All credit to...

Borja Sotomayor

http://www.casa-sotomayor.net/gt3-tutorial
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Acronyms

- **OGSA** - Open Grid Services Architecture
- **OGSI** - Open Grid Services Infrastructure
- **GT3** - Globus Toolkit Version 3
- **WSDL** - Web Services Description Language
- **SOAP** - Simple Object Access Protocol
Key Concepts

- **Grid Services** extend web services
- **OGSA** defines architecture (high-level)
- **OGSI** is the specification (the details)
- **GT3** is an implementation of OGSI
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Web Services

- Distributed computing technology which allow us to create client/server applications
- Not intended for human consumption (i.e., not a web site)
Why are web services good?

- Platform-independent
- Language-independent
- Use standard XML languages
- Use HTTP for transmitting messages (proxies and firewalls don’t mess with HTTP traffic)
- Oriented towards loosely coupled systems (e.g., client may not have prior knowledge of web service until it invokes it)
Disadvantages of web services

- Overhead of XML
  - Acceptable for most applications

- Only support basic forms of service invocations -- not very versatile
  - E.g., stateless
  - Grid services help here
Example

1. Where can I find a Web Service that does X?
   - UDDI Registry

2. Server A is capable of doing X!
   - UDDI

3. How exactly should I invoke you?

4. Take a look at this:
   - WSDL

5. Request operation X
   - SOAP

6. Result of operation X
   - SOAP
Web Services Architecture

- Directory service
- Self-describing
- Message Passing
- Protocol
In practice...

- Programmers never write SOAP or WSDL
- A stub is an API that handles marshalling/unmarshalling (SOAP and network communication)
  - Generated automatically via tools
- Basic Steps
  1. Locate a web service through UDDI (returned a URI)
  2. Retrieve WSDL description
  3. Generate the stubs (one time operation)
  4. Application uses stubs
Sample invocation

Globus Toolkit 3 Tutorial
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Grid Services

- Web services +
- Stateful (factories)
- Lifecycle management
- Service data
- Notifications
GT3 Architecture

- Replica Management
- Managed Job Service, Index Service, Reliable File Transfer
- Restrict access
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Tutorial Setup - Step 1

- Log into machine with your account information (supplied with your handouts)
- You should see the following in your home directory
  - jakarta-tomcat-4.1.24 - web services container
  - ogsi-1.0 - GT3 implementation
  - ogsi - symbolic link to ogsi-1.0
  - tutorial - source files for this tutorial
- If not, run
  - /scratch/s1/gt3/setup
  - ~ssmallen/fix_gt3tutorial

or download from http://grid-devel.rocksclusters.org/gt3.html (use gnu tar to unpack)
Tutorial Setup - Step 2

Source the setup file to setup tutorial-related environment variables

- source ${HOME}/tutorial/setup.csh (tcsh)
- . ${HOME}/tutorial/setup.sh (bash)
Writing a Web Math Service: (using GT3 tools)

- Addition
- Subtraction
- Multiplication
- Division
Math Service: Step 1
(Define service interface)

- **PortType** - defines operations
- We can either
  1. Write the WSDL directly (more flexible)
  2. Generate WSDL from an interface language (easier)

```java
package gt3tutorial.core.first.impl;

public interface Math {
    public int add(int a, int b);
    public int subtract(int a, int b);
    public int multiply(int a, int b);
    public float divide(int a, int b);
}
```

`${TUTORIAL_DIR}/gt3tutorial/core/first/impl/Math.java`
Math Service: Step 2 (generate WSDL)

- `cd ${TUTORIAL_DIR}/gt3tutorial/core/first/impl`
- `javac Math.java`
- `cd ${TUTORIAL_DIR}`
- `more MathService.wsdl`
Math Service: Step 3
(decorate WSDL to work with GT3)


- *Opening service file: MathService.wsdl*
Math Service: Step 4
(generating stubs)

- `java org.globus.ogsa.tools.wsd1.GSDL2Java
  MathService.wsdl`

- Creates:
  - `${TUTORIAL_DIR}/gt3tutorial/core/first`
  - `${TUTORIAL_DIR}/org`
Math Service: Step 5
(implementing the service)

