException e) {
            throw new IllegalStateException(e);
    }
    if (base == null) {
            throw new WebApplicationException(Status.NOT_FOUND);
    }
    Page nextPage = base.getNextPage();
    if (nextPage == null) {
            throw new WebApplicationException(Status.NOT_FOUND);
    }
    // Due to OSLC4J limitation, not Base but NextPage will be returned.
    // See org.eclipse.lyo.rio.trs.resources.BaseResource.getBasePage(Long)
    return nextPage;
}
```

This method returns the TRS Base resource kept in the CreateBugzillaHistories class. Please see also **getChangeLogPage()** method:

```
public ChangeLog getChangeLogPage(
    @PathParam("page") String pagenum ) {
    try {
            return ChangeBugzillaHistories.getChangeLog(pagenum, httpServletRequest);
    } catch (URISyntaxException e) {
            throw new IllegalStateException(e);
    }
}
```

This method also returns the TRS ChangeLog resource kept in the **CreateBugzillaHistories** class.

In summary, the TRS resources are kept in the **CreateBugzillaHistories** class. They are built or refreshed by the **getTrackedResourceSet()** method, which is triggered by HTTP GET requests to "/trs". Subsequent requests are delegated to the **CreateBugzillaHistories** class and the TRS resources kept in the class are returned.

__2. Find the **getBase()** method. This method is mapped to "/trs/base" path due to the **@Path** annotation. Requests to this method are redirected to the URL that indicates the first page of the base resource. The base resource can be split into several pages for performance. The **getBasePage()** method is for the page and it returns the page from the **CreateBugzillaHistories** class See TRS specification for the paging:

http://open-services.net/wiki/core/TrackedResourceSet-2.0/#Base-Resources

__3. Find the **getChangeLog()** method. This method is mapped to "/trs/changeLog" path due to **@Path** annotation. Requests to this method are redirected to the URL that indicates the first page of the change log. The change log can also be split into several pages for performance. The **getChangeLogPage()** method is for the page and it returns the page from the **CreateBugzillaHistories** class.

__4. Find the **buildBaseResourcesAndChangeLogsInternal()** method in the **ChangeBugzillaHistories** class in org.eclipse.lyo.oslc4j.bugzilla.trs package. It's the method for building the TRS resources.

__a. Around the line 346, you see **histories** array variable and in the several lines of the code, they obtain Bug histories from Bugzilla and put them into the variable.

__b. Around the line 372, you see **changeLog** variable, it's the change log to be built.

__c. Around 380, there is a ***for*** loop for the **histories**. The TRS resources are built in the loop. You can find TRS change event creation

```
ChangeEvent ce = historyData.getType() == HistoryData.CREATED ?
  new Creation(changedUri, uri, changeOrder) :
  new Modification(changedUri, uri, changeOrder);
```

__d. You can also find the change log resource creation `changeLog = new ChangeLog();` The change log pages are stored into ***changeLogs*** static variable.

__e. You can also find the base resource creation `base = new Base();` The base pages are stored into ***baseResouces*** static variable.

The number of resources in each page is statically specified in the ChangeBugzillaHistories class.
BASE_PAGELIMIT (3 by default) is for the number of members in a Base page.
CHANGELOG_PAGELIMIT (3 by default) is for the number of resources in a ChangeLog page.

As for now, the number of Bugs TRS provider can extract from one product is **100**. You can change it by fixing the code in the following function.

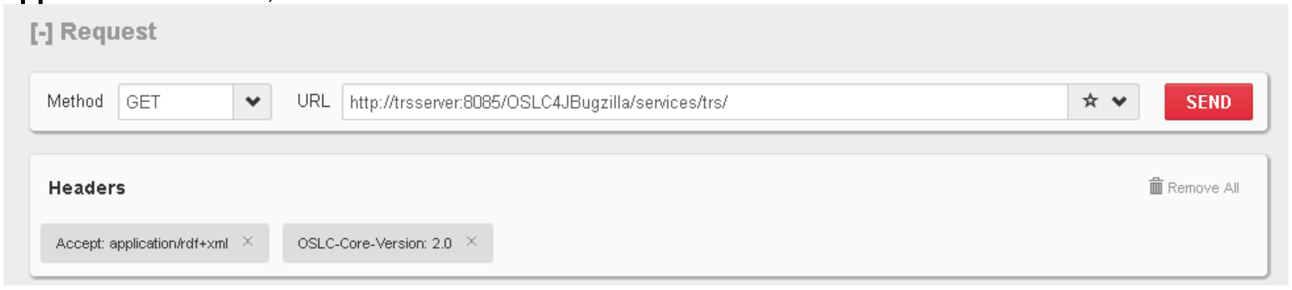
MAXNUMBEROFBUGS in the ChangeBugzillaHistories calss.

__5. Time for the final tests of TRS

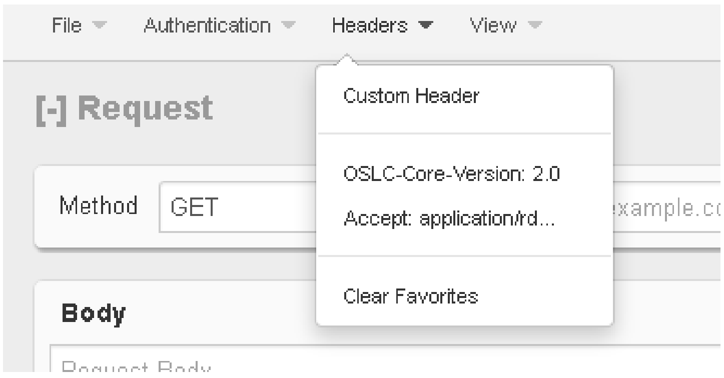
__a. Start OSLC4JBugzilla server if not already started. Run > Run Configuration > Maven Build > **Launch TRS Reference App for Bugzilla**.

__b. In Firefox, go to Web Developer ->REST Client

__c. Select Headers > Custom Header and add a header with the name **Accept** and a value of **application/rdf+xml**, and the name **OSLC-Core-Version** and a value of **2.0**.

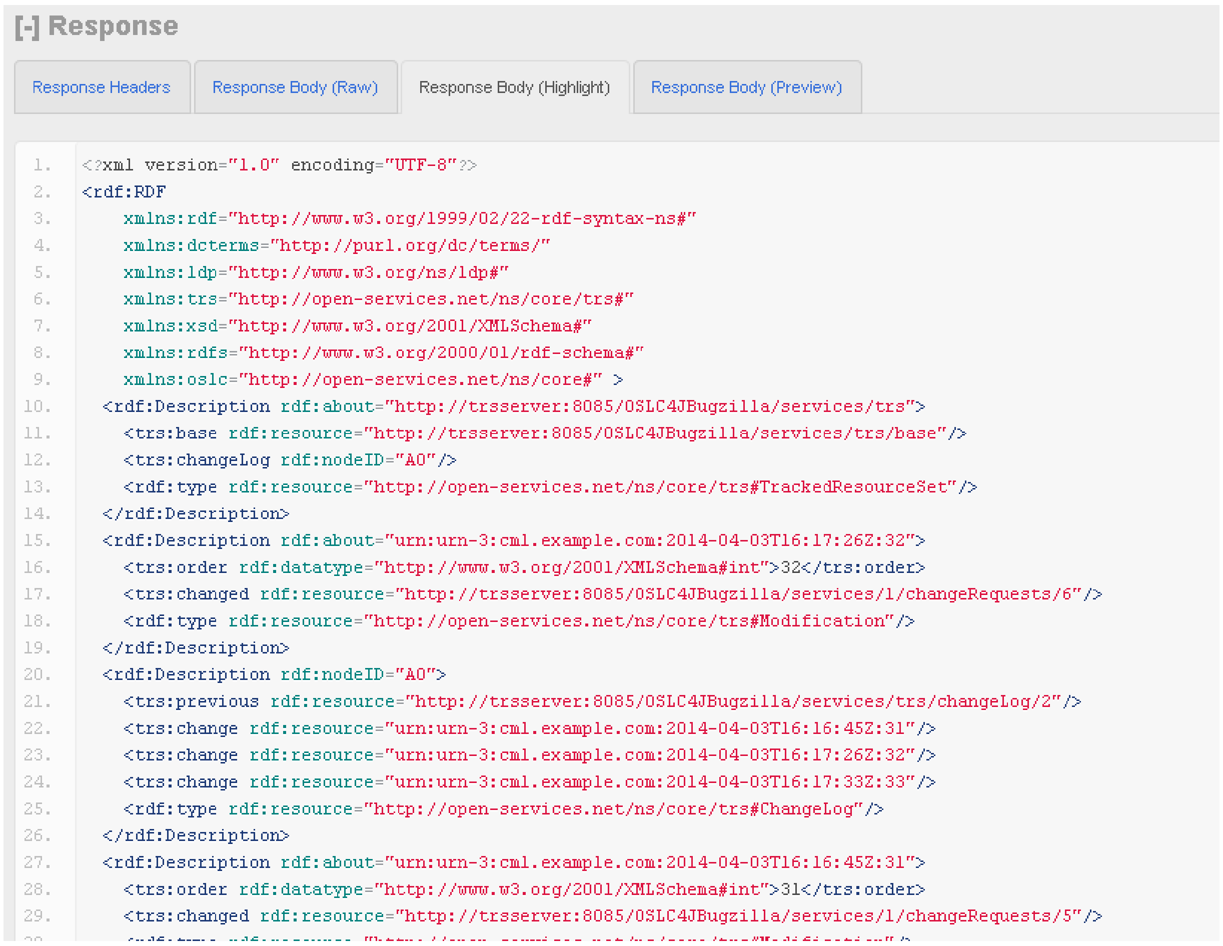

[-] Request
Method
GET
URL
http://trsserver:8085/OSLC4JBugzilla/services/trs/
SEND
Headers
Remove All
Accept: application/rdf+xml
OSLC-Core-Version: 2.0

These header values are already defined in **Headers** menu, so you can insert them from the menu.



__d. Use the url http://trsserver:8085/OSLC4JBugzilla/services/trs/ (Note: "trsserver" is the host name where Tracked Resource Set Provider runs. This host name is also specified in org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/resources/bugz.properties.)

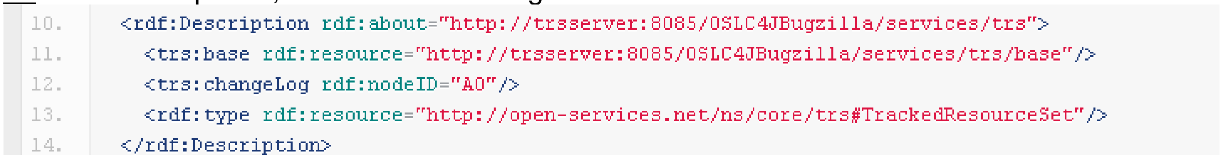
__e. You will see Change Logs as well as the url to the base resources. For example,


[-] Response

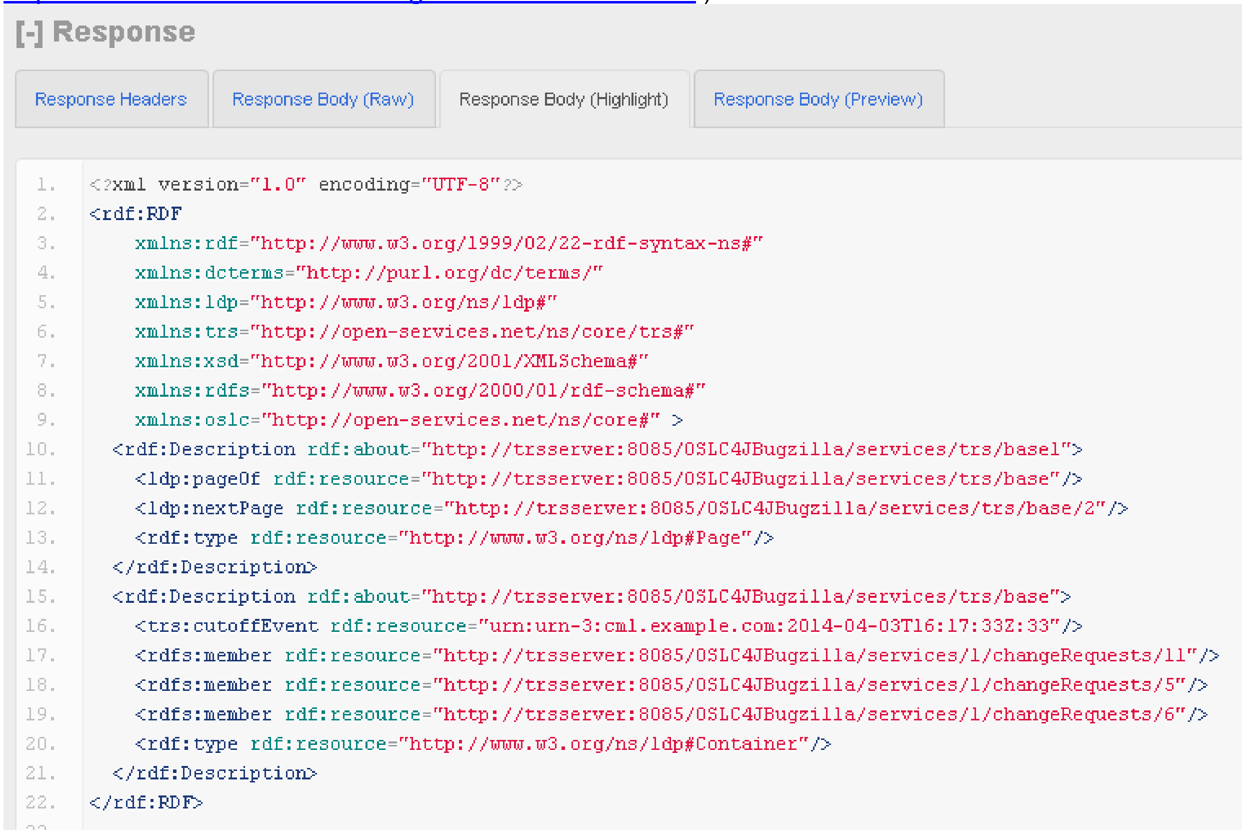
Response Headers | Response Body (Raw) | Response Body (Highlight) | Response Body (Preview)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:ldp="http://www.w3.org/ns/ldp#"
    xmlns:trs="http://open-services.net/ns/core/trs#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:oslc="http://open-services.net/ns/core#" >
  <rdf:Description rdf:about="http://trsserver:8085/OSLC4JBugzilla/services/trs">
    <trs:base rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/trs/base"/>
    <trs:changeLog rdf:nodeID="A0"/>
    <rdf:type rdf:resource="http://open-services.net/ns/core/trs#TrackedResourceSet"/>
  </rdf:Description>
  <rdf:Description rdf:about="urn:urn-3:cml.example.com:2014-04-03T16:17:26Z:32">
    <trs:order rdf:datatype="http://www.w3.org/2001/XMLSchema#int">32</trs:order>
    <trs:changed rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/6"/>
    <rdf:type rdf:resource="http://open-services.net/ns/core/trs#Modification"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <trs:previous rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/trs/changeLog/2"/>
    <trs:change rdf:resource="urn:urn-3:cml.example.com:2014-04-03T16:16:45Z:31"/>
    <trs:change rdf:resource="urn:urn-3:cml.example.com:2014-04-03T16:17:26Z:32"/>
    <trs:change rdf:resource="urn:urn-3:cml.example.com:2014-04-03T16:17:33Z:33"/>
    <rdf:type rdf:resource="http://open-services.net/ns/core/trs#ChangeLog"/>
  </rdf:Description>
  <rdf:Description rdf:about="urn:urn-3:cml.example.com:2014-04-03T16:16:45Z:31">
    <trs:order rdf:datatype="http://www.w3.org/2001/XMLSchema#int">31</trs:order>
    <trs:changed rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/5"/>
```

__f. In the response, find "<trs:base>" tag


```
  <rdf:Description rdf:about="http://trsserver:8085/OSLC4JBugzilla/services/trs">
    <trs:base rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/trs/base"/>
    <trs:changeLog rdf:nodeID="A0"/>
    <rdf:type rdf:resource="http://open-services.net/ns/core/trs#TrackedResourceSet"/>
  </rdf:Description>
```

