Building Services in WSRF

Ben Clifford
GGF Summer School
July 2004
TODOs

- This should be a hidden slide
- Modify RP exercise to use Query not GMRP
- Interop slide
- 2 hours exercise = 60 slides = 15 slides per module
Module 1

- Overview
- WSRF
- Globus Alliance WSRF implementation
Overview

- 4 modules
- Each module has:
  - Slides & talk
  - Hands on
- Covers:
  - WSRF specification
  - Globus Alliance implementation of WSRF
History and Motivation (1)

- Often we think we want standard APIs
  - Eg. MPI
- But on the grid, we actually want standard wire protocols
  - the API can be different on each system
History and Motivation (2)

- Open Grid Services Infrastructure (OGSI)
- GGF standard
- Identified a number of common ‘building blocks’ used in grid protocols
  - Inspecting state, creating and removing state, detecting changes in state, naming state
- Defined standard ways to do these things, based on web services (defined a thing called a Grid Service)
But then...

- Realised that this was useful for web services in general, not just for the grid.
- Moved out of GGF, into OASIS
- Split the single OGSI specification into a number of other specifications called WSRF.
WSRF

WSRF is a framework consisting of a number of specifications.

- **WS-Resource Properties** *
- **WS-Resource Lifetime** *
- WS-Service Groups
- WS-Notification
- WS-BaseFaults
- WS-Renewable References (unpublished)

Other WS specifications such as:
- **WS-Addressing** *

* will be talked about in this tutorial
How WS-RF fits in with other standards, specifications and protocols.

<table>
<thead>
<tr>
<th>Grid stuff</th>
<th>Globus (GRAM, MDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSRF</td>
<td>WSDL, SOAP</td>
</tr>
<tr>
<td>Web services</td>
<td>HTTP, TCP/IP</td>
</tr>
<tr>
<td>Internet protocols</td>
<td></td>
</tr>
</tbody>
</table>
WS-Resources

- Web services often provide access to state
  - Job submissions, databases

- A WS-Resource is standard way of representing that state.

- In this tutorial, we will be using ‘counter’ resources which are simple accumulators.
WS-Resources

- WSRF specifications provide:
  - XML-based Resource Properties
  - Lifetime management (creation/destruction) of resources
  - Servicegroups, which group together WS-Resources
  - Notification
    - (for example of changes in resource properties)
  - Faults
  - Renewable References
Examples of WS-Resources

- Files on a file server
- Rows in a database
- Jobs in a job submission system
- Accounts in a bank
Web service

Web service
Web service with WS-Resource
Web Service with WS-Resources
Web Service with WS-Resources
GT WSRF core

- **Container**
  - Hosts services
  - Built on top of Apache Axis

- **Clients**
  - Interact with services

- **Build tools**
  - For writing new services
  - Based around Apache Ant
Files used in the exercise

- **WSDL and XML Schema:**
  - counter_port_type.wsdl

- **Java**
  - Several Java source files

- **Deployment information**
  - deploy-server.wsdd
  - deploy-jndi-config.xml

- **Build.xml**
  - Tells Ant how to build and deploy the code
Notes on the exercises

- Read notes.txt for information on each exercise.
- Only do one exercise at a time, then wait for next module.
- Each exercise consists of uncommenting code fragments. However, you should **READ AND UNDERSTAND** what you are uncommenting.
- If you are brave, you can make your own extra changes too – but be careful not to break anything!
Exercise 1

- **Exercise**: stand up supplied installation and check it works.
- **Install software**
- **Start the container**
  - this will have a counter service and one counter resource.
- **Interact with the counter resource**

- Do the exercise now.
Exercise 1 overview

One host (your own machine)
One web service running on own machine
One counter resource, which will already exist
Client running on own machine
Exercise 1 overview

globus-start-container

Starts up container, with counter service and a single counter resource.
Exercise 1 overview

show-counter and increment-counter clients interact with the resource through the web service.
increment-counter invokes the add operation in counter_port_type.wsdl
Module 2 – Resource Addressing

- Endpoint References
Why?