- cd $TUTORIAL_DIR/gt3tutorial/core/first/impl

- Package headers

```java
package gt3tutorial.core.first.impl;

import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import gt3tutorial.core.first.Math.MathPortType;
import java.rmi.RemoteException;
```

${TUTORIAL_DIR}/gt3tutorial/core/first/impl/MathImpl.java
Math Service: Step 5
(implementing the service)

- Class definition

```java
${TUTORIAL_DIR}/gt3tutorial/core/fir
st/impl/MathImpl.java

public class MathImpl extends GridServiceImpl
implements MathPortType {
    [constructor, add, divide, subtract,
multiply]
}
```

Contains basic functionality common to every Grid Service; must implement this class

Our autogenerated stub
Math Service: Step 6
(create deployment descriptor)

- Written in WSDD (Web Services Deployment Descriptor)

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig">
  <service name="tutorial/core/first/MathService">
    <parameter name="allowedMethods" value="*"/>
    <parameter name="className" value="gt3tutorial.core.first.impl.MathImpl"/>
  </service>
</deployment>
```

This class implements the service:

Math Service: Step 7
(Compile Grid Service)

- cd $TUTORIAL_DIR
- Stubs
  - javac -sourcepath ./
    gt3tutorial/core/first/Math/*.java
- Implementation
  - javac -sourcepath ./
    gt3tutorial/core/first/impl/*.java
Math Service: Step 8
(create a GAR)

- `jar cvf Math-stub.jar
gt3tutorial/core/first/Math/*.class`
- `jar cvf Math.jar
gt3tutorial/core/first/impl/*.class`
- `mkdir -p gar/schema/tutorial`
- `mv Math.jar Math-stub.jar gar`
- `cp MathService.wsdl gar/schema/tutorial`
- `cp gt3tutorial/core/first/Math.wsdd
gar/server-deploy.wsdd`
- `jar cvf Math.gar -C gar .`

→ Result is Math.gar
Setting up tomcat to run services

To deploy services in tomcat

- Step 1: deploy service in the gt3 container
  (run this command from \${GLOBUS_LOCATION})
  ```
  ant deploy -Dgar.name=\${TUTORIAL_DIR}/Math.gar
  ```
- Step 2: deploy gt3 container services to tomcat
  (run this command from your osgi directory)
  ```
  ant -Dtomcat.dir=\${CATALINA_HOME} deployTomcat
  ```
  (extra) `cp server-config.wsdd \${CATALINA_HOME}/webapps/ogsa/WEB-INF`
- Step 3: Restart tomcat
  ```
  \${CATALINA_HOME}/bin/tomcat -s start
  ```
- Test: `http://<machine name>:<port>/ogsa/services`
Math Client: Step 1

```java
int a = Integer.parseInt(args[1]);
int b = Integer.parseInt(args[2]);
URL GSH = new java.net.URL(args[0]);
// Get a reference to the remote web service
MathServiceLocator mathService = new MathServiceLocator();
MathPortType math = mathService.getMathService(GSH);
// Call remote method 'add'
int sum = math.add(a, b);
// Print result
System.out.println(a + " + " + b + " = " + sum);
```
Math Client: Step 2
(compile)

- cd $TUTORIAL_DIR
- javac -sourcepath ./gt3tutorial/core/first/client/*.java
Math Client: Step 3
(run it!)

- java gt3tutorial.core.first.client.MathClient
  http://localhost:<port>/ogsa/services/tutorial/core/first/MathService 3 2
Lots of steps!
Simplifying with ‘ant’

- ant -Djava.interface=true
  -Dpackage=gt3tutorial.core.first
  -Dinterface.name=Math
  -Dpackage.dir=gt3tutorial/core/first/
  -Dservices.namespace=first.core.gt3tutorial

- Even more so
  - ./tutorial_build.sh
  gt3tutorial/core/first/impl/Math.java
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Grid Service Factories

- **Stateful**: Grid Service remembers information from one call to the next

- **Transient**:
  - One instance for each client
  - Instance will eventually be destroyed

- **Non-transient**
  - One instance shared by several clients
  - Instance isn't destroyed when a client finishes with it (because other clients might need to access it)
Math Service v2: Step 1
(Define service interface)
- Just do add and subtract
- Adding new method getValue