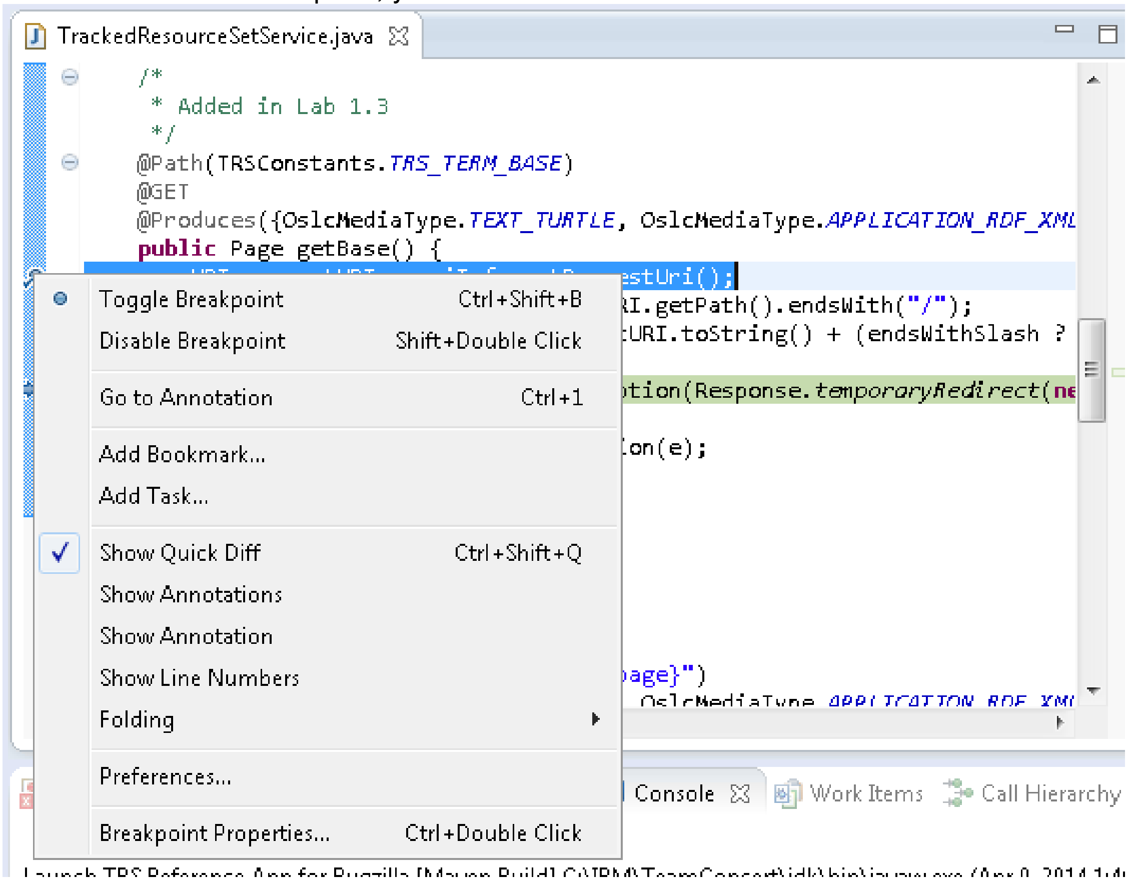
__g. Send GET command to that url (For example, http://trsserver:8085/OSLC4JBugzilla/services/trs/base)



```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:ldp="http://www.w3.org/ns/ldp#"
    xmlns:trs="http://open-services.net/ns/core/trs#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:oslc="http://open-services.net/ns/core#" >
  <rdf:Description rdf:about="http://trsserver:8085/OSLC4JBugzilla/services/trs/base1">
    <ldp:pageOf rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/trs/base"/>
    <ldp:nextPage rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/trs/base/2"/>
    <rdf:type rdf:resource="http://www.w3.org/ns/ldp#Page"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://trsserver:8085/OSLC4JBugzilla/services/trs/base">
    <trs:cutoffEvent rdf:resource="urn:urn-3:cml.example.com:2014-04-03T16:17:33Z:33"/>
    <rdfs:member rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/11"/>
    <rdfs:member rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/5"/>
    <rdfs:member rdf:resource="http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/6"/>
    <rdf:type rdf:resource="http://www.w3.org/ns/ldp#Container"/>
  </rdf:Description>
</rdf:RDF>
```

__h. In addition to REST Client, if you will use Java Debugger and put a break point at, for example, org.eclipse.lyo.oslc4j.bugzilla.services.TrackedResourceSetService.getBase(), and try to access the url for <trs:base>, for example, http://trsserver:8085/OSLC4JBugzilla/services/trs/base, by REST Client, Java Debugger tells you how this method is called, what parameters are passed, and what value is returned to.

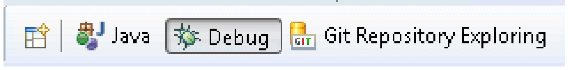
In order to set the break point, you can do it on **Java Editor**



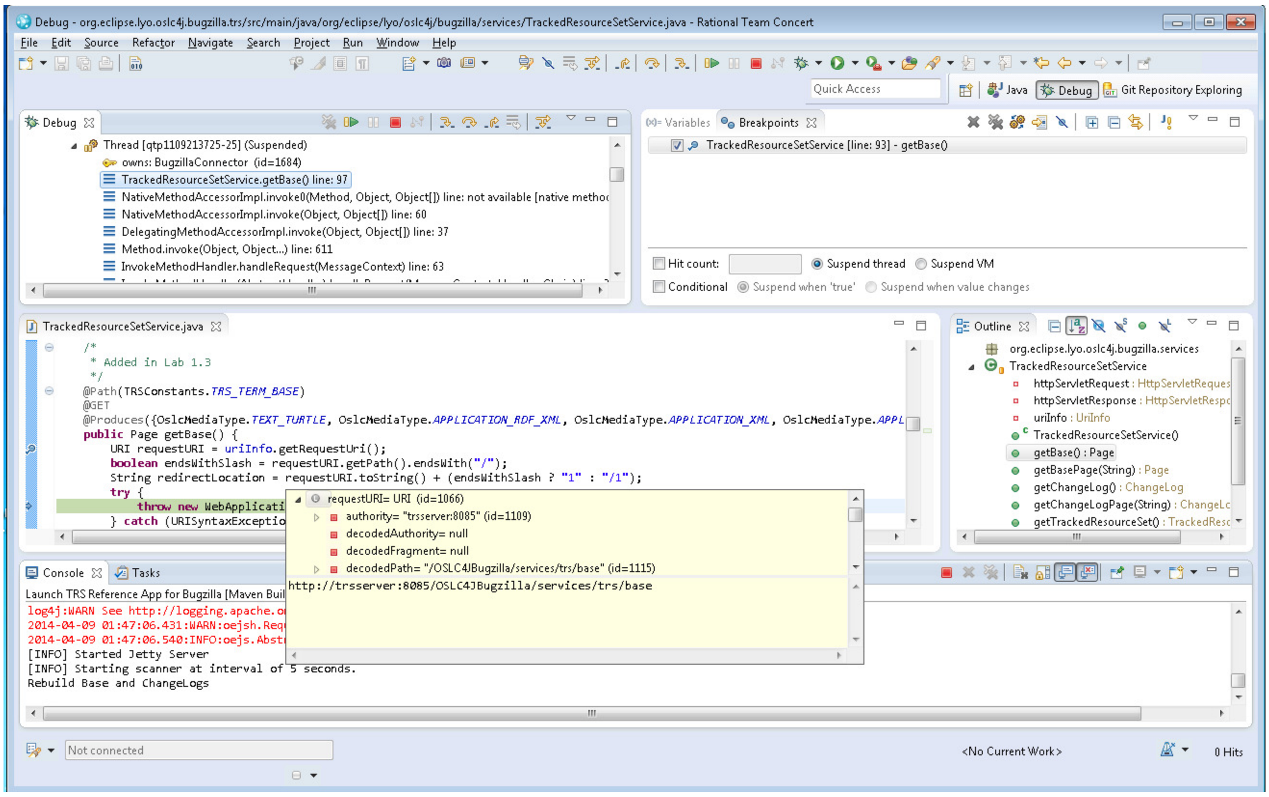
TrackedResourceSetService.java
Toggle Breakpoint Ctrl+Shift+B
Disable Breakpoint Shift+Double Click
Go to Annotation Ctrl+1
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Annotations
Show Annotation
Show Line Numbers
Folding
Preferences...
Breakpoint Properties... Ctrl+Double Click
Console
Work Items
Call Hierarchy

```
    /*
     * Added in Lab 1.3
     */
    @Path(TRSConstants.TRS_TERM_BASE)
    @GET
    @Produces({OslcMediaType.TEXT_TURTLE, OslcMediaType.APPLICATION_RDF_XML
    public Page getBase() {
                                          estUri();
                                          I.getPath().endsWith("/");
                                          URI.toString() + (endsWithSlash ?

                                          tion(Response.temporaryRedirect(ne

                                          on(e);

                                          age}")
                                           OslcMediaType.APPLICATION_RDF_XML
```

__i. Once the process reaches the break point, Eclipse workbench automatically switches the perspective to **Debug** perspective. Or you can switch it by the following button.



The **Debug** perspective looks like :.



Debug - org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/services/TrackedResourceSetService.java - Rational Team Concert
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Git Repository Exploring
Debug
Variables
Breakpoints
TrackedResourceSetService.java
Outline
Console
Tasks
<No Current Work>
0 Hits

```
Thread [qtp1109213725-25] (Suspended)
    owns: BugzillaConnector  (id=1684)
    TrackedResourceSetService.getBase() line: 97
    NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native methoc
    NativeMethodAccessorImpl.invoke(Object, Object[]) line: 60
    DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 37
    Method.invoke(Object, Object...) line: 611
    InvokeMethodHandler.handleRequest(MessageContext) line: 63
```

```
☑ TrackedResourceSetService [line: 93] - getBase()

☐ Hit count: [    ]   (●) Suspend thread   ( ) Suspend VM
☐ Conditional   (●) Suspend when 'true'   ( ) Suspend when value changes
```

```
    /*
     * Added in Lab 1.3
     */
    @Path(TRSConstants.TRS_TERM_BASE)
    @GET
    @Produces({OslcMediaType.TEXT_TURTLE, OslcMediaType.APPLICATION_RDF_XML, OslcMediaType.APPLICATION_XML, OslcMediaType.APPL
    public Page getBase() {
        URI requestURI = uriInfo.getRequestUri();
        boolean endsWithSlash = requestURI.getPath().endsWith("/");
        String redirectLocation = requestURI.toString() + (endsWithSlash ? "1" : "/1");
        try {
            throw new WebApplicati
        } catch (URISyntaxExceptio
```

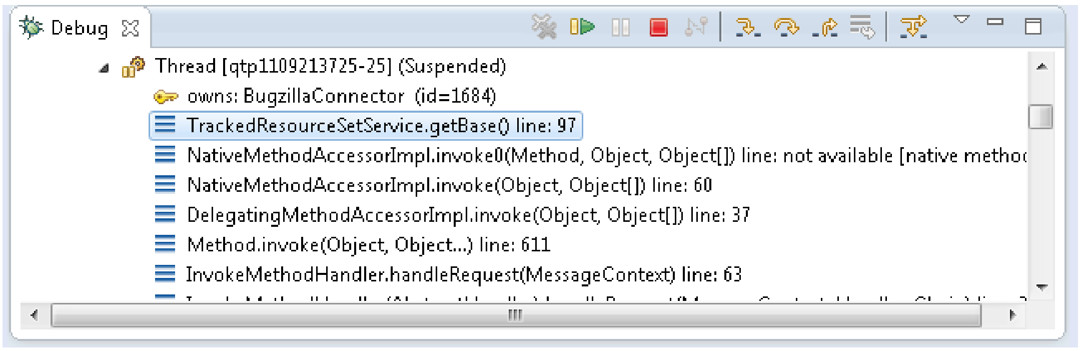
```
requestURI= URI  (id=1066)
    authority= "trsserver:8085" (id=1109)
    decodedAuthority= null
    decodedFragment= null
    decodedPath= "/OSLC4JBugzilla/services/trs/base" (id=1115)
http://trsserver:8085/OSLC4JBugzilla/services/trs/base
```

```
org.eclipse.lyo.oslc4j.bugzilla.services
TrackedResourceSetService
    httpServletRequest : HttpServletReques
    httpServletResponse : HttpServletRespo
    uriInfo : UriInfo
    TrackedResourceSetService()
    getBase() : Page
    getBasePage(String) : Page
    getChangeLog() : ChangeLog
    getChangeLogPage(String) : ChangeLo
    getTrackedResourceSet() : TrackedReso
```

```
Launch TRS Reference App for Bugzilla [Maven Buil
log4j:WARN See http://logging.apache.o
2014-04-09 01:47:06.431:WARN:oejsh.Req
2014-04-09 01:47:06.540:INFO:oejs.Abst
[INFO] Started Jetty Server
[INFO] Starting scanner at interval of 5 seconds.
Rebuild Base and ChangeLogs
```

```
Not connected
```

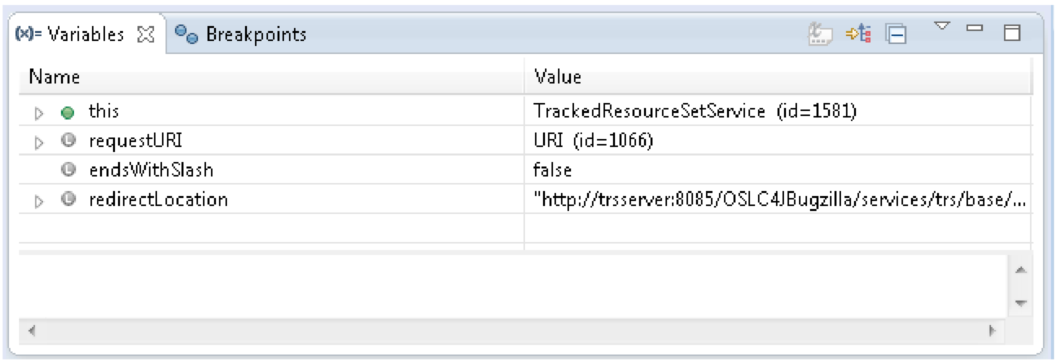
From this workbench, you can use the following **Java Debug View** to resume, step in, step over the stopped thread. Also this view shows the stack trace.



```
Thread [qtp1109213725-25] (Suspended)
    owns: BugzillaConnector  (id=1684)
    TrackedResourceSetService.getBase() line: 97
    NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native methoc
    NativeMethodAccessorImpl.invoke(Object, Object[]) line: 60
    DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 37
    Method.invoke(Object, Object...) line: 611
    InvokeMethodHandler.handleRequest(MessageContext) line: 63
```

In detail, please refer to
http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Freference%2Fviews%2Fdebug%2Fref-debug_view.htm&cp=1_4_7_2

In order to explore the variables, you can use the following **Variables View**.



Name	Value
this	TrackedResourceSetService (id=1581)
requestURI	URI (id=1066)
endsWithSlash	false
redirectLocation	"http://trsserver:8085/OSLC4JBugzilla/services/trs/base/...

In detail, please refer to
http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Freference%2Fviews%2Fvariables%2Fref-variables_view.htm&cp=1_4_7_6

As for the overall debugger usage, please refer to
http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftasks%2Ftask-running_and_debugging.htm&cp=1_3_6

1.4 Summary

This lab provided an introduction to defining OSLC resources and REST services using the Eclipse Lyo OSLC4J SDK (http://eclipse.org/lyo) for TRS. You now have the basics for exposing the change log of Bugzilla bugs as an OSLC Tracked Resource Set and you are able to retrieve them in a variety of representations which can be used by humans and other tools.

Lab 2 Integrating TRS provider with Lifecycle Query Engine (LQE) (optional)

Objectives

- Understand how TRS provider validates the authentication without an user interaction
- Understand how TRS URI can be provided by Jazz Root Services document
- Understand how TRS provider can be integrated with Lifecycle Query Engine (LQE)
- Understand how LQE uses TRS resources as a consumer of TRS provider
- This is an optional lab and can be skipped if you are constrained for time.

Description:

Now you have learned how a TRS provider works, you'd like to connect it to a TRS client. (Note that in this workshop, Lifecycle Query Engine (LQE) is used as a TRS client.) In the Lyo OSLC Workshop, you have connected the Bugzilla adapter to Rational Team Concert as OSLC-based Jazz integration. You have also done some additional steps that are needed for the integration.

The configuration to RTC and TRS provider described in the Lyo OSLC Workshop is already done in the given VM image.

This lab will explore the steps necessary to achieve those integrations in a TRS provider. It will show some additional requirements that are needed such as the Jazz Root Services document and OAuth. Also this lab will show you how to configure LQE, which is a TRS client, and TRS provider to be integrated in order for LQE to start the indexing of the resources.

Steps:

- Understand additional steps needed for integration
    - OAuth
    - Update Jazz Root services
    - OAuth configuration between TRS provider and LQE

2.1 Enhance OAuth implementation for intra-server communication

You have implemented OAuth 1.0 consumer in “Lab 6. Connecting to Rational Team Concert” in the Lyo OSLC Workshop document. It allows RTC being an OAuth consumer and the user can see a login page to OSLC4JBugzilla in RTC.

The configuration to RTC and TRS provider described in the Lyo OSLC Workshop is already done in the given VM image.

Here you will see an enhanced OAuth implementation for typical usage of TRS provider and client.

In most cases, a TRS client runs in a back-end server and users might not be involved. For example, a TRS client that keeps the latest clone of a Tracked Resource Set is certainly implemented like a daemon and there is no chance to ask the user to submit credentials via login page. Consequently, OAuth communication code needs to be extended so that a Provider (implemented in Lab 1), and a Consumer (a TRS client), can communicate without user interaction (a.k.a. two-legged OAuth). The details about this kind of communication are in:

http://oauth.googlecode.com/svn/spec/ext/consumer_request/1.0/drafts/1/spec.html

We designed the two-legged authentication in OSLC4JBugzilla as follow:

1) The OAuth consumer used for two-legged authentication must be registered as “trusted” consumer
2) A request by the “trusted” consumer is processed using a pre-defined Bugzilla user account (a.k.a. functional user). In this workshop, the user is admin user and is specified in **bugz.properties** file.