- Need some way to refer to web services and WS-Resources from anywhere on the network.
Endpoint References

- WS-Addressing specification
- An **Endpoint Reference** (EPR) points to a web service by including a URL.
Endpoint References

- WS-Addressing specification
- An Endpoint Reference (EPR) points to a web service by including a URL.

```xml
<EPR
    xsi:type="ns2:EndpointReferenceType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing">
    <ns2:Address xsi:type="ns2:Address">
        http://localhost:8080/wsrf/services/CounterService
    </ns2:Address>
</EPR>
```
Endpoint References

- WS-Addressing specification
- An Endpoint Reference (EPR) points to a web service by including a URL.
- EPRs can also contain extra information
- For WSRF, can include `ReferenceProperties` that identify a resource – will see this later on.
Endpoint References

- WS-Addressing specification
- An Endpoint Reference (EPR) points to a web service by including a URL.
- EPRs can also contain extra information
- For WSRF, can include ReferenceProperties that identify a resource – will see this later on.
- Can also contain other information
  - Security
  - Renewable Reference information
Client code fragment

```java
CounterServiceAddressingLocator locator =
    new CounterServiceAddressingLocator();
EndpointReferenceType endpoint;
    endpoint = EPRUtils.loadEPR(args);
CounterPortType counterPort =
    locator.getCounterPortTypePort(endpoint);
counterPort.add(1);
```
Automatically Generated types

```java
CounterServiceAddressingLocator locator =
    new CounterServiceAddressingLocator();
EndpointReferenceType endpoint;
    endpoint = EPRUtils.loadEPR(args);
CounterPortType counterPort =
    locator.getCounterPortTypePort(endpoint);
counterPort.add(1);
```

Highlighted types are generated by the build system automatically, based on XSD and WSDL.
Every WSDL service has a corresponding AddressingLocator Java class automatically generated.

For the CounterService, we get:

- CounterServiceAddressingLocator

An AddressingLocator knows how to take an EPR and return a java stub for the remote service:

- CounterPortType counterPort = locator.getCounterPortTypePort(endpoint);
Every port type has a PortType Java interface automatically generated.

For the counter port type, we have
- CounterPortType

The interface has a method for each operation on the port type:
- counterPort.add(1);
Exercise 2

- Talk to someone else’s service on a different laptop
- Modify clients to read an EPR file
- Should be able to run the clients against any machine in the room.

- Do the exercise now.
Exercise 2 scenario

Two hosts (your own machine and your friend’s machine)
One web service running on friend’s machine
One counter resource on friend’s machine
Client running on your own machine
Exercise 2 scenario

Client can talk to everyone’s servers – so the situation in this room looks more like this.
Module 3 – Resource Properties

- Resources have Resource Properties
- Defined in XML
- Resource Properties document in portType
- Querying Resource Properties
Why?

- Resources represent state
- Often we want to inspect that state

- In this tutorial, we want to know the value stored in each counter
  - show-counter client
XML based

- Each resource has a **Resource Properties document**.
- Defined in XML schema
- Each element in the Resource Properties document is a **Resource Property (RP)**.
Ways to access RPs

- **Pull**
  - Client can query the RP document
    - GetResourceProperty
    - GetMultipleResourceProperties
    - QueryResourceProperties

- **Push**
  - Allows services to send changes in their resources’ RPs to interested parties.
    - WS-Notification
    - Not covered in this tutorial
Pull operations

- **GetResourceProperty**
  - Requests a single resource property by name

- **GetMultipleResourceProperties**
  - Requests several resource properties (from the same resource) by name

- **QueryResourceProperties**
  - More advanced queries against RP document.
  - eg. XPath
Counter example

- The counter service’s Resource Property Document is defined in
  schema/core/samples/counter/counter_port_type.wsdl

- `<xsd:element name="CounterRP">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:Value"
          minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>`
Operation Providers

- WSRF Core allows operations to be implemented by Operation Providers.
- Service writers can include these in WSDD, rather than writing Java code.
- Exercise will involve adding operation provider to support QueryResourceProperties operation.
Exercise 3

- Exercise: add a resource property to the service to give ‘last incremented time’. New client to query this RP.
- Query own counters and query other peoples’ counters.
- Do the exercise now.
Module 4 – Resource Lifetime

- Creating new resources
- Destroying old resources
- Soft-state lifetime management
Why?