```java
package gt3tutorial.core.factory.impl;
public interface Math {
    public int add(int a, int b);
    public int subtract(int a, int b);
    public int getValue();
}
```
Math Service v2: Step 2 (implementing the service)

```java
private int value = 0;
public MathImpl() {
    super("Simple Math Factory Service");
}
public void add(int a) throws RemoteException {
    value = value + a;
}
public void subtract(int a) throws RemoteException {
    value = value - a;
}
public int getValue() throws RemoteException {
    return value;
}
```

$TUTORIAL_DIR/gt3tutorial/core/factory/impl/MathImpl.java
Math Service v2: Step 3
(create deployment descriptor)

```
${TUTORIAL_DIR}/gt3tutorial/core/factory/Math.wsdd

<service name="tutorial/core/factory/MathFactoryService">
  <parameter name="name" value="MathService Factory"/>
  <parameter name="instance-name"
    value="MathService Instance"/>
  <parameter name="instance-schemaPath"
    value="schema/gt3tutorial.core.factory/Math/MathService.wsdl"/>
  <parameter name="instance-baseClassName"
    value="gt3tutorial.core.factory.impl.MathImpl"/>
  ...
```

description of a Grid Service instance

wsdl path

Base class of the Grid Service instances
Math Service v2: Step 4 (compile and deploy)

- ./tutorial_build.sh
gt3tutorial/core/factory/impl/Math.java
- cd ${OGSA_HOME}
- ${CATALINA_HOME}/bin/tomcat stop
- ant deploy -
  Dgar.name=${TUTORIAL_DIR/build/lib/gt3tutorial.core.factory.Math.gar
- ant -Dtomcat.dir=${CATALINA_HOME} deployTomcat
- ${CATALINA_HOME}/bin/tomcat -s start
Math Client v2: Step 1 (compile)

- cd $TUTORIAL_DIR

- javac -classpath 
  ./build/classes:$CLASSPATH 
  gt3tutorial/core/factory/client/MathClient
  .java
Math Client v2: Step 2 (run it!)

- Create an instance
  - `ogsi-create-service http://localhost:<port>/ogsa/services/tutorial/core/factory/MathFactoryService math`

- Run it (at least 2 times)
  - `java gt3tutorial.core.factory.client.MathClient http://localhost:<port>/ogsa/services/tutorial/core/factory/MathFactoryService/math 5`

- Destroy instance
  - `ogsi-destroy-service http://localhost:<port>/ogsa/services/tutorial/core/factory/MathFactoryService/math`
Math Client v3

1. connects to MathService factory
2. requests that a new instance be created
3. uses it
4. destroys it
Math Client v3

// Get a reference to the MathService Factory
OGSIServiceGridLocator gridLocator;
gridLocator = new OGSIServiceGridLocator();
Factory factory = gridLocator.getFactoryPort(GSH);
GridServiceFactory mathFactory;
mathFactory = new GridServiceFactory(factory);

// Create a new MathService instance and get a reference to its Math PortType.
LocatorType locator = mathFactory.createService();
MathServiceGridLocator mathLocator;
mathLocator = new MathServiceGridLocator();
MathPortType math;
math = mathLocator.getMathService(locator);
Math Client v3

```java
// Get a reference to the GridService
// PortType and destroy the instance
GridService gridService =
    gridLocator.getGridServicePort(locator);
gridService.destroy();
```

$TUTORIAL_DIR/gt3tutorial/core/factory/client/MathClient2.java
Math Client v2: Step 1 (compile)

- cd $TUTORIAL_DIR

- javac
gt3tutorial/core/factory/client/MathClient2.java

- java gt3tutorial.core.factory.client.MathClient2
  http://localhost:<port>/ogsa/services/tutorial/core/factory/MathFactoryService

Answer is 5 because we create new instances each time
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
More about GWSDL

- Writing by hand allows for more expressiveness and more control
Operation providers

- Suppose you want to create a Grid Service by subclassing an existing class

- However
  - In Java, you can only extend one class
  - And every Grid Service has to extend GridServiceImpl

- Use operation providers! (but we won’t go into the details here)
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Lifecycle Management

- Time between instance creation and destruction
- May outlive their clients
- May outlive lifetime of the server (i.e., persistence)
- You can either
  - Define in Grid Service (inherited from GridServiceImpl)
  - Use operation providers
  - Use lifecycle monitor class

{Callback methods differ}
Lifecycle methods from GridServiceImpl

Grid service

- created
- configured
- activated
- deactivated
- destroyed

Callback Methods

- preCreate
- postCreate
- activate
- deactivated
- preDestroy

Great stuff happens
Lifecycle Monitor

- Define and then reuse with multiple Grid Services

- Implements ServiceLifecycleMonitor, interface with callback methods
  - Methods called by container at specific points in the instance’s lifetime
  - You don’t call them directly in your code
Lifecycle Monitor Methods

Grid service

1. created
2. configured
3. activated
4. deactivated
5. destroyed

Callback Methods

- create
  - preSerializationCall
  - preCall
  - Function happens
  - postCall
  - postSerializationCall

- destroy

Great stuff happens
Math Client Lifecycle Monitor:
Step 1 (prerequisites)

- Ensure logging is turned on