2.1.1 Changed Resources

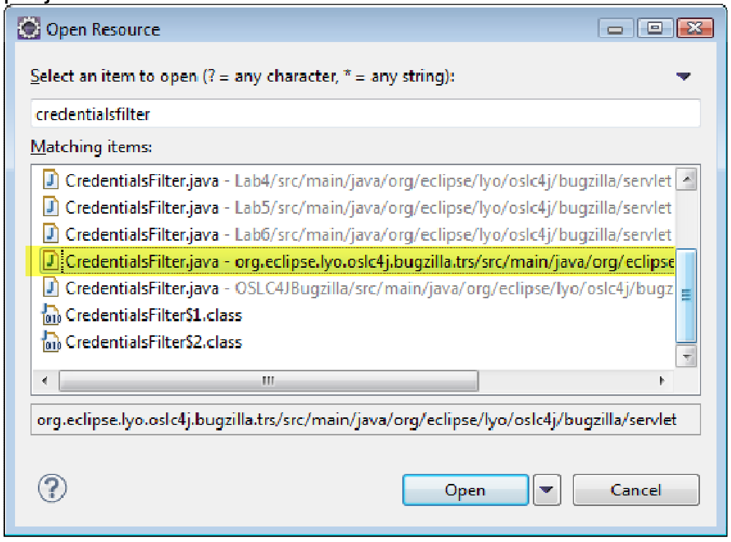
All the changed resources for this section are the following:

- ✓ **/resources/bugz.properties**: Modified. (Add properties)
- ✓ **org.eclipse.lyo.oslc4j.bugzilla.servlet.CredentialsFilter.java**: Modified.
- ✓ **org.eclipse.lyo.oslc4j.bugzilla.BugzillaManager.java**: Modified.
  - ➢ Added `adapter_host` and `admin_password` fields.
  - ➢ Added statements starting with `admin_password` and `adapter_host` in the static initializer.
  - ➢ Added `getAdminCreadentials()` and `getHost()` methods.

2.1.2 Steps

__1. You will see a two-legged OAuth beside the existing OAuth implementation. The child sections show how it is implemented in org.eclipse.lyo.oslc4j.bugzilla.trs example.

__a. Open **CredentialsFilter.java** in **org.eclipse.lyo.oslc4j.bugzilla.servlet** and then locate the **doFilter()** method. The class is the existing OAuth implementation in org.eclipse.lyo.oslc4j.bugzilla.trs. Type Ctrl+Shift+R to open “Open Resource” dialog, and in the dialog, type some characters of the file name. Then, select one in org.eclipse.lyo.oslc4j.bugzilla.trs project.



credentialsfilter

CredentialsFilter.java - Lab4/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/servlet
CredentialsFilter.java - Lab5/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/servlet
CredentialsFilter.java - Lab6/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/servlet
CredentialsFilter.java - org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/java/org/eclipse
CredentialsFilter.java - OSLC4JBugzilla/src/main/java/org/eclipse/lyo/oslc4j/bugz
CredentialsFilter$1.class
CredentialsFilter$2.class

org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/servlet

__b. The first step for handling two-legged OAuth is to validate the request. The following fragment takes an HTTP request and raises the **isTwoLeggedOAuthRequest** flag when the request is a valid two-legged OAuth request. The flag is considered in the next step to fill the functional user credentials.

```
OAuthMessage message = OAuthServlet.getMessage(request, null);
// test if this is a valid two-legged oauth request
if ("".equals(message.getToken())) {
        validateTwoLeggedOAuthMessage(message);
        isTwoLeggedOAuthRequest = true;
}
```

__c. The **validateTwoLeggedOAuthMesage()** method is to confirm the message being signed using valid consumer key and consumer secret. The fragments above and below are reusable in your component to validate the request.

```
private void validateTwoLeggedOAuthMessage(OAuthMessage message)
                throws IOException, OAuthException, URISyntaxException {
        OAuthConfiguration config = OAuthConfiguration.getInstance();
        LyoOAuthConsumer consumer = config.getConsumerStore()
                                        .getConsumer(message.getConsumerKey());
        if (consumer != null && consumer.isTrusted()) {
                // The request can be a two-legged request because it's a trusted consumer
                // Validate the message with an empty token and an empty secret
                OAuthAccessor accessor = new OAuthAccessor(consumer);
                accessor.requestToken = "";
                accessor.tokenSecret = "";
                config.getValidator().validateMessage(message, accessor);
        } else {
                throw new OAuthProblemException(
                                OAuth.Problems.TOKEN_REJECTED);
        }
}
```

__d. Once the request is validated, the next step is to grant the request as a (pre-authorized) functional user. In TRS4JBugzilla example, it's done by associating the request with an authorized BugzillaConnector instance.

__i. Open **bugz.properties** in **src/main/resources**. The property named **admin** is for specifying Bugzilla admin user name and **admin_password** property is for the password.

In the given VM image, [[Bugzilla admin id (**root@localhost.here**)]] and the password is [[Bugzilla admin password (**passw0rd**)]]. They are already set.

The values are used to create a credential for logging in to Bugzilla for two-legged OAuth when the **isTwoLeggedOAuthRequest** flag is set. Go back to the **doFilter()** method and scroll down to the following fragment. In the fragment, ensure that a BugzillaConnector is associated to a token ("") used for two-legged authentication when it's not associated when the flag is **true**, :

```
Credentials credentials;
if (isTwoLeggedOAuthRequest) {
      connector = keyToConnectorCache.get("");
      if (connector == null) {
              credentials = BugzillaManager.getAdminCredentials();
              connector = getBugzillaConnector(credentials);
              keyToConnectorCache.put("", connector);
      }
} else {
      ...
}
```

__2. You have seen how a two-legged OAuth is implemented. The changes are tested in the later section.

2.1.3 Implementing a Credentials Filter

You have seen the **CredentailsFilter** class for this workshop. The class takes credentials from requests to the adapter, and then passes it to the Bugzilla server. When you create your adapter, your "CredentailsFilter" might be similar. So, there is the base class, **AbstractAdapterCredentialsFilter**, which provides the common part. Please find the javadoc of the class.

In the workshop code, there is also the **BugzillaAdapterCredentialsFilter** class, an example implementation of "CredentialsFilter" based on the base class. You can try the **BugzillaAdapterCredentialsFilter** following the instruction in its javadoc.

> The running workshop code uses org.eclipse.lyo.oslc4j.bugzilla.servlet.CredentialsFilter as a filter. So if you would like to put the break point, please use this class.

2.2 Enhance Jazz Root Services support

You have added Jazz Root Services document in "Lab 6. Connecting to Rational Team Concert in Lyo OSLC Workshop document" in the Lyo OSLC Workshop

> Root Services document described here is already added in the given VM image.

In this section, you will see the enhanced Jazz Root Services document in order for the Jazz Team Server as well as other TRS clients, such as LQE, to be able to query the TRS services your adapter provides.

In addition to Lab 6 in the Lyo OSLC Workshop, the following resources provide details:

https://jazz.net/wiki/bin/view/Main/RootServicesSpec

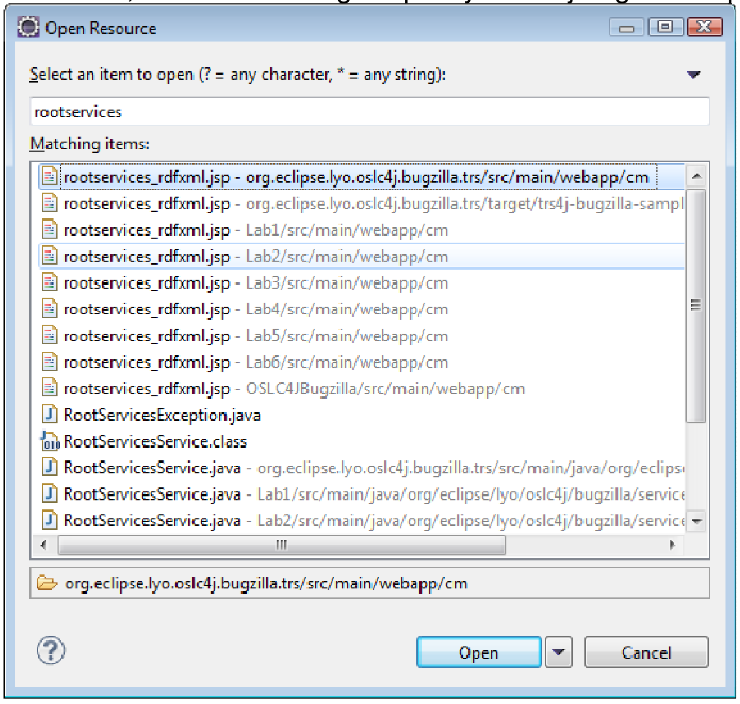
https://jazz.net/wiki/bin/view/Main/RootServicesSpecAddendum2

2.2.1 Steps

__1. Open **rootservices_rdfxml.jsp** in org.eclipse.lyo.oslc4j.bugzilla.trs project. The file is a template of the rootservices document in org.eclipse.lyo.oslc4j.bugzilla.trs.

__a. Type Ctrl+Shift+R to open "Open Resource" dialog, and type some characters of the file name in the edit.

__b. And then, select one in org.eclipse.lyo.oslc4j.bugzilla.trs project.


Open Resource

Select an item to open (? = any character, * = any string):

rootservices

Matching items:

```
rootservices_rdfxml.jsp - org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/webapp/cm
rootservices_rdfxml.jsp - org.eclipse.lyo.oslc4j.bugzilla.trs/target/trs4j-bugzilla-sampl
rootservices_rdfxml.jsp - Lab1/src/main/webapp/cm
rootservices_rdfxml.jsp - Lab2/src/main/webapp/cm
rootservices_rdfxml.jsp - Lab3/src/main/webapp/cm
rootservices_rdfxml.jsp - Lab4/src/main/webapp/cm
rootservices_rdfxml.jsp - Lab5/src/main/webapp/cm
rootservices_rdfxml.jsp - Lab6/src/main/webapp/cm
rootservices_rdfxml.jsp - OSLC4JBugzilla/src/main/webapp/cm
RootServicesException.java
RootServicesService.class
RootServicesService.java - org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/java/org/eclips
RootServicesService.java - Lab1/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/service
RootServicesService.java - Lab2/src/main/java/org/eclipse/lyo/oslc4j/bugzilla/service
```

org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/webapp/cm

__2. Locate an element **bugz:trackedResourceSetProvider**. The element is for a definition of the TRS in rootservices document and points the TRS URI. In org.eclipse.lyo.oslc4j.bugzilla.trs, the URI is "`baseUri + "/trs"`". In RDF point of view, the rootservices RDF resource has a new predicate, `bugz:trackedResourceSetProvider`, and the object is an anonymous resource whose type is `trs:TrackedResourceSetProvider`. The anonymous resource further has a predicate `trs:trackedResourceSet` and the object is the TRS URI.

```
<!-- Bugzilla Tracked Resource Set Provider -->
<bugz:trackedResourceSetProvider>
        <trs:TrackedResourceSetProvider>
                <trs:trackedResourceSet rdf:resource="<%= baseUri + "/trs" %>" />
        </trs:TrackedResourceSetProvider>
</bugz:trackedResourceSetProvider>
```

__a. For the **bugz** and **trs** prefixes in the XML, you'll have added **xmlns:bugz** and **xmlns:trs** attributes to the root **rdf:Description** element.

```
<rdf:Description rdf:about="<%= baseUri + "/rootservices" %>"
        xmlns:bugz="http://www.bugzilla.org/rdf#"
        xmlns:trs="http://open-services.net/ns/core/trs#"
        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#">
```

2.3 Configure LQE to be connected to TRS provider

You have seen TRS service URI to be added to Jazz Rootservices document. So the next step is to configure OAuth key between TRS provider and LQE so that LQE can access TRS resources provided by TRS provider.

2.3.1 Steps

__1. Restart TRS provider to read **bugz.properties** in **src/main/resources** again in order to let TRS provider know the correct Bugzilla admin user id and its password if they were updated.

__2. In Browser, open LQE JTS Admin page (for example, https://trsserver:9443/jts/admin . Note: “trsserver” is the host name where LQE runs.) and login it with the admin user id and its password

In the given VM image, Jazz applications’(JTS, LQE and RELM) admin user id is “**rational**” and its password is “**rational**”.

__a. Click **Manage Server** in Manage Server pane, and then navigate to **Friends (Outbound)** in the left pane. From this page, you are going to ask TRS provider to generate OAuth Consumer Key and its OAuth Consumer Secret for LQE to access resources TRS provider provides. Once this OAuth key is configured, LQE can start indexing of such resources without asking the user the access credentials.

__b. Click **Add...** link at the right of the **Friends List** section title

__c. Fill the fields

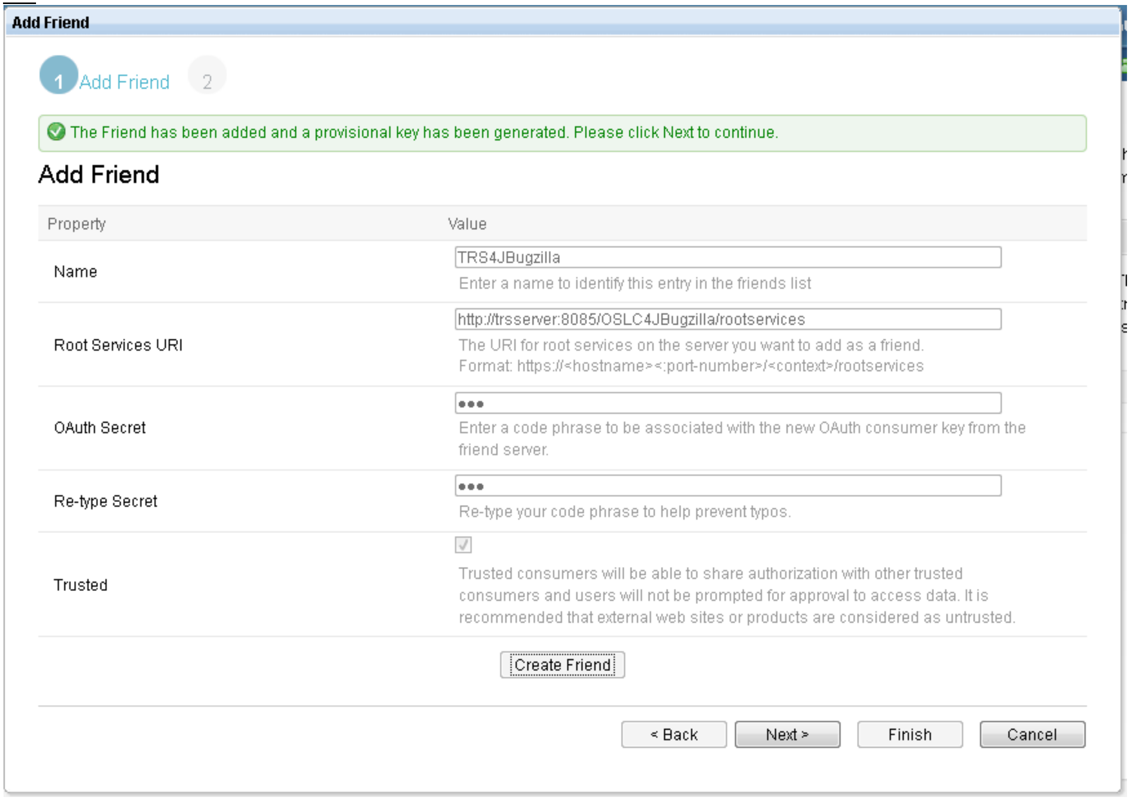
__i. Name: **TRS4JBugzilla**

__ii. Root Services URI: **http://trsserver:8085/OSLC4JBugzilla/rootservices** (Note: “trsserver” is the host name where Tracked Resource Set Provider runs. This host name is also specified in org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/resources/bugz.properties.)

__iii. OAuth Secret: **xyz**

__iv. Trusted: **Checked**

__d. Click **Create Friend** button



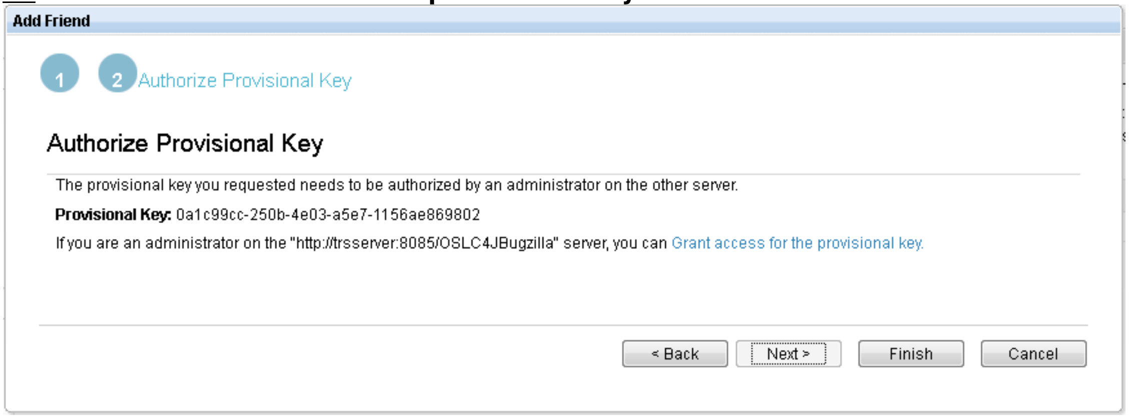
The Friend has been added and a provisional key has been generated. Please click Next to continue.