- Resources come and go
- For example:
  - jobs in a batch submission system could be represented as resources
  - submitting a new job causes a new resource to be created
  - when the job is completed, the resource goes away
Creating new resources

- Factory pattern
- A web service operation causes a new resource to come into existence.

- For example, in job submission:
  - submit(JobDescription)
Destroying resources

- **Two ways:**
  - Immediate Destruction
  - Scheduled Destruction
Immediate destruction

- Destroy the resource now!

- Destroy operation
Scheduled Destruction

- Scheduled destruction allows soft management of state.
- `TerminationTime RP`
- Keep state alive for as long as we need it, by calling `SetTerminationTime` operation periodically.
Scheduled Destruction

- Remote service is ‘self-cleaning’
  - old unwanted state gets cleaned up automatically if no-one keeps it alive
- Problem: if interested party is disconnected from network for a long time, then it cannot extend lifetime and state may be cleaned up prematurely.
EPRs with ReferenceProperties

- If there are several counters accessible through a service, we need some way to tell them apart when making calls.
- Add ReferenceProperties to EPR with a key that identifies counter.
EPRs with ReferenceProperties

```xml
<EPR
  xsi:type="ns2:EndpointReferenceType"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing">
  <ns2:Address xsi:type="ns2:Address">
    http://localhost:8080/wsrfservices/CounterService
  </ns2:Address>
  <ns2:ReferenceProperties xsi:type="ns2:ReferencePropertiesType">
    <ns3:CounterKey
      xmlns:ns3="http://counter.com">42</ns3:CounterKey>
  </ns2:ReferenceProperties>
</EPR>
```

- Note that the CounterKey field is meaningless to everyone apart from the service.
EPRs with ReferenceProperties

```
<EPR xsi:type="ns2:EndpointReferenceType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing">
  <ns2:Address xsi:type="ns2:Address">
    http://localhost:8080/wsrf/services/CounterService
  </ns2:Address>
  <ns2:ReferenceProperties xsi:type="ns2:ReferencePropertiesType">
    <ns3:CounterKey
      xmlns:ns3="http://counter.com">42</ns3:CounterKey>
  </ns2:ReferenceProperties>
</EPR>
```

- Note that the CounterKey field is meaningless to everyone apart from the service.
EPRs with ReferenceProperties

<EPR xsi:type="ns2:EndpointReferenceType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing">
    <ns2:Address xsi:type="ns2:Address">
        http://localhost:8080/wsrf/services/CounterService
    </ns2:Address>
    <ns2:ReferenceProperties xsi:type="ns2:ReferencePropertiesType">
        <ns3:CounterKey
            xmlns:ns3="http://counter.com">42</ns3:CounterKey>
    </ns2:ReferenceProperties>
</EPR>

- Note that the CounterKey field is meaningless to everyone apart from the service.
Resource Homes

- Resource Homes map from key in EPR to a resource object
- So far, CounterService has used SingletonResourceHome.
  - Always returns the same single resource
  - So CounterService only provides access to one resource
  - No key needed in EPR
- Will now use ResourceHomeImpl
  - Allows creation of new resource objects
  - Maps from key in EPR to resource objects
  - Counter service will provide access to many resource objects
Exercise 4

- **Exercise:** create new counters. Destroy old counters.
- **Two new clients:**
  - create-counter
  - destroy-counter
Exercise 4 scenario

Created new counters
Destroyed existing counters
The rest of WSRF

- WS-Resource Properties
- WS-Resource Lifetime
- WS-Servicegroups
- WS-BaseFaults
- WS-Renewable References
- WS-Notification
WS-ServiceGroups

- Form groups of services and/or resources
- Represent those groups as Resources.
- Registries etc
WS-BaseFaults

- Standard datatype for transmitting webservice faults
  - Originator
  - Timestamp
  - Etc...
WS-Renewable References

- EPRs can become stale
  - Service might move to a different host
- Renewable References provide a way to take a stale reference and try to a fresh one.
WS-Notification

- A group of 3 standards
- Deliver notifications of events
- For example, change in value of a resource property
- Started as one WSRF standard, but split off into three separate standards.
Fin