```properties
$GLOBUS_LOCATION/ogsilogging.properties

gt3tutorial.core.lifecycle.impl.MathLifecycleMonitor=console,info
```

- Add lifecycle monitor to deployment descriptor

```xml
${TUTORIAL_DIR}/gt3tutorial/core/lifecycle/Math.wsdd

<parameter name="lifecycleMonitorClass" value="gt3tutorial.core.lifecycle.impl.MathLifecycleMonitor"/>
```
Lifecycle Monitor code

```
public class MathLifecycleMonitor implements ServiceLifecycleMonitor{
    ...
    public void create(GridContext context) throws GridServiceException {
        logger.info("Instance is going to be created (create)" newX);
    }
    public void destroy(GridContext context) throws GridServiceException {
        logger.info("Instance is going to be destroyed (destroy)" newX);
    }
    ...
```

$TUTORIAL_DIR/gt3tutorial/core/lifecycle/impl/MathLifecycleMonitor.java
But first...

- Since we’re moving the logging section to the end and the other steps depend on the logging libraries, run this:

  - 
  ```bash
  ./tutorial_build.sh
gt3tutorial/core/logging/schema/Math.g wsdl
  ```
Math Client Lifecycle Monitor:
Step 2 (compile, deploy, and run!)

- ./tutorial_build.sh
  gt3tutorial/core/lifecycle/schema/Math.gwsdl
- cd ${GLOBUS_LOCATION}
- ant deploy -
  Dgar.name=${TUTORIAL_DIR}/build/lib/gt3tutorial.core.lifecycle.Math.gar
- globus-start-container -p XXXX (log will be on the window)
- Start another window
- cd ${TUTORIAL_DIR}
- javac
  gt3tutorial/core/lifecycle/client/MathClient.java
- java gt3tutorial.core.lifecycle.client.MathClient
  http://localhost:<port>/ogsa/services/tutorial/core/lifecycle/MathFactoryService
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Service Data

- Used to advertise what kind of services it offers
- Replaces GRIS
- Index service - query service data and chooses services
Service Data (cont.)

- Service can have any number of service data elements
Service Data Elements

- Must have a unique name (within a Grid Service)
- Can be complex types
- Implemented as a Java Bean
  - java object with a null constructor and a standard convention for access to object attributes
- Generated automatically from a service data description
XML Schema --> Java Bean

```xml
<schema>
  <complexType name=MathDataType>
    <attribute name="speed" type="int"/>
    <attribute name="cost" type="float"/>
    <attribute name="statistic" type="boolean"/>
  </complexType>
</schema>

class MathDataType {
  public int getSpeed ();
  public void setSpeed( int speed);

  public float getCost();
  public void setCost( float cost);

  public boolean getStatistic();
  public void setStatistic( boolean statistic)
}
```
Grid Service w/ Service Data
Step 1: Define Service Data

```
$TUTORIAL_DIR gt3tutorial/core/servicedata/schema/MathDataType.xsd

<complexType name="MathDataType">
    <sequence>
        <element name="lastOp" type="string"/>
        <element name="numOps" type="int"/>
    </sequence>
</complexType>
```
Math Service w/ Service Data

Step 2: Adding to GWSDL

- Specify properties of Service Data in GWSDL
  - Add new namespaces
    - MathDataType
    - Service-data related tags
  - Import MathDataType.xsd
  - Specify properties of SDE
    - minOccurs/maxOccurs - min/max # of SDEs
    - Modifiable - can/can’t be changed by client
    - Nillable - can/can’t be null
    - Mutability - static, constant, extendable, mutable
Math Service w/ Service Data

Step 3: Modifying implementation

- Add SDEs as two new private attributes