Add Friend

Property	Value
Name	TRS4JBugzilla Enter a name to identify this entry in the friends list
Root Services URI	http://trsserver:8085/OSLC4JBugzilla/rootservices The URI for root services on the server you want to add as a friend. Format: https://<hostname>:<port-number>/<context>/rootservices
OAuth Secret	●●● Enter a code phrase to be associated with the new OAuth consumer key from the friend server.
Re-type Secret	●●● Re-type your code phrase to help prevent typos.
Trusted	✓ Trusted consumers will be able to share authorization with other trusted consumers and users will not be prompted for approval to access data. It is recommended that external web sites or products are considered as untrusted.

__e. Click **Next**

__f. Click **Grant access for the provisional key** link



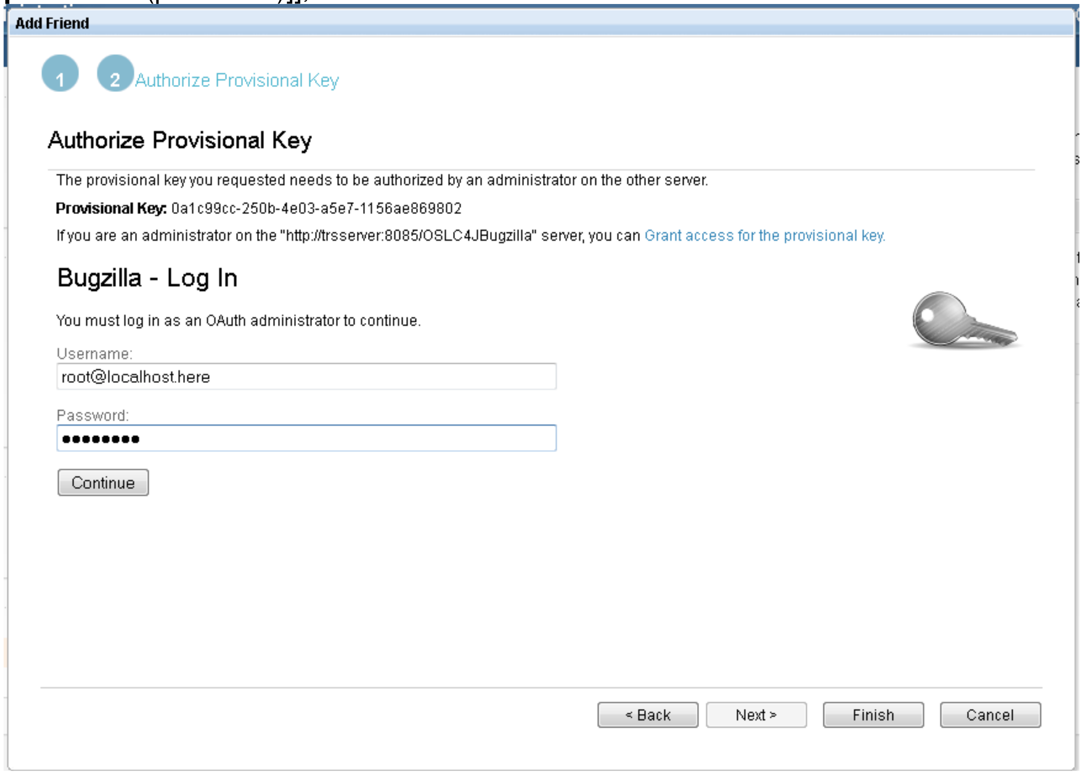
Authorize Provisional Key

The provisional key you requested needs to be authorized by an administrator on the other server.

Provisional Key: 0a1c99cc-250b-4e03-a5e7-1156ae869802

If you are an administrator on the "http://trsserver:8085/OSLC4JBugzilla" server, you can Grant access for the provisional key.

__g. Fill the form using [[**Bugzilla admin id** (default root@localhost.here)]] and [[**Bugzilla admin password** (passw0rd)]], and then click **Continue**



Add Friend

1 2 Authorize Provisional Key

Authorize Provisional Key

The provisional key you requested needs to be authorized by an administrator on the other server.

Provisional Key: 0a1c99cc-250b-4e03-a5e7-1156ae869802

If you are an administrator on the "http://trsserver:8085/OSLC4JBugzilla" server, you can Grant access for the provisional key.

Bugzilla - Log In

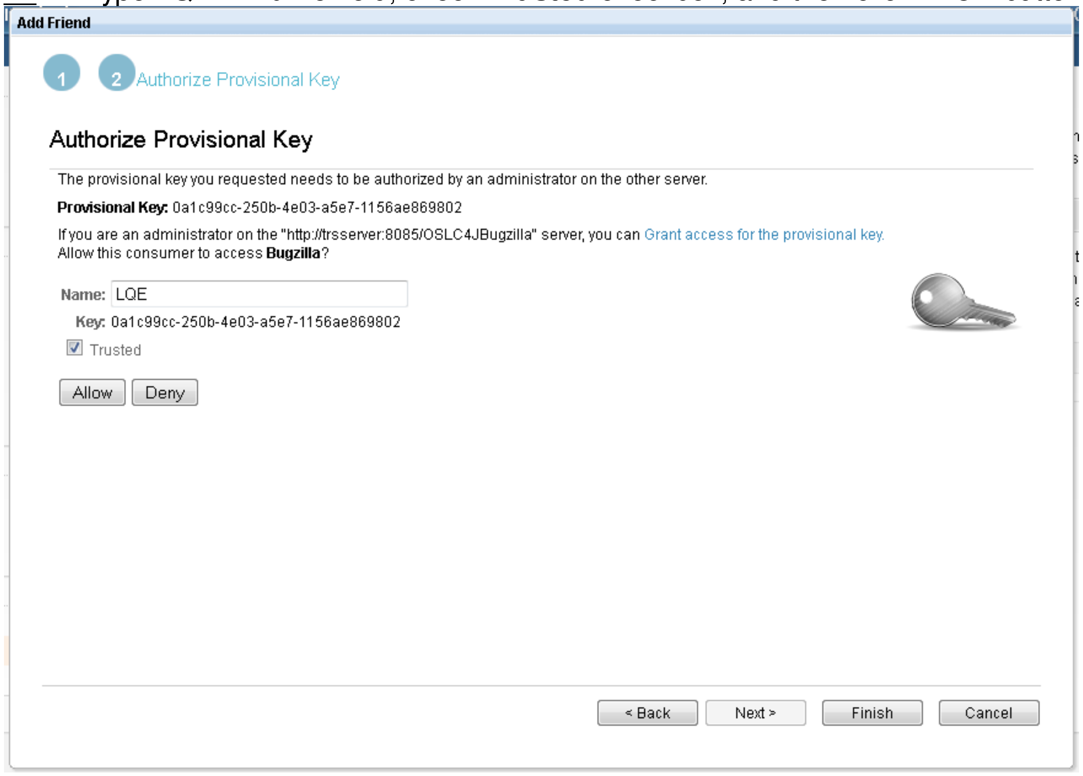
You must log in as an OAuth administrator to continue.

Username: root@localhost.here

Password: ●●●●●●●●●

NOTE: You may not see the login message if you have already logged in to the TRS4JBugzilla adapter as the administrator.

__h. Type *LQE* in **Name** field, check **Trusted** checkbox, and then click **Allow** button



Authorize Provisional Key

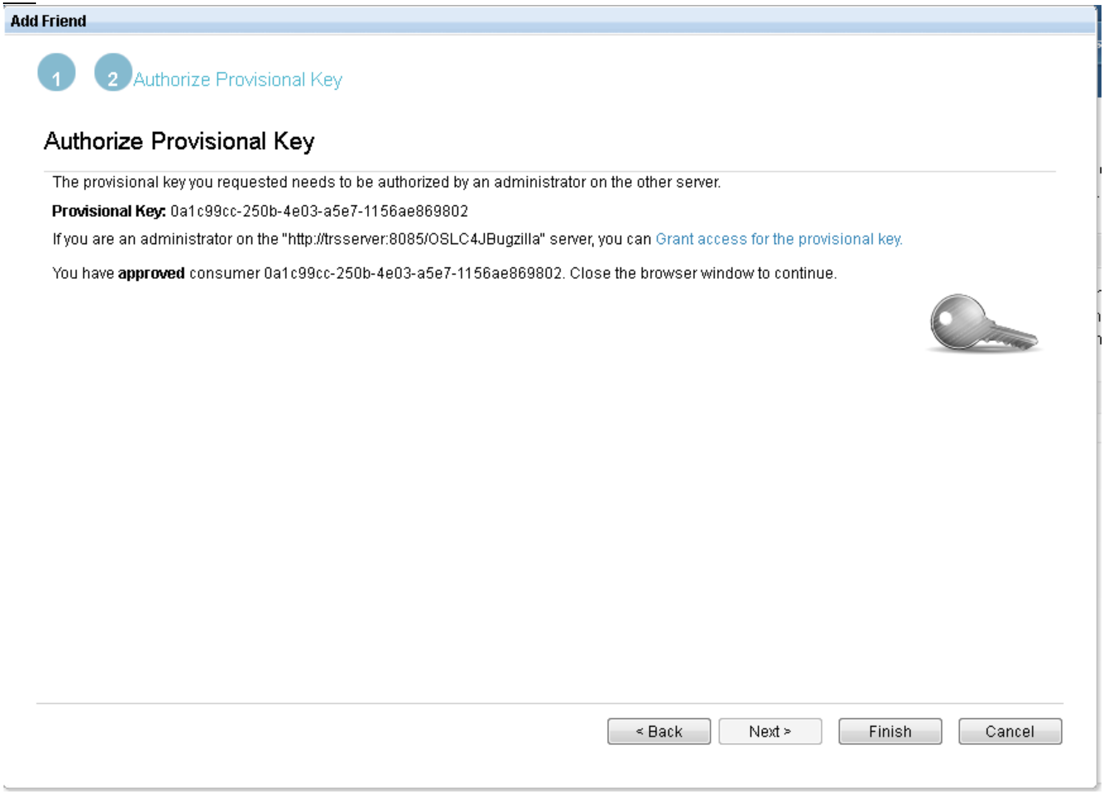
The provisional key you requested needs to be authorized by an administrator on the other server.

Provisional Key: 0a1c99cc-250b-4e03-a5e7-1156ae869802

If you are an administrator on the "http://trsserver:8085/OSLC4JBugzilla" server, you can Grant access for the provisional key.
Allow this consumer to access **Bugzilla**?

Name: LQE
Key: 0a1c99cc-250b-4e03-a5e7-1156ae869802
☑ Trusted

__i. Click **Finish**



Authorize Provisional Key

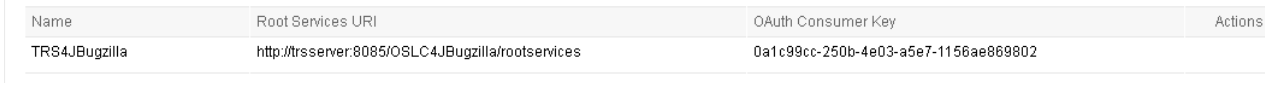
The provisional key you requested needs to be authorized by an administrator on the other server.

Provisional Key: 0a1c99cc-250b-4e03-a5e7-1156ae869802

If you are an administrator on the "http://trsserver:8085/OSLC4JBugzilla" server, you can Grant access for the provisional key.

You have **approved** consumer 0a1c99cc-250b-4e03-a5e7-1156ae869802. Close the browser window to continue.

__j. Now you have TRS4JBugzilla friend definition



Name	Root Services URI	OAuth Consumer Key	Actions
TRS4JBugzilla	http://trsserver:8085/OSLC4JBugzilla/rootservices	0a1c99cc-250b-4e03-a5e7-1156ae869802	

__k. Copy (or keep) the OAuth Consumer Key for the next step. This key value will be used later.

2.4 Add TRS provider to LQE as Data Source

You have configured OAuth key between TRS provider and LQE. So the next step is to add TRS provider to LQE as its Data Source in order for LQE to start the indexing. Also you can try Query by using SPARQL.

2.4.1 Steps

__1. In Browser, open LQE Data Sources page (https://trsserver:9443/lqe/web/admin/data-sources Note: "trsserver" is the host name where LQE runs.) and login it with the admin user id and its password. From this page, you are going to add TRS provider as LQE Data Source.

__a. Click **Add Data Source** button at the right top

__b. Fill the dialog as follow:

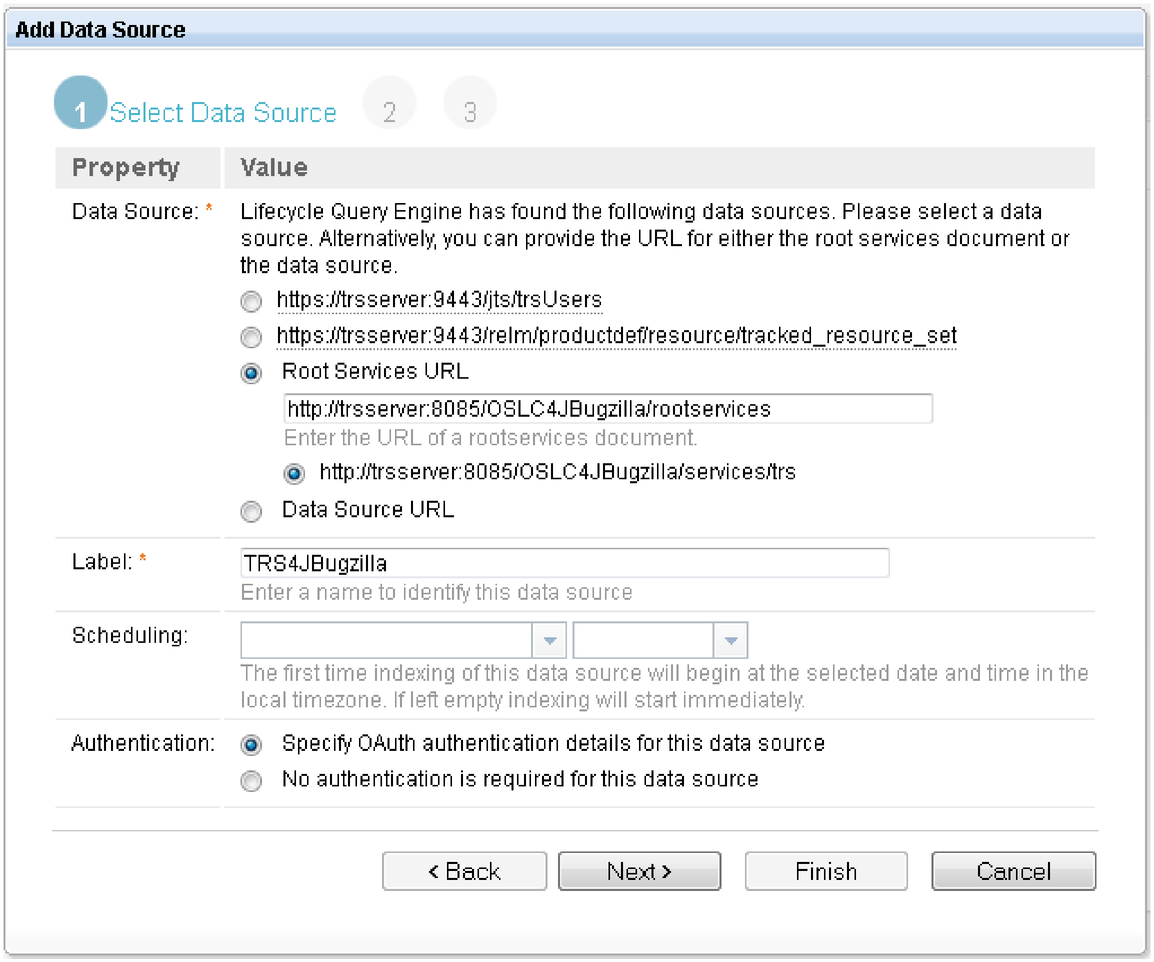
__i. Data Source: Check **Root Services document URL**

__ii. Put **http://trsserver:8085/OSLC4JBugzilla/rootservices** into the edit box. (Note: "trsserver" is the host name where Tracked Resource Set Provider runs. This host name is also specified in org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/resources/bugz.properties.)