```java
private ServiceData serviceData;  //SDE
private MathDataType mathData;  //SDE value
```

- Creation of SDE takes place in postCreate

```java
serviceData = this.getServiceDataSet().create("MathData");
mathData = new MathDataType();
serviceData.setValue(mathData);
mathData.setLastOp("NONE");mathData.setNumOps(0);
this.getServiceDataSet().add(serviceData);
```
Math Service w/ Service Data

Step 3: Modifying implementation

- Every math method must modify SDE
  - mathData.setLastOp( "method" )
  - Increment number of ops by 1
Math Service w/ Service Data

Step 4: compile and deploy

1. $TUTORIAL_DIR> ./tutorial_build.sh
gt3tutorial/core/servicedata/schema/Math.gwsdl
2. cd ${GLOBUS_LOCATION}
3. ${CATALINA_HOME}/bin/tomcat stop
4. ant deploy -
   Dgar.name=$TUTORIAL_DIR/build/lib/gt3tutorial.core.servicedata.Math.gar
5. ant -Dtomcat.dir=${CATALINA_HOME} deployTomcat
6. ${CATALINA_HOME}/bin/tomcat -s start
7..osgi-create-service
   http://localhost:<port>/ogsa/services/tutorial/core/servicedata/MathFactoryService math
Math Client w/ Service Data

Step 1: modify implementation

- Get a reference to the GridService PortType

  ```java
  OGSIServiceGridLocator locator = new OGSIServiceGridLocator();
  GridService gridService = locator.getGridServicePort(GSH);
  ```

- Get Service Data Element "MathData"

  ```java
  ExtensibilityType extensibility =
  gridService.findServiceData(QueryHelper.getNamesQuery("MathData"));
  ServiceDataValuesType serviceData =
  AnyHelper.getAsServiceDataValues(extensibility);
  MathDataType mathData =
  (MathDataType) AnyHelper.getAsSingleObject(serviceData, MathDataType.class);
  ```
Math Client w/ Service Data

Step 1: modify implementation

- Write service data

```
$TUTORIAL__DIR/gt3tutorial/core/servicedata/client/MathClient.java

System.out.println("Previous operation: " +
        mathData.getLastOp());
System.out.println("# of operations: " +
        mathData.getNumOps());
```
Math Client w/ Service Data

Step 2: Compile and run

- javac
gt3tutorial/core/servicedata/client/MathClient.java

- Run it a couple times
  - java
gt3tutorial.core.servicedata.client.MathClient
  http://localhost:<port>/ogsa/services/tutorial/core/servicedata/MathFactoryService/math 5 5
Standard SDEs

- **gridServiceHandle** - contains the GridService's GSHs
- **factoryLocator** - the locator for the factory which created this Grid Service. If the Grid Service was not created by a factory, the value of this SDE will be null.
- **terminationTime** - information about the termination time of the Grid Service.
- **serviceDataNames** - names of all the SDEs in the Grid Service.
- **interfaces** - names of all the interfaces (PortTypes) implemented by this Grid Service.
PrintGridServiceData

- **compile**
  - `javac`  
    `gt3tutorial/core/servicedata/client/PrintGridServiceData.java`

- **Run on factory**
  - `java`  
    `gt3tutorial.core.servicedata.client.PrintGridServiceData`  
    `http://localhost:<port>/ogsa/services/tutorial/core/servicedata/MathFactoryService`

- **Run on instance**
  - `java`  
    `gt3tutorial.core.servicedata.client.PrintGridServiceData`  
    `http://localhost:<port>/ogsa/services/tutorial/core/servicedata/MathFactoryService/math`
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
    A. Factories
    B. GSWDL and Operation Providers (brief)
    C. Lifecycle management
    D. Service data
    E. Notifications

IV. Extras
    A. Logging
Notifications

- Model 1: Polling
  - Client periodically polls the server and the server responds when it has information of an event.
  - Not efficient

- Model 2: Subscribe
  - Client subscribes to the server for messages and get notified.
  - Much better
Notification (..contd)

- Observer is the client and Observable is the server
- Pull: Observer makes an extra call to retrieve e-mails
- Push: Observable will send the e-mails without the observer asking for it.
Notifications in GT3

- **addListener**: This call subscribes the calling client to a particular SDE (which is specified in the call).
- **notifyChange**: Whenever a change happens, the MathService instance will ask the SDE to notify its subscribers.
- **deliverNotification**: The SDE notifies the subscribers that a change had happened.
Notification: A ‘pull’ Service

- Inform clients of the internal values at all times
- Grid Service extends Notification Source

```xml
<gwSDL:portType name="MathPortType"
extends="ogsi:GridService
ogsi:NotificationSource ">
<!-- Operations -->
</gwSDL:portType>
```
Notification

```java
Public class MathProvider implements OperationProvider {

    private ServiceData servicedata;

    public void postCreate(GridContext context) throws GridServiceException {
        serviceData = base.getServiceDataSet().create("DummySDE");
        base.getServiceDataSet().add(serviceData);
    }

    public void add(int a) {
        value = value + a;
        serviceData.notifyChange();
    }
```
Notification

- Namespace Mappings
- Compile, Deploy & Create

- ./tutorial_build.sh
gt3tutorial/core/notificationsPull/schema/Math.gwsdl
- ${CATALINA_HOME}/bin/tomcat stop
- cd ${GLOBUS_LOCATION}
- ant deploy -
  Dgar.name=${TUTORIAL_DIR}/build/lib/gt3tutorial.core.
  notificationsPull.Math.gar
- ant -Dtomcat.dir=${CATALINA_HOME} deployTomcat
- ${CATALINA_HOME}/bin/tomcat -s start
- ogsi-create-service
  http://localhost:<port>/ogsa/services/tutorial/core/n
  otificationsPull/MathFactoryService math
Notification: Pull Client

Listener Client

Public class MathListener extends ServicePropertiesImpl
    implements NotificationSinkCallback
{
    public static void main()
    {
        public MathListener(HandleType GSH) throws exception
        {
            ......
            // Start listening to the MathService
            NotificationSinkManager notifManager =
                NotificationSinkManager.getManager();
            notifManager.startListening(NotificationSinkManager.MAIN_THREAD);
            String sink = notifManager.addListener("DummySDE", null, GSH, this);
            System.out.println("Listening...");
Notifications

//stop listening
notifManager.removeListener(sink);
notifManager.stopListening();
}

public void deliverNotification(ExtensibilityType any)
    throws RemoteException
{
    try{ // Value has changed, so we need to get the new value
        int value = math.getValue();
        System.out.println("Value has changed: " + value);
    }
    catch(Exception exc){
        System.out.println("ERROR!");
        exc.printStackTrace();
    }
}
Notification

- Compile and Run Listener Client
  - `javac`
    
    `gt3tutorial/core/notificationsPull/client/MathListener.java`

  - `cd $GLOBUS_LOCATION`

Notification

- Compile and Run Adder Client (in another window)
  - `javac`
    
    `javac
    gt3tutorial/core/notificationsPull/client/MathAdder.java`
  
  - `java`
    
    `java
    gt3tutorial.core.notificationsPull.client.MathAdder
    http://localhost:<port>/ogsa/services/tutorial/core/notificationsPull/MathFactoryService/math`
Notification: A ‘push’ Service

- Internal value will be sent along with the notification
- The SDE contains value of additions and subtractions & other information

```xml
<complexType name="MathDataType">
  <sequence>
    <element name="value" type="int"/>
    <element name="lastOp" type="string"/>
    <element name="numOps" type="int"/>
  </sequence>
</complexType>
```
Notification

Service Interface

```java
public interface Math
{
    public void add(int a);
    public void subtract(int a);
}
```

Add Namespace of SDD, extend NotificationSource

```xml
xmlns:data="http://www.gt3tutorial.org/namespaces/0.2/core
/notificationsPush/MathData"