__iii. Select **http://trsserver:8085/OSLC4JBugzilla/services/trs** radio button

__iv. Label: **TRS4JBugzilla**

__c. Click **Next**

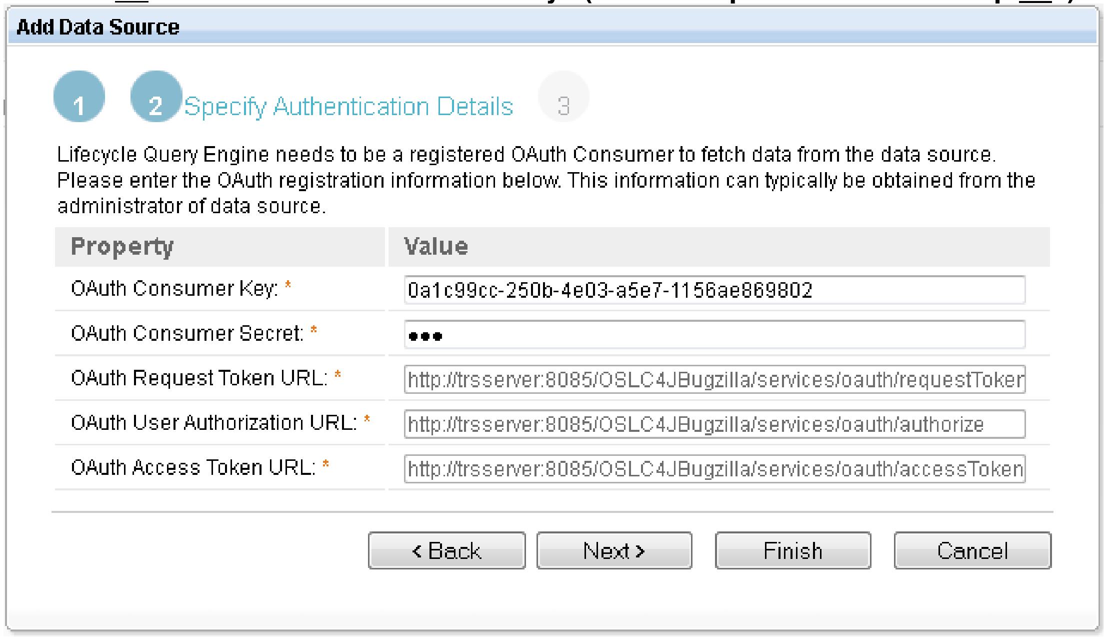

Add Data Source

1 Select Data Source 2 3

Property	Value
Data Source: *	Lifecycle Query Engine has found the following data sources. Please select a data source. Alternatively, you can provide the URL for either the root services document or the data source. ○ https://trsserver:9443/jts/trsUsers ○ https://trsserver:9443/relm/productdef/resource/tracked_resource_set ● Root Services URL http://trsserver:8085/OSLC4JBugzilla/rootservices Enter the URL of a rootservices document. ● http://trsserver:8085/OSLC4JBugzilla/services/trs ○ Data Source URL
Label: *	TRS4JBugzilla Enter a name to identify this data source
Scheduling:	The first time indexing of this data source will begin at the selected date and time in the local timezone. If left empty indexing will start immediately.
Authentication:	● Specify OAuth authentication details for this data source ○ No authentication is required for this data source

__d. Fill the form:

__i. OAuth Consumer Key: **[one copied in Friends screen in 2.3.1 Step __k]** __ii. OAuth Consumer Secret: **xyz (this was specified in 2.3.1 Step __c)**



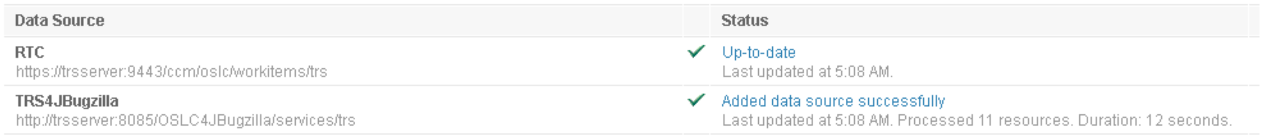
Lifecycle Query Engine needs to be a registered OAuth Consumer to fetch data from the data source. Please enter the OAuth registration information below. This information can typically be obtained from the administrator of data source.

Property	Value
OAuth Consumer Key: *	0a1c99cc-250b-4e03-a5e7-1156ae869802
OAuth Consumer Secret: *	●●●
OAuth Request Token URL: *	http://trsserver:8085/OSLC4JBugzilla/services/oauth/requestToker
OAuth User Authorization URL: *	http://trsserver:8085/OSLC4JBugzilla/services/oauth/authorize
OAuth Access Token URL: *	http://trsserver:8085/OSLC4JBugzilla/services/oauth/accessToken

__e. Click **Finish**. Now the TRS provider is added to LQE

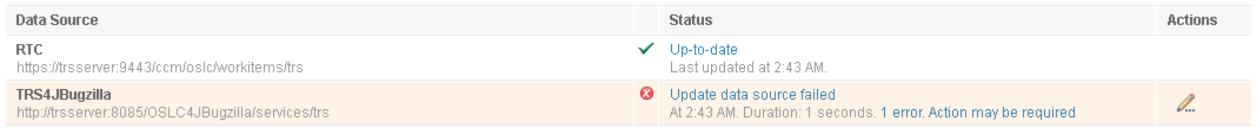
2.4.2 Time for the final tests for LQE integration

__1. Wait for the index to be completed.



Data Source	Status
RTC https://trsserver:9443/ccm/oslc/workitems/trs	✓ Up-to-date Last updated at 5:08 AM.
TRS4JBugzilla http://trsserver:8085/OSLC4JBugzilla/services/trs	✓ Added data source successfully Last updated at 5:08 AM. Processed 11 resources. Duration: 12 seconds.

If you will see a red X icon like :

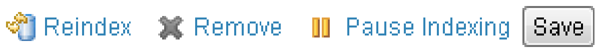


Data Source	Status	Actions
RTC https://trsserver:9443/ccm/oslc/workitems/trs	✓ Up-to-date Last updated at 2:43 AM.	
TRS4JBugzilla http://trsserver:8085/OSLC4JBugzilla/services/trs	✗ Update data source failed At 2:43 AM. Duration: 1 seconds. 1 error. Action may be required	✎

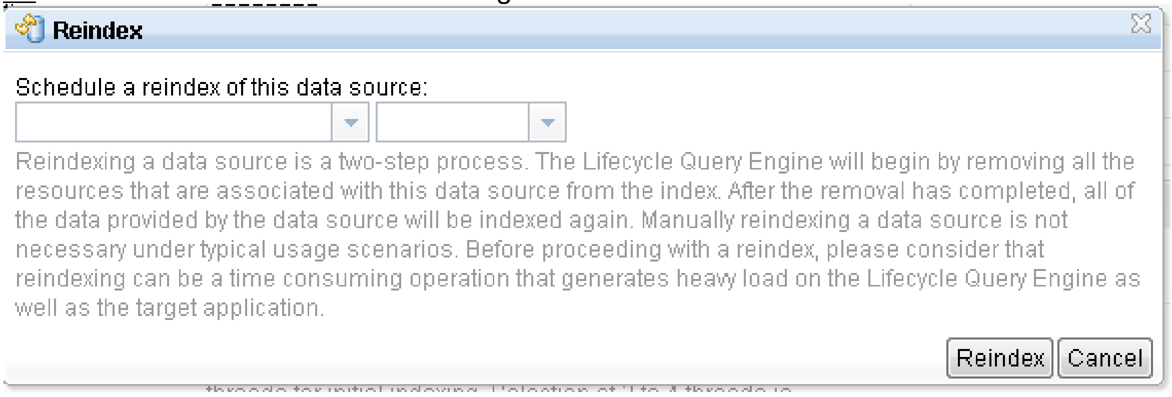
please try to re-index the resources by the following steps.

__a. Click a **pencil** icon.

__b. Click **Reindex** link.



__c. Click **Reindex** button in the dialog.

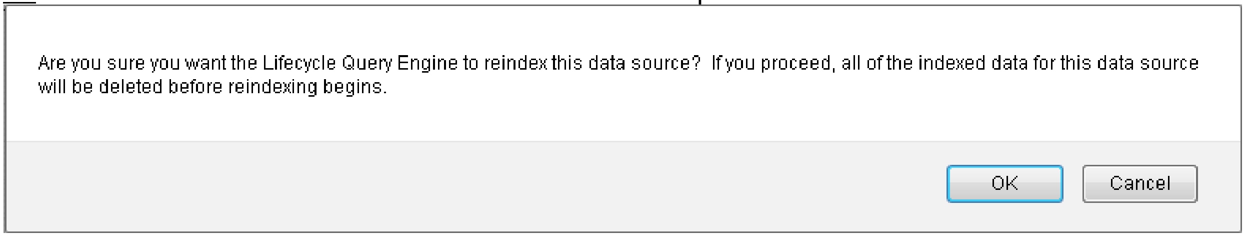


Reindex

Schedule a reindex of this data source:

Reindexing a data source is a two-step process. The Lifecycle Query Engine will begin by removing all the resources that are associated with this data source from the index. After the removal has completed, all of the data provided by the data source will be indexed again. Manually reindexing a data source is not necessary under typical usage scenarios. Before proceeding with a reindex, please consider that reindexing can be a time consuming operation that generates heavy load on the Lifecycle Query Engine as well as the target application.

__d. Click **OK** button to wait for the reindex to be completed.

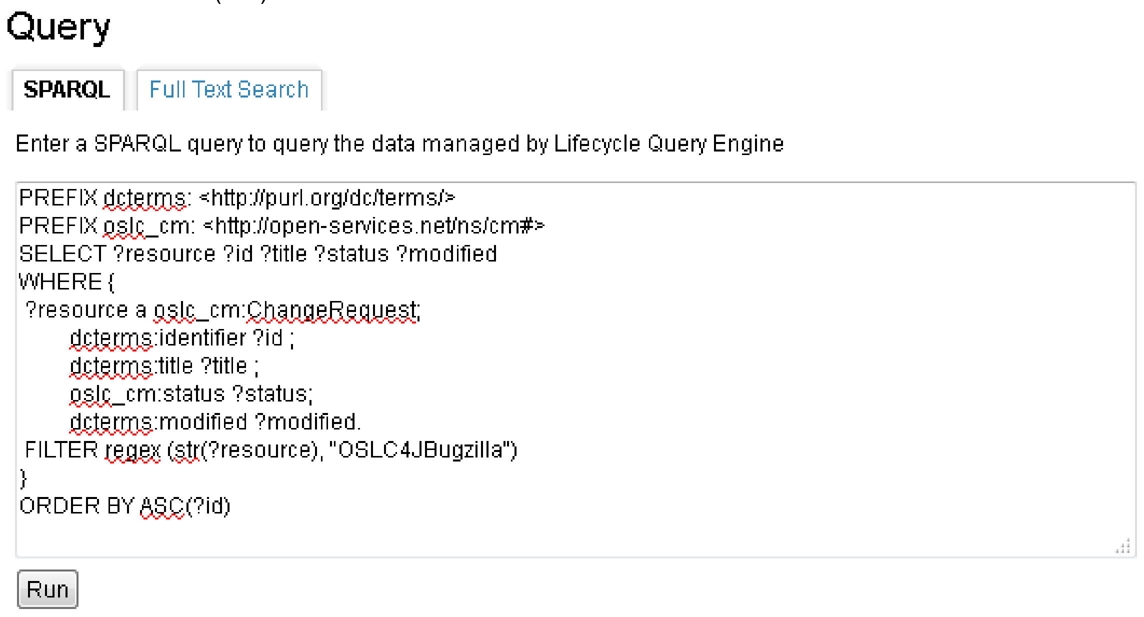


Are you sure you want the Lifecycle Query Engine to reindex this data source? If you proceed, all of the indexed data for this data source will be deleted before reindexing begins.

__2. Click **Query** link.

__3. Type the following SPARQL in the text field. (You can copy it from “SampleSPARQL.txt” file on the desktop) . This query will search and display all Bugs in Bugzilla with the url, the id, the title, the status and the modified date.

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX oslc_cm: <http://open-services.net/ns/cm#>
SELECT ?resource ?id ?title ?status ?modified
WHERE {
 ?resource a oslc_cm:ChangeRequest;
        dcterms:identifier ?id ;
        dcterms:title ?title ;
        oslc_cm:status ?status;
       dcterms:modified ?modified.
 FILTER regex (str(?resource), "OSLC4JBugzilla")
}
ORDER BY ASC(?id)
```



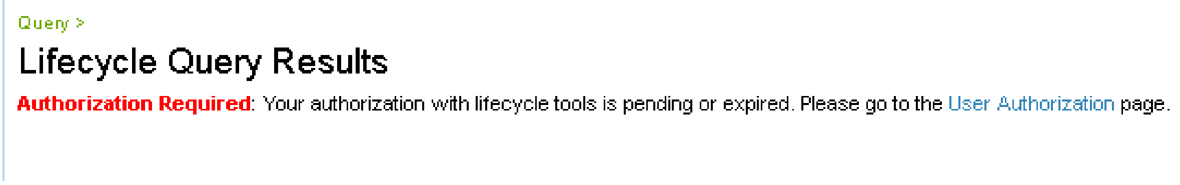
Enter a SPARQL query to query the data managed by Lifecycle Query Engine

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX oslc_cm: <http://open-services.net/ns/cm#>
SELECT ?resource ?id ?title ?status ?modified
WHERE {
 ?resource a oslc_cm:ChangeRequest;
        dcterms:identifier ?id ;
        dcterms:title ?title ;
        oslc_cm:status ?status;
        dcterms:modified ?modified.
 FILTER regex (str(?resource), "OSLC4JBugzilla")
}
ORDER BY ASC(?id)
```

Note: The details about SPARQL are available at http://www.w3.org/TR/sparql11-query/.

__4. Click **Run** button

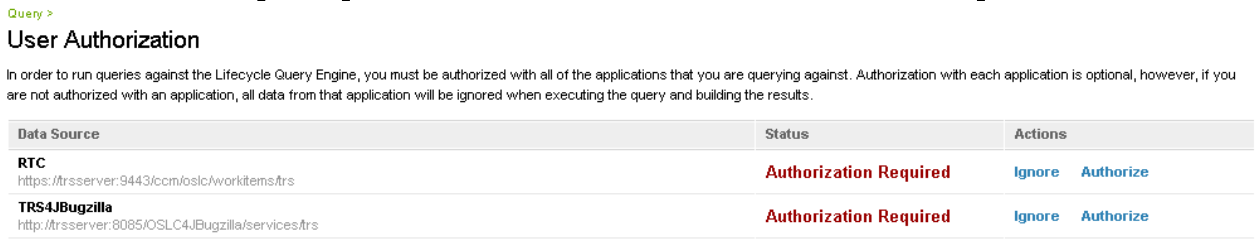
__5. If you see the following message, click **User Authorization**. If not, you can skip to step __6



Lifecycle Query Results

Authorization Required: Your authorization with lifecycle tools is pending or expired. Please go to the User Authorization page.

__a. In the following dialog, click **Authorize** links for both RTC and TRS4Bugzilla



Query >

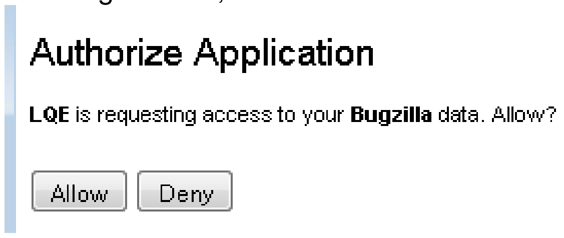
User Authorization

In order to run queries against the Lifecycle Query Engine, you must be authorized with all of the applications that you are querying against. Authorization with each application is optional, however, if you are not authorized with an application, all data from that application will be ignored when executing the query and building the results.

Data Source	Status	Actions
RTC https://trsserver:9443/ccm/oslc/workitems/trs	Authorization Required	Ignore Authorize
TRS4JBugzilla http://trsserver:8085/OSLC4JBugzilla/services/trs	Authorization Required	Ignore Authorize

__b. For RTC, you may not be asked, but if you are asked the user id and its password, fill "trsuser" for the user id and "passw0rd" for its password

__c. For TRS4JBugzilla, you may not be asked, but if you are asked the user id and its password, fill "trsuser@localhost.here" for the user id and "passw0rd" for its password. Or if you see the following window, click **Allow**. button

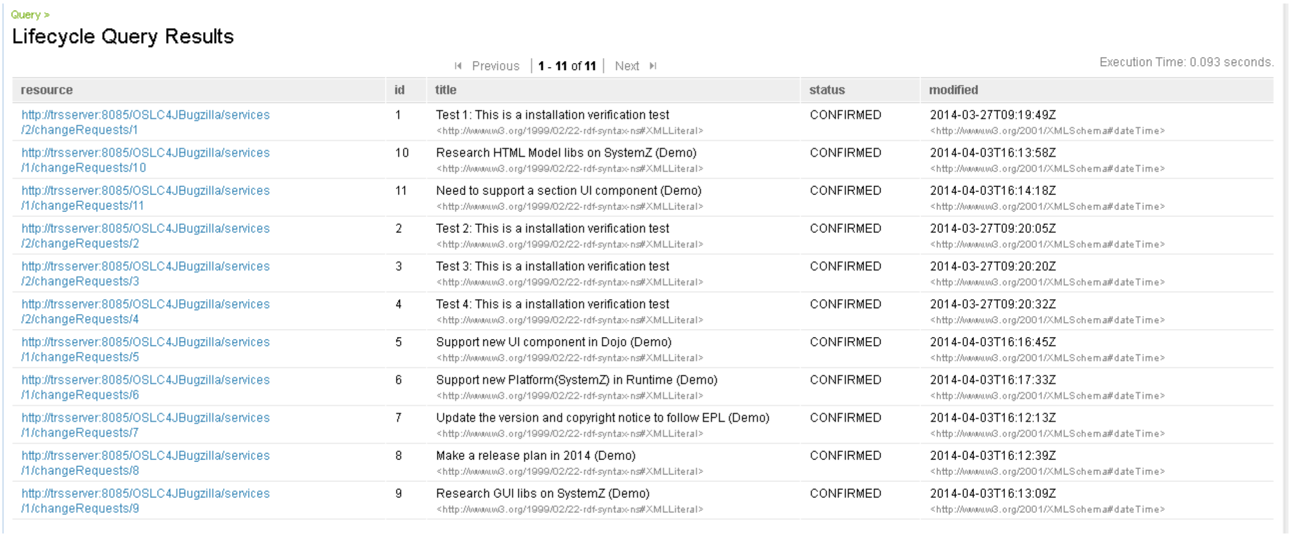


Authorize Application

LQE is requesting access to your **Bugzilla** data. Allow?