<gwsdl:portType name="MathPortType"
    extends="ogsi:GridService ogsi:NotificationSource">
    <!-- <operation>s -->
    <sd:serviceData name="MathData"
        type="data:MathDataType" minOccurs="1"
        maxOccurs="1" mutability="mutable"
        modifiable="false" nillable="false">
    </gwsdl:portType>
```
Notification

public class MathProvider implements OperationProvider,
    GridServiceCallback
{
    private MathDataType mathData; ..........

    public void postCreate(GridContext context) throws
        GridServiceException
    {
        serviceData = base.getServiceDataSet().create("MathData");
        mathData = new MathDataType();
        serviceData.setValue(mathData);
        mathData.setValue(0);
        mathData.setLastOp("NONE");
        mathData.setNumOps(0); .......... }

    private void incrementOps()
    {
        int numOps = mathData.getNumOps();
        mathData.setNumOps(numOps + 1); }

    public void add(int a) throws RemoteException
    {
        mathData.setLastOp("Addition");
        incrementOps();
        mathData.setValue(mathData.getValue() + a);
        serviceData.notifyChange(); }
}
Notification

Compile and Deploy

- `./tutorial_build.sh`  
  `gt3tutorial/core/notificationsPush/schema/Math.gws dl`
- `${CATALINA_HOME}/bin/tomcat` stop
- `cd `${GLOBUS_LOCATION}``
- `ant deploy -Dgar.name=$TUTORIAL_DIR/build/lib/gt3tutorial.core.notificationsPush.Math.gar`
- `ant -Dtomcat.dir=${CATALINA_HOME} deployTomcat`
- `${CATALINA_HOME}/bin/tomcat -s start`
Notification: push Client

public void deliverNotification(ExtensibilityType any) throws RemoteException {
    try {
        // Service Data has changed. Show new data.
        ServiceDataValuesType serviceData = AnyHelper.getAsServiceDataValues(any);
        MathDataType mathData = (MathDataType) AnyHelper.getAsSingleObject(serviceData, MathDataType.class);

        // Write service data
        System.out.println("Current value: "+ mathData.getValue());
        System.out.println("Previous operation: "+ mathData.getLastOp());
        System.out.println("# of operations: "+ mathData.getNumOps());
    }
}
Notification: push Client compile

- `javac`
  `gt3tutorial/core/notificationsPush/client/MathListener.java`

- `cd $GLOBUS_LOCATION`

Notification

- Compile and Run Adder Client
  - `javac`
    
    `gt3tutorial/core/notificationsPush/client/MathAdder.java`
  
  - `java`
    
    `gt3tutorial.core.notificationsPush.client.MathAdder`
    
Outline

I. Introduction
   A. Web Services
   B. Grid Services

II. Building a web service

III. Building a Grid service
   A. Factories
   B. GSWDL and Operation Providers (brief)
   C. Lifecycle management
   D. Service data
   E. Notifications

IV. Extras
   A. Logging
Logging

- Write a log of interesting events to console or file (warnings, errors, debug info, etc.)
- Based on Apache Jakarta Commons Logging
- Logging messages with varying levels of ‘severity’ to allow filtering.
- Levels include (Debug, Trace, Info, Warn, Error, Fatal)
Logging

- In the class implementing the methods add:

```java
//...
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
Public class MathImpl extends ......
{
    //create this class’s logger
    static Log logger =
    LogFactory.getLog(MathImpl.class.getName());
    public void add(int a) throws ...
    {
        logger.info("addition invoked with
                    parameter a=“ + String.valueOf(a));
        if (a==0)
        logger.warn("Adding zero doesn’t modify the
                    internal value)
        value = value + a;
    }
```
Logging

To write log to the console or file

- Add following line to $globus_dir/ogsiloggIng.properties

```
gt3tutorial.core.logging.impl.MathImpl=console,info
```

Tells logger where to write the messages

Specifies how to filter messages (according to their level)
Additional information

- Security is covered in Sotomayor tutorial
  - http://www.casa-sotomayor.net/gt3-tutorial

- GT3 website
  - http://www.globus.org/gt3/
Globus Toolkit 4 / Web Services
Resource Framework

- Integrated with web services
- Syntax is different; concepts are the same
- Switch should be straightforward
Thanks for your participation!

Shava Smallen - ssmallen@sdsc.edu
Sandeep Chandra - chandras@sdsc.edu
Catherine Olschanowsky - cmills@sdsc.edu

Grid Development Group
San Diego Supercomputer Center