__d. Click **Query** link and **Run** button again

__6. You will see some resources in the result screen



Query >

Lifecycle Query Results

Previous | **1 - 11** of **11** | Next

Execution Time: 0.093 seconds.

resource	id	title	status	modified
http://trsserver:8085/OSLC4JBugzilla/services/2/changeRequests/1	1	Test 1: This is a installation verification test <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-03-27T09:19:49Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/10	10	Research HTML Model libs on SystemZ (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:13:58Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/11	11	Need to support a section UI component (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:14:18Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/2/changeRequests/2	2	Test 2: This is a installation verification test <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-03-27T09:20:05Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/2/changeRequests/3	3	Test 3: This is a installation verification test <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-03-27T09:20:20Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/2/changeRequests/4	4	Test 4: This is a installation verification test <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-03-27T09:20:32Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/5	5	Support new UI component in Dojo (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:16:45Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/6	6	Support new Platform(SystemZ) in Runtime (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:17:33Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/7	7	Update the version and copyright notice to follow EPL (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:12:13Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/8	8	Make a release plan in 2014 (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:12:39Z <http://www.w3.org/2001/XMLSchema#dateTime>
http://trsserver:8085/OSLC4JBugzilla/services/1/changeRequests/9	9	Research GUI libs on SystemZ (Demo) <http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral>	CONFIRMED	2014-04-03T16:13:09Z <http://www.w3.org/2001/XMLSchema#dateTime>

2.5 Summary

This lab provided requires steps to implement OAuth authentication and to integrate your TRS provider with LQE. You now have basic understandings of the steps and how to achieve them. Also you have learned how to configure OAuth key between LQE and TRS provider in order for LQE to start indexing of the resources TRS provider provides, and how to search the data in such indexes.

Lab 3 Integrating TRS provider with Rational Engineering Lifecycle Manager (RELM) (optional)

Objectives

- Understand how TRS provider can be integrated with Rational Engineering Lifecycle Manager (RELM)
- Understand how RELM uses indirectly TRS resources through LQE to provide its Query an View functions
- This is an optional lab and can be skipped if you are constrained for time.

Description:

Now you have integrated TRS provider with LQE, and LQE now indexes the resources TRS provider provides. As you have learned in the previous lab, such indexed data can be searched by using SPARQL query language. Rational Engineering Lifecycle Manager (RELM) uses SPARQL to extract indexed resources from LQE database and provides the additional capabilities, such as the visualization of the resources.

This lab will explore the steps necessary to configure TRS provider and RELM to be integrated. So that you can try RELM visualization capability, such as traceability view of Bugs in Bugzilla and Workitems in Rational Team Concert (RTC) as the example of the usage of the indexed resources by LQE and TRS providers.

3.1 Configure RELM to be connected to TRS provider for the preview.

You have added TRS provider to LQE's Data Source. So the next step is to configure RELM to be able to preview the indexed resources. The preview function is nothing to do with TRS but is provided by OSLC provider. Since TRS provider in this workshop is based on OSLC provider, RELM can preview the indexed resource.

3.1.1 Steps

__1. In Browser, open RELM admin page (https://trsserver:9443/relm/admin Note: "trsserver" is the host name where RELM runs.) and login it with the admin user id and its password. From this page, you are going to ask TRS provider to generate OAuth Consumer Key and its OAuth Consumer Secret for RELM to access resources TRS provider provides. Once this OAuth key is configured, RELM can visualize the preview of such resources by using OSLC REST API.

__a. Click **Friends (Outbound)** in the left pane

__b. Click **Add...** link at the right of the Friends List section title

__c. Fill the dialog as follow:

__i. Name: **TRS4JBugzilla**

__ii. Root Services URI: **http://trsserver:8085/OSLC4JBugzilla/rootservices** (Note: “trsserver” is the host name where Tracked Resource Set Provider runs. This host name is also specified in org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/resources/bugz.properties.)

__iii. OAuth Secret: **xyz**

__iv. Trusted: **Checked**

__d. Click **Create Friend,** click **Next**

__e. Click **Grant access for the provisional key** link

__f. Fill the form using [[**Bugzilla admin id** (default root@localhost.here)]] and [[**Bugzilla admin password** (passw0rd)]], and then click **Continue**. (NOTE: You may not see the login message if you have already logged in to the TRS4JBugzilla adapter as the administrator.)

__g. Type *RELM* in **Name** field, check **Trusted** checkbox, and then click **Allow** button

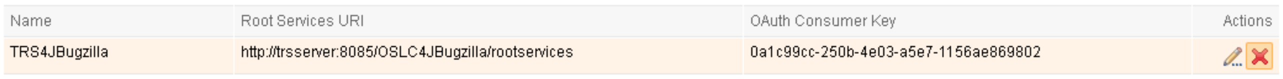
__h. Click **Finish**

__2. If you will change the value of the property, **adapter_host**, in **org.eclipse.lyo.oslc4j.bugzilla.trs/src/main/resources/bugz.properties**, you need to remove the generated OAuth Consumer Key and Secret in the previous steps. (In this workshop, you can skip the following steps and jump to step 3.1.2) The steps to remove them are :

__a. Open RELM JTS admin page (https://trsserver:9443/jts/admin Note: “trsserver” is the host name where RELM JTS runs.) and login it with the admin user id and its password

__b. Click **Manage Server** in Manage Server pane, and then navigate to **Friends (Outbound)** in the left pane.

__c. Click a red X icon in **Actions** field for "**TRS4JBugzilla**" to remove it.



Name	Root Services URI	OAuth Consumer Key	Actions
TRS4JBugzilla	http://trsserver:8085/OSLC4JBugzilla/rootservices	0a1c99cc-250b-4e03-a5e7-1156ae869802	✎ ✗

__d. Open RELM admin page (https://trsserver:9443/relm/admin Note: “trsserver” is the host name where RELM runs.) and login it with the admin user id and its password

__e. Click **Friends (Outbound)** in the left pane

__f. Click a red X icon in **Actions** field for "**TRS4JBugzilla**" to remove it.

__g. Remove **org.eclipse.lyo.docs/org.eclipse.lyo.oslc4j.bugzilla.trs/bugzillaOAuthStore.xml** file.

__h. Copy **org.eclipse.lyo.docs/org.eclipse.lyo.oslc4j.bugzilla.trs/ bugzillaOAuthStore_ORG_CCMOnly.xml** to **org.eclipse.lyo.docs/org.eclipse.lyo.oslc4j.bugzilla.trs/bugzillaOAuthStore.xml** (**bugzillaOAuthStore_ORG_CCMOnly.xml** is a back up file which has OAuth key for CCM)

__i. Restart TRS provider

3.1.2 Time for the final tests for RELM integration

__1. In Browser, logout if you have still logged in the application (JTS, RELM or LQE) in order to clear the credentials.

__2. Open RELM page (https://trsserver:9443/relm/admin Note: "trsserver" is the host name where RELM runs.) and login it with the test user id and its password

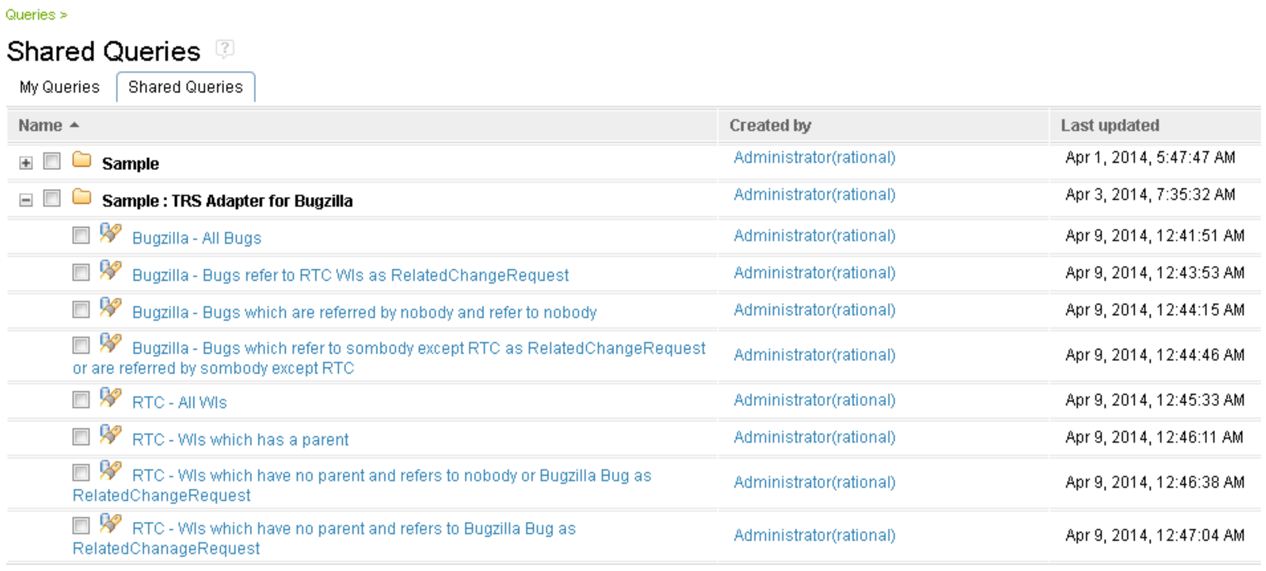
In the given VM image, a test user id for RELM is "**trsuser**" and its password is "**passw0rd**".

__3. Visit RELM Project Area : **DemoProduct**

In the given VM image, "**DemoProduct**" is created already.

__4. Select **Queries** -> **Browse Queries** -> **Shared Queries**

__5. Open **Sample : TRS Adapter for Bugzilla** so that you will see "**Bugzilla – All Bugs**". This query will search all Bugs in Bugzilla. Click it.



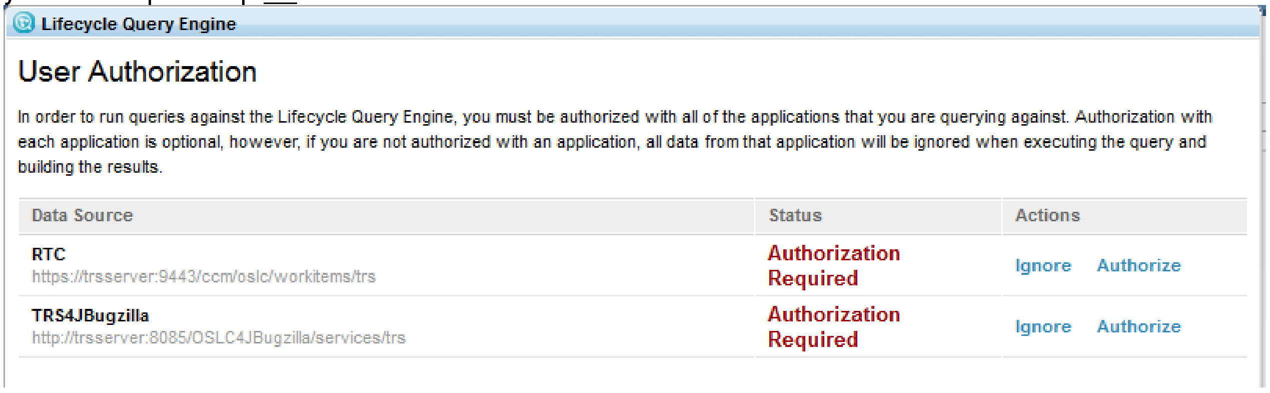
Queries >

Shared Queries

My Queries | Shared Queries

Name ▲	Created by	Last updated
Sample	Administrator(rational)	Apr 1, 2014, 5:47:47 AM
Sample : TRS Adapter for Bugzilla	Administrator(rational)	Apr 3, 2014, 7:35:32 AM
Bugzilla - All Bugs	Administrator(rational)	Apr 9, 2014, 12:41:51 AM
Bugzilla - Bugs refer to RTC WIs as RelatedChangeRequest	Administrator(rational)	Apr 9, 2014, 12:43:53 AM
Bugzilla - Bugs which are referred by nobody and refer to nobody	Administrator(rational)	Apr 9, 2014, 12:44:15 AM
Bugzilla - Bugs which refer to sombody except RTC as RelatedChangeRequest or are referred by sombody except RTC	Administrator(rational)	Apr 9, 2014, 12:44:46 AM
RTC - All WIs	Administrator(rational)	Apr 9, 2014, 12:45:33 AM
RTC - WIs which has a parent	Administrator(rational)	Apr 9, 2014, 12:46:11 AM
RTC - WIs which have no parent and refers to nobody or Bugzilla Bug as RelatedChangeRequest	Administrator(rational)	Apr 9, 2014, 12:46:38 AM
RTC - WIs which have no parent and refers to Bugzilla Bug as RelatedChanageRequest	Administrator(rational)	Apr 9, 2014, 12:47:04 AM

__6. If you see the following dialog, click **Authorize** links for both RTC and TRS4Bugzilla. Otherwise you can skip to step __7



Lifecycle Query Engine

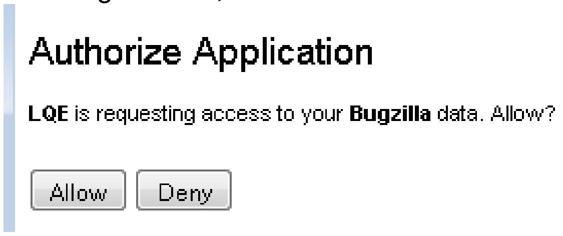
User Authorization

In order to run queries against the Lifecycle Query Engine, you must be authorized with all of the applications that you are querying against. Authorization with each application is optional, however, if you are not authorized with an application, all data from that application will be ignored when executing the query and building the results.

Data Source	Status	Actions
RTC https://trsserver:9443/ccm/oslc/workitems/trs	Authorization Required	Ignore Authorize
TRS4JBugzilla http://trsserver:8085/OSLC4JBugzilla/services/trs	Authorization Required	Ignore Authorize

__a. For RTC, you may not be asked, but if you are asked the user id and its password, fill "trsuser" for the user id and "passw0rd" for its password

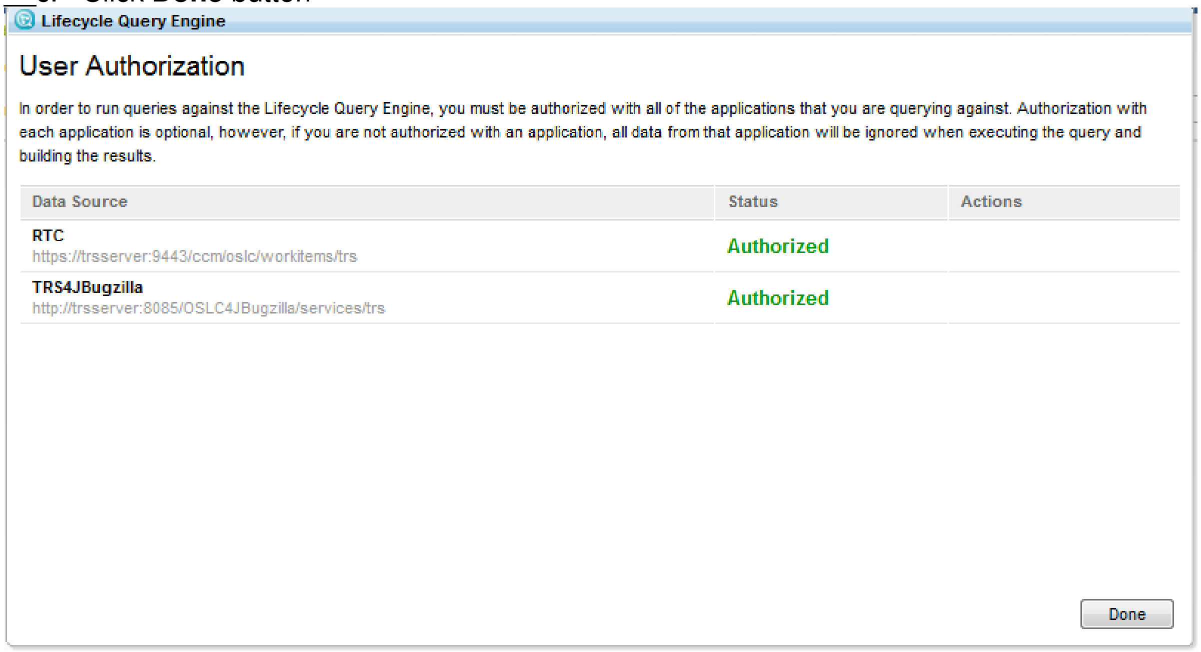
__b. For TRS4JBugzilla, you may not be asked, but if you are asked the user id and its password, fill "trsuser@localhost.here" for the user id and "passw0rd" for its password. Or if you see the following window, click **Allow**. button



Authorize Application

LQE is requesting access to your **Bugzilla** data. Allow?

__c. Click **Done** button



Lifecycle Query Engine

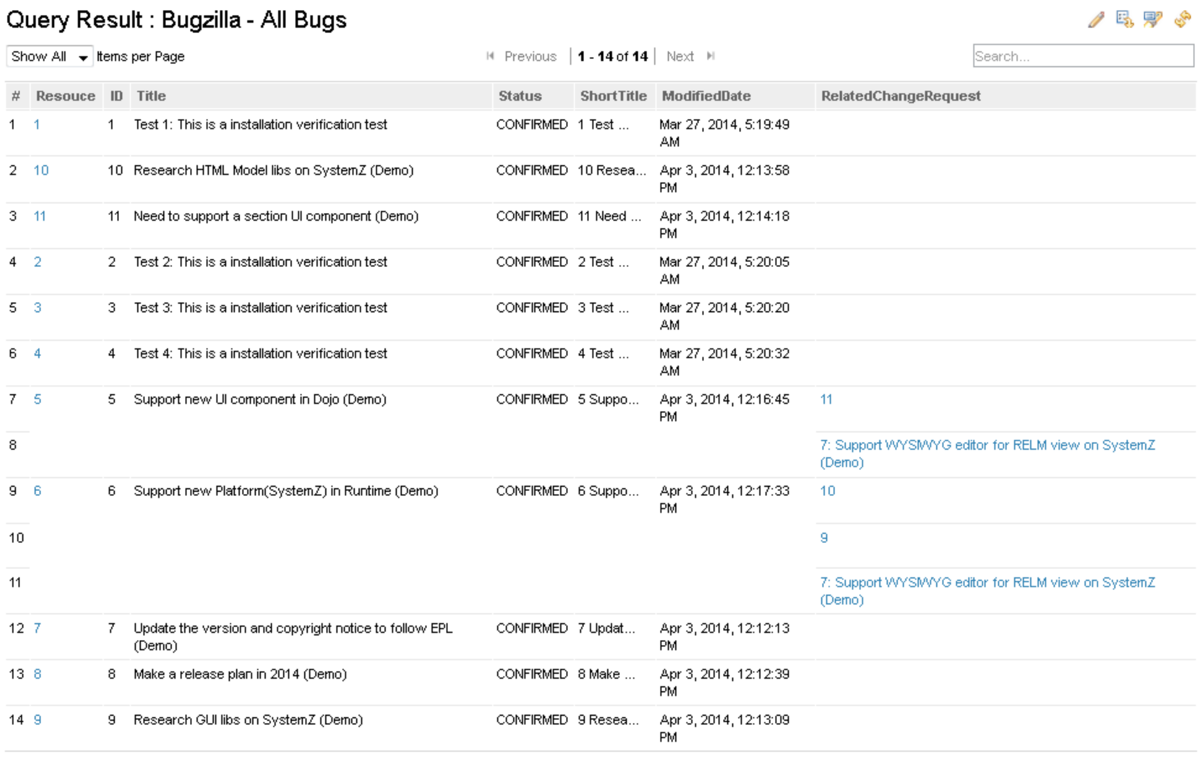
User Authorization

In order to run queries against the Lifecycle Query Engine, you must be authorized with all of the applications that you are querying against. Authorization with each application is optional, however, if you are not authorized with an application, all data from that application will be ignored when executing the query and building the results.

Data Source	Status	Actions
RTC https://trsserver:9443/ccm/oslc/workitems/trs	Authorized	
TRS4JBugzilla http://trsserver:8085/OSLC4JBugzilla/services/trs	Authorized	

__7. You will see the following result. The content of this result is

- **Resource** : Link(URL) to Bug. Either **ID** or **Summary** is displayed instead of URL itself.
- **ID** : Bug's **ID**
- **Title** : Bug's **Summary**
- **Status** : Bug's **Status**
- **ShortTitle** : Bug's **ID** + the first five characters of **Summary** + "…"
- **ModifiedDate** : Bug's **Modified** date.
- **RelatedChangeRequest** : Links which are specified in **URL** or **See Also** field. Links to RTC Workitems might display its **ID** and **Summary** instead.

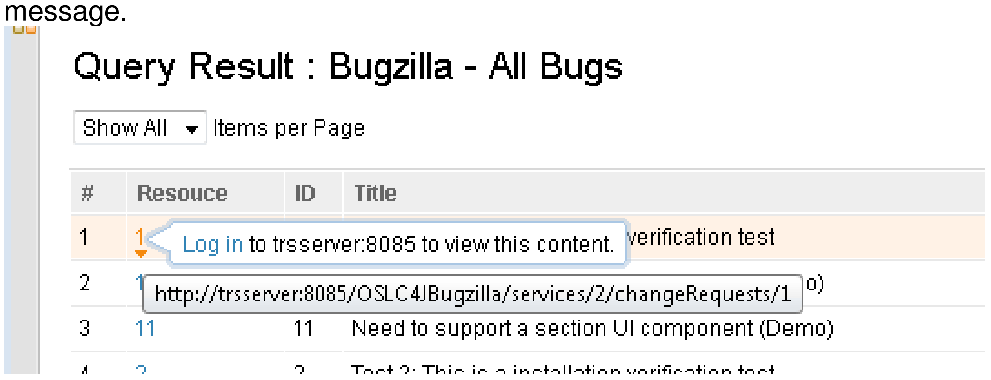


Query Result : Bugzilla - All Bugs

Show All ▾ Items per Page   Previous | 1 - 14 of 14 | Next   Search...

#	Resouce	ID	Title	Status	ShortTitle	ModifiedDate	RelatedChangeRequest
1	1	1	Test 1: This is a installation verification test	CONFIRMED	1 Test ...	Mar 27, 2014, 5:19:49 AM	
2	10	10	Research HTML Model libs on SystemZ (Demo)	CONFIRMED	10 Resea...	Apr 3, 2014, 12:13:58 PM	
3	11	11	Need to support a section UI component (Demo)	CONFIRMED	11 Need ...	Apr 3, 2014, 12:14:18 PM	
4	2	2	Test 2: This is a installation verification test	CONFIRMED	2 Test ...	Mar 27, 2014, 5:20:05 AM	
5	3	3	Test 3: This is a installation verification test	CONFIRMED	3 Test ...	Mar 27, 2014, 5:20:20 AM	
6	4	4	Test 4: This is a installation verification test	CONFIRMED	4 Test ...	Mar 27, 2014, 5:20:32 AM	
7	5	5	Support new UI component in Dojo (Demo)	CONFIRMED	5 Suppo...	Apr 3, 2014, 12:16:45 PM	11
8							7: Support WYSMYG editor for RELM view on SystemZ (Demo)
9	6	6	Support new Platform(SystemZ) in Runtime (Demo)	CONFIRMED	6 Suppo...	Apr 3, 2014, 12:17:33 PM	10
10							9
11							7: Support WYSMYG editor for RELM view on SystemZ (Demo)
12	7	7	Update the version and copyright notice to follow EPL (Demo)	CONFIRMED	7 Updat...	Apr 3, 2014, 12:12:13 PM	
13	8	8	Make a release plan in 2014 (Demo)	CONFIRMED	8 Make ...	Apr 3, 2014, 12:12:39 PM	
14	9	9	Research GUI libs on SystemZ (Demo)	CONFIRMED	9 Resea...	Apr 3, 2014, 12:13:09 PM	

__8. Move the cursor in **Resource** column, you will see the hover window to tell you “**Log in …**” message.



Query Result : Bugzilla - All Bugs

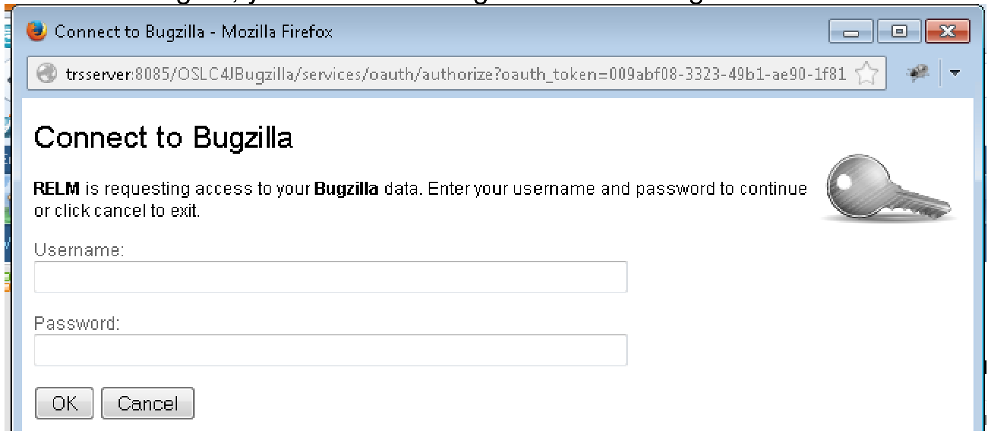
Show All ▾ Items per Page

#	Resouce	ID	Title
1	1		verification test
2	1		o)
3	11	11	Need to support a section UI component (Demo)
4	2	2	Test 2: This is a installation verification test

Log in to trsserver:8085 to view this content.

http://trsserver:8085/OSLC4JBugzilla/services/2/changeRequests/1

__9.	Click “Log in”, you will see the login window to Bugzilla


Connect to Bugzilla - Mozilla Firefox

trsserver:8085/OSLC4JBugzilla/services/oauth/authorize?oauth_token=009abf08-3323-49b1-ae90-1f81

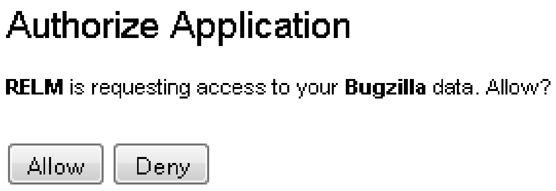
Connect to Bugzilla

RELM is requesting access to your **Bugzilla** data. Enter your username and password to continue or click cancel to exit.

Username:

Password:

__10.	Or if you see the following dialog instead, click **Allow** button. Go to the step __11

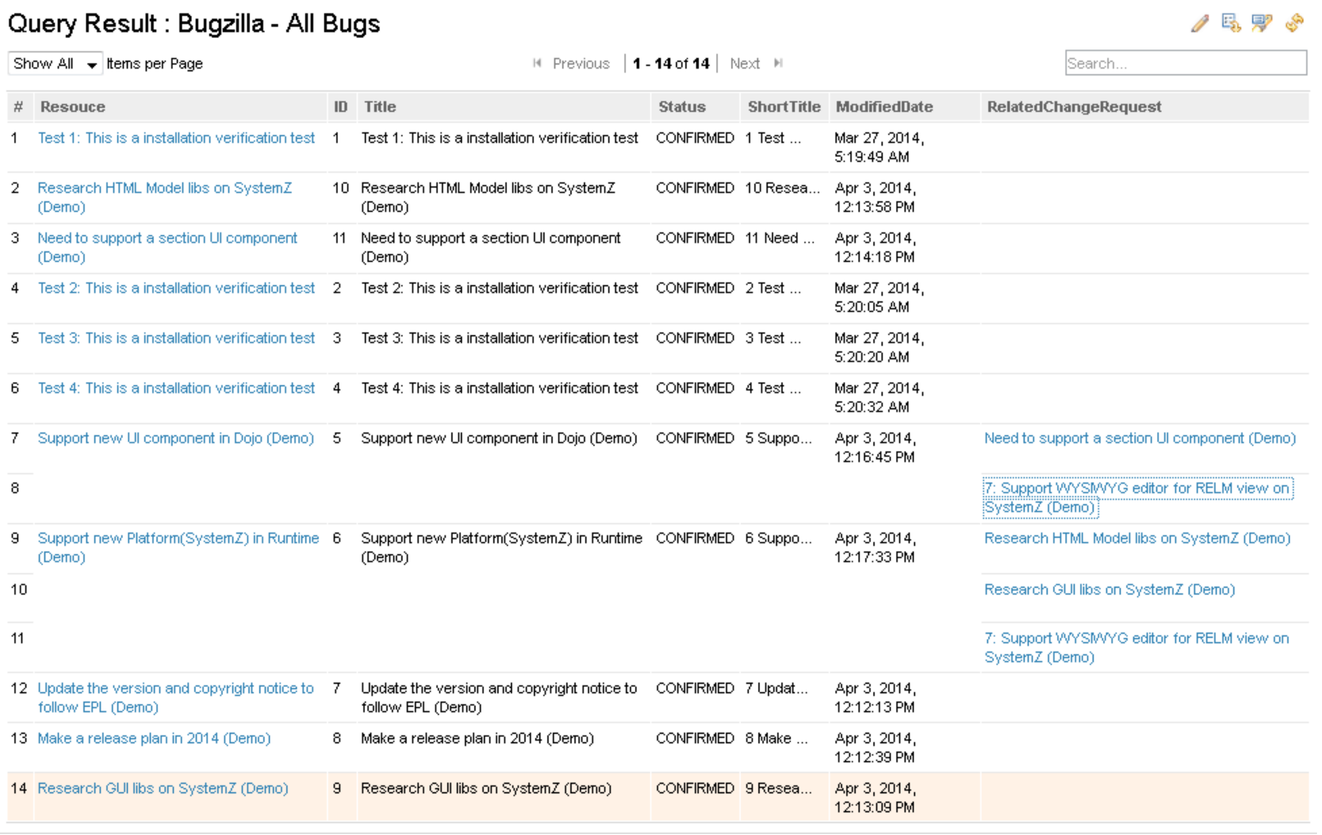

Authorize Application

RELM is requesting access to your **Bugzilla** data. Allow?

__11.	Try to login with the test user for Bugzilla.

In the given VM image, a test user id for Bugzilla is “**trsuser@localhost.here**” and its password is “**passw0rd**”.

__12. Wait for a while. You will see the title to be displayed in **Resource** and **RelatedChangeRequest** columns. This is an enhancement of RELM's Query. LQE's Query doesn't have this alternative display capability.

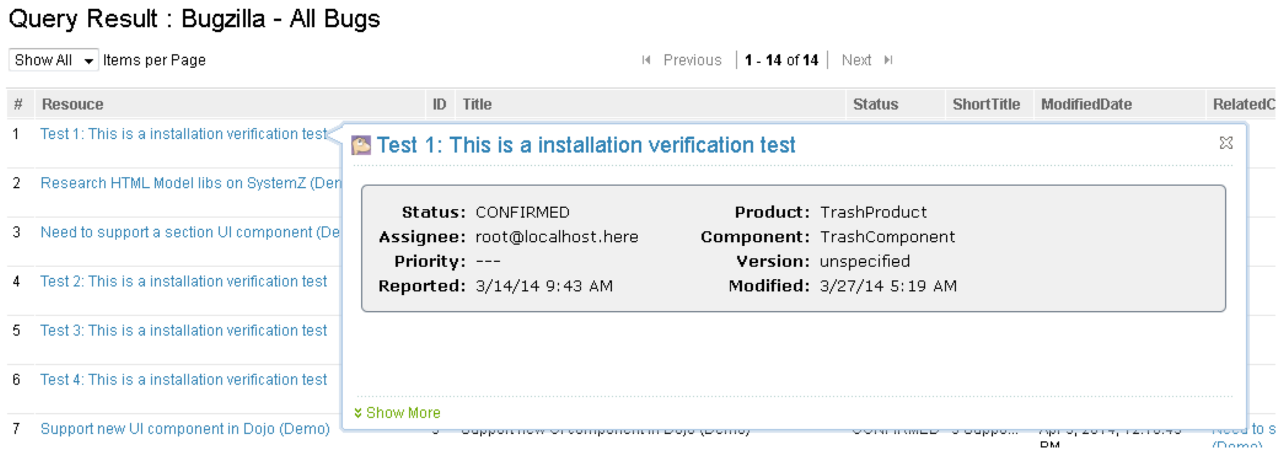


Query Result : Bugzilla - All Bugs

Show All Items per Page   Previous | 1 - 14 of 14 | Next   Search...

#	Resouce	ID	Title	Status	ShortTitle	ModifiedDate	RelatedChangeRequest
1	Test 1: This is a installation verification test	1	Test 1: This is a installation verification test	CONFIRMED	1 Test ...	Mar 27, 2014, 5:19:49 AM	
2	Research HTML Model libs on SystemZ (Demo)	10	Research HTML Model libs on SystemZ (Demo)	CONFIRMED	10 Resea...	Apr 3, 2014, 12:13:58 PM	
3	Need to support a section UI component (Demo)	11	Need to support a section UI component (Demo)	CONFIRMED	11 Need ...	Apr 3, 2014, 12:14:18 PM	
4	Test 2: This is a installation verification test	2	Test 2: This is a installation verification test	CONFIRMED	2 Test ...	Mar 27, 2014, 5:20:05 AM	
5	Test 3: This is a installation verification test	3	Test 3: This is a installation verification test	CONFIRMED	3 Test ...	Mar 27, 2014, 5:20:20 AM	
6	Test 4: This is a installation verification test	4	Test 4: This is a installation verification test	CONFIRMED	4 Test ...	Mar 27, 2014, 5:20:32 AM	
7	Support new UI component in Dojo (Demo)	5	Support new UI component in Dojo (Demo)	CONFIRMED	5 Suppo...	Apr 3, 2014, 12:16:45 PM	Need to support a section UI component (Demo)
8							7: Support WYSIWYG editor for RELM view on SystemZ (Demo)
9	Support new Platform(SystemZ) in Runtime (Demo)	6	Support new Platform(SystemZ) in Runtime (Demo)	CONFIRMED	6 Suppo...	Apr 3, 2014, 12:17:33 PM	Research HTML Model libs on SystemZ (Demo)
10							Research GUI libs on SystemZ (Demo)
11							7: Support WYSIWYG editor for RELM view on SystemZ (Demo)
12	Update the version and copyright notice to follow EPL (Demo)	7	Update the version and copyright notice to follow EPL (Demo)	CONFIRMED	7 Updat...	Apr 3, 2014, 12:12:13 PM	
13	Make a release plan in 2014 (Demo)	8	Make a release plan in 2014 (Demo)	CONFIRMED	8 Make ...	Apr 3, 2014, 12:12:39 PM	
14	Research GUI libs on SystemZ (Demo)	9	Research GUI libs on SystemZ (Demo)	CONFIRMED	9 Resea...	Apr 3, 2014, 12:13:09 PM	

__13. Move the mouse cursor in **Resource** column again. You will see the small preview window with "**Show more**" link.



Query Result : Bugzilla - All Bugs

Show All Items per Page   Previous | 1 - 14 of 14 | Next

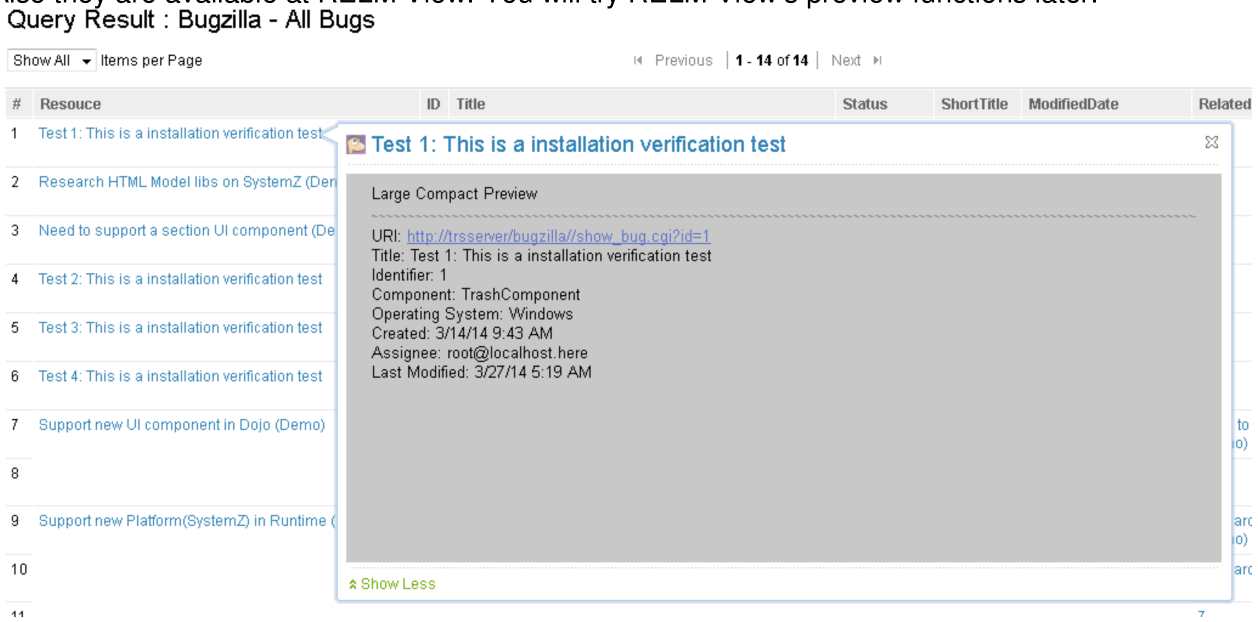
#	Resouce	ID	Title	Status	ShortTitle	ModifiedDate	RelatedC
1	Test 1: This is a installation verification test						
2	Research HTML Model libs on SystemZ (Den						
3	Need to support a section UI component (De						
4	Test 2: This is a installation verification test						
5	Test 3: This is a installation verification test						
6	Test 4: This is a installation verification test						
7	Support new UI component in Dojo (Demo)						

Test 1: This is a installation verification test

Status: CONFIRMED **Product:** TrashProduct
Assignee: root@localhost.here **Component:** TrashComponent
Priority: --- **Version:** unspecified
Reported: 3/14/14 9:43 AM **Modified:** 3/27/14 5:19 AM

Show More

__14. Click “Show More” link in the preview window. You will see the large preview window. These small and large preview functions are implemented by REST API which is provide by OSLC provider. Also they are available at RELM View. You will try RELM View’s preview functions later.



Query Result : Bugzilla - All Bugs

Show All Items per Page   Previous | 1 - 14 of 14 | Next

#	Resouce	ID	Title	Status	ShortTitle	ModifiedDate	Related
1	Test 1: This is a installation verification test						
2	Research HTML Model libs on SystemZ (Den						
3	Need to support a section UI component (De						
4	Test 2: This is a installation verification test						
5	Test 3: This is a installation verification test						
6	Test 4: This is a installation verification test						
7	Support new UI component in Dojo (Demo)						
8							
9	Support new Platform(SystemZ) in Runtime (						
10							

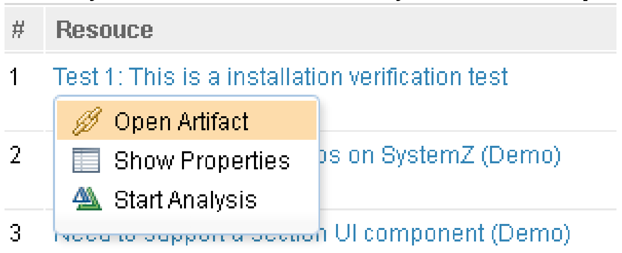
Test 1: This is a installation verification test

Large Compact Preview

URI: http://trsserver/bugzilla//show_bug.cgi?id=1
Title: Test 1: This is a installation verification test
Identifier: 1
Component: TrashComponent
Operating System: Windows
Created: 3/14/14 9:43 AM
Assignee: root@localhost.here
Last Modified: 3/27/14 5:19 AM

Show Less

__15. If you click the left button, you will see “**Open Artifact**” menu in its context menu.



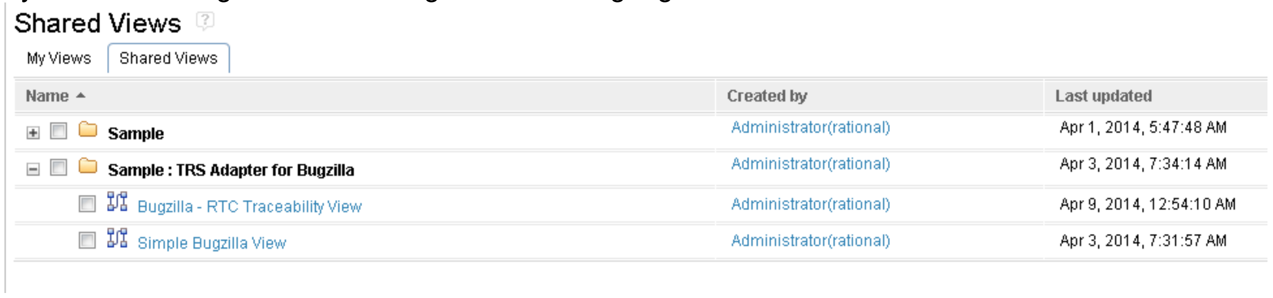
#	Resouce
1	Test 1: This is a installation verification test
2	…s on SystemZ (Demo)
3	…n UI component (Demo)

If you will click “Open Artifact”, the associated tool (in this case, Bugzilla) will be opened to display the selected resource.

__16. You can try other supplied Queries if you like.

__17. Select **Welcome** -> **Views** -> **Browse Views** -> **Shared Views**

__18. Open **Sample : TRS Adapter for Bugzilla** so that you will see "**Bugzilla – RTC Traceability View**". This view will show some Bugs in Bugzilla in "DemoProduct" product, some Workitems in RTC in "DemoProduct" project area, and the links within them. This visualization can be configured by the view configuration including SPARQL language. Click it



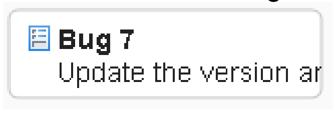
Shared Views

My Views | Shared Views

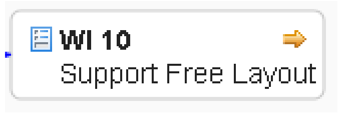
Name	Created by	Last updated
Sample	Administrator(rational)	Apr 1, 2014, 5:47:48 AM
Sample : TRS Adapter for Bugzilla	Administrator(rational)	Apr 3, 2014, 7:34:14 AM
Bugzilla - RTC Traceability View	Administrator(rational)	Apr 9, 2014, 12:54:10 AM
Simple Bugzilla View	Administrator(rational)	Apr 3, 2014, 7:31:57 AM

__19. You will see the following result.

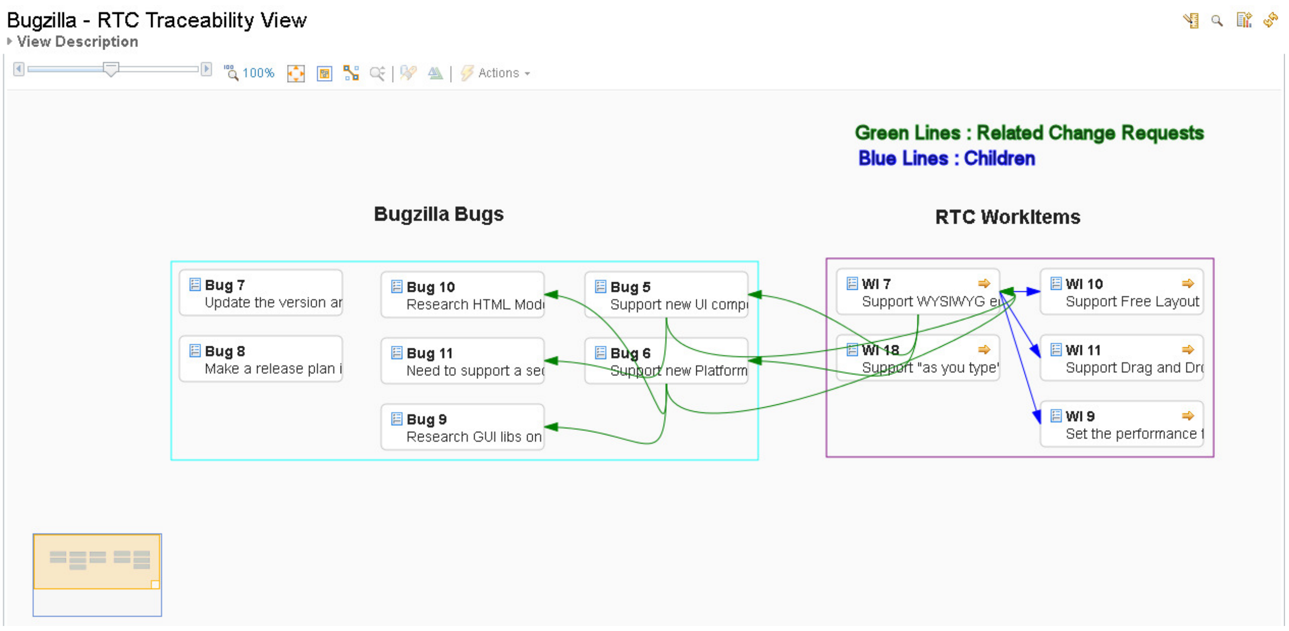
- The rectangle under **Bugzilla Bugs** has the following information.
  - The first row : "Bug" + Bug's **ID**
  - The second row : Bug's **Summary**



- The rectangle under **RTC Workitems** has the following information.
  - The first row : "WI" + Workitem's **ID**. The icon stands for its **Status**.
  - The second row : Workitem's **Summary**

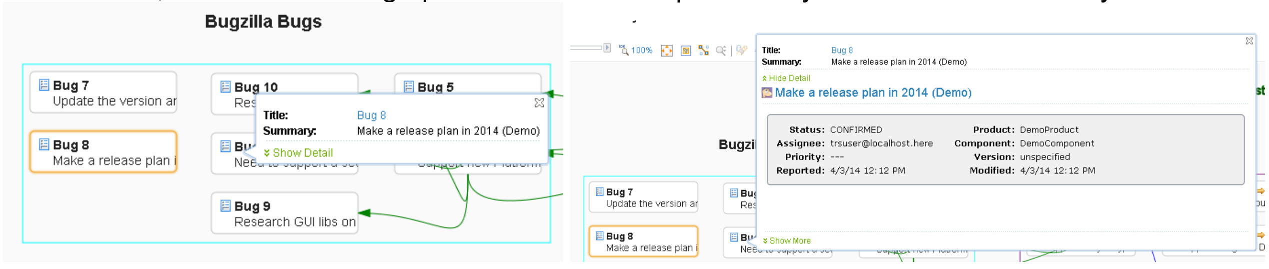


- Three types of link are displayed.
  - Links within Bugs in Bugzilla which are specified in "see also" field. Their colour is green
  - Links within Workitems in RTC which are specified as "Parent/Children" relationship. Their colour is blue.
  - Links between Bugs in Bugzilla and Workitems in RTC. Their colour is green.

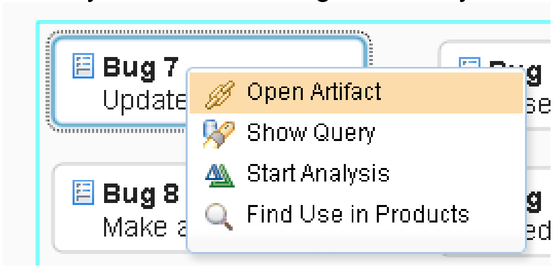


From this view, you will see the traceability within Bugs and Workitems.

__20. This view also supports the preview. When you move the mouse cursor on the rectangle of Bug or Workitem, the small and large preview window is opened as you tried in RELM Query.



__21. If you will click the right button, you will see "**Open Artifact**" menu in its context menu.



If you will click "Open Artifact", the associated tool (in this case, Bugzilla) will be opened to display the selected resource.

__22. You can try other supplied Views if you like.

3.2 Summary

This lab provided requires steps to integrate your TRS provider with RELM. You now have basic understandings of the steps and how to achieve them. Also you have learned how to configure OAuth key between RELM and TRS provider in order for RELM to preview the resources TRS provider provides.

Also you've learned how RELM works with the indexed data by trying its Query and View functions in order to understand how the application can utilize the indexed resource data by LQE and TRS provider.

Appendix A. Troubleshooting

There are always a number of expected things that you might encounter. If there is anything that you may not be able to overcome, please see a proctor or the workshop leader.

In Eclipse, check for compilation errors in the Eclipse project

In Eclipse, Make sure multiple servers aren't running (or right servers are running)

In Eclipse, check Console for Server errors.

Firefox browser has installed an add-in called Firebug that can be used to inspect the web page's DOM and JavaScript.

Ask a proctor for help.

Appendix B. Sources

The sources for this workshop are based on materials that are available in an open source project around OSLC enablement material. Also there are additional materials available to help with consuming OSLC services and more advanced topics are available around authentication and query. See references below for sources:

Eclipse Lyo project http://eclipse.org/Lyo and http://wiki.eclipse.org/Lyo

Everyone is encouraged to join and contribute to these efforts that can greatly help build the ecosystem of OSLC-based tool integration

OSLC Tutorial

Check http://open-services.net for the development of a 2 part tutorial where you can establish your own development environment and follow along with the examples.

Part 1 Consuming OSLC Services

Part 2 Providing OSLC Services

General Resources

As new resources and enablement material are being made available, be sure to check out the OSLC website for these developments at http://open-services.net

Appendix C. Legal

1. Notices

2. About this Content

April 18, 2014

License

NOTES

NOTES

NOTES