



## Sistemas Grid Basados en GT3



### Módulo 2 GT3 Core

*Borja Sotomayor*  
*4 de marzo de 2004*

Sistemas Grid Basados en GT3



## Introducción

- ▶ En este módulo aprenderemos a programar Grid Services sencillos utilizando únicamente el núcleo de GT3 (la parte correspondiente a OGSI)
- ▶ Para ello utilizaremos un servicio sencillo llamado MathService.

Sistemas Grid Basados en GT3





## Índice

- ▶ Instalación de GT3
- ▶ Un Servicio Sencillo
- ▶ Enfoques de Implementación
- ▶ Service Data
- ▶ Notificaciones
- ▶ Servicios Transientes



## Índice

- ▶ Instalación de GT3
- ▶ Un Servicio Sencillo
- ▶ Enfoques de Implementación
- ▶ Service Data
- ▶ Notificaciones
- ▶ Servicios Transientes



## Requisitos Software

- ▶ Software que debemos tener instalado
  - ▶ JDK 1.4 o superior
    - ▶ <http://java.sun.com/j2se>
    - ▶ También es posible utilizar JDK 1.3.1 o superior, pero requiere una instalación especial.
  - ▶ Jakarta Ant 1.5 o superior
    - ▶ <http://jakarta.apache.org/ant>
  - ▶ Junit 3.8.1 (Opcional)
    - ▶ <http://www.junit.org/>
- ▶ Ya está instalado en las máquinas del curso (excepto Junit)



## Modalidades de Instalación (I)

- ▶ Existen distintas modalidades de instalación:
  - ▶ Completa: Incluye núcleo, servicios de alto nivel, y GT2.4
    - ▶ Source
    - ▶ Binary
  - ▶ Núcleo
    - ▶ Source
    - ▶ **Binary**



## Modalidades de Instalación (II)

- ▶ Nosotros utilizaremos la distribución binaria del núcleo de GT3.2beta.
  - ▶ `ogsa-3.2.tar.gz`
  - ▶ “Descomprimir y usar”
  - ▶ Las otras modalidades requieren un proceso de compilación bastante largo.



## Instalación (I)

- ▶ Es posible instalar GT3 en sistemas sin acceso root, pero vamos a suponer que sí lo tenemos.
- ▶ Como root:
  - ▶ Copiar `ogsa-3.2.tar.gz` a `/usr/local`
  - ▶ Descomprimir
    - ▶ `tar xvzf ogsa-3.2.tar.gz`
  - ▶ Por comodidad creamos un enlace simbólico:
    - ▶ `ln -s ogsa-3.2.tar.gz gt3`
  - ▶ Crear usuario globus
    - ▶ `useradd -d /usr/local/gt3 -s /bin/bash globus3`
    - ▶ `passwd globus`
  - ▶ Dar permisos en la carpeta de GT3
    - ▶ `chown -R globus /usr/local/gt3`



## Instalación (II)

- ▶ El usuario globus va a ser el administrador de GT3 en nuestro sistema.
- ▶ Para cada cuenta desde la que queramos tener acceso a GT3 debemos añadir lo siguiente a ~/.bash\_profile (o a /etc/profile para *todos* los usuarios)

```
CLASSPATH="."
JAVA_HOME="directorio de java"
ANT_HOME="directorio de ant"
GLOBUS_LOCATION="directorio de GT3"
PATH="$PATH:$JAVA_HOME/bin:$ANT_HOME/bin:$GLOBUS_LOCATION/bin"
export PATH CLASSPATH JAVA_HOME ANT_HOME GLOBUS_LOCATION
```



## Instalación (III)

- ▶ Para finalizar la instalación (y además asegurarnos de que todo está correctamente instalado), desde \$GLOBUS\_LOCATION:

```
ant setup
```



## Instalación (IV)

### ▶ Guías de instalación más completas:

- ▶ `$GLOBUS_LOCATION/docs/admin/index.html`
- ▶ “GT3 Quick Start” (Redpaper de IBM, muy recomendable)  
<http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp3697.html>  
(o sencillamente “GT3 Quick Start Redpaper” en Google)
- ▶ From Zero to GT3  
<http://www-pnp.physics.ox.ac.uk/~stokes/twiki/bin/view/DIRAC/GT3Express>



## Ejemplos

- ▶ Los ejemplos del curso se encuentran en el fichero `ejemplos.tar.gz`
- ▶ Podemos descomprimir este fichero en cualquier directorio de nuestra cuenta de usuario (*no* en la cuenta `globus`)
- ▶ Nos referiremos a este directorio como `$TUTORIAL_DIR`



## Índice

- ▶ Instalación de GT3
- ▶ **Un Servicio Sencillo**
- ▶ Enfoques de Implementación
- ▶ Service Data
- ▶ Notificaciones
- ▶ Servicios Transientes



## MathService (I)

- ▶ Vamos a programar y desplegar un servicio sencillo llamado MathService.

<b>MathService</b>
-int value
+void add(int a) +void subtract(int a) +int getValue(void)



## MathService (II)

- ▶ Es un servicio con estado.
  - ▶ Veremos como el valor del atributo *value* se conserva de una invocación a otra.
- ▶ Es un ejemplo trivial. En el siguiente módulo veremos un ejemplo más 'real' (FileShare)



## Metodología de Desarrollo

- ▶ Metodología de desarrollo
  - ▶ Definir el interfaz del servicio – **GWSDL**
  - ▶ Implementar el servicio – **Java**
  - ▶ Definir detalles del despliegue – **WSDD**
    - ▶ ¿Qué dirección tendrá el servicio?
    - ▶ ¿En qué fichero se encuentra la implementación?
    - ▶ etc.
  - ▶ Compilar todo, generando un fichero GAR – **Ant**
    - ▶ Compilar Java, generar stubs
    - ▶ GAR: Grid ARchive. Incluye ficheros WSDL y Java compilados
  - ▶ Desplegar servicio (fichero GAR) en entorno de ejecución – **Ant**



## Definir el interfaz (I)

- ▶ Debemos definir el interfaz en GWSDL:
  - ▶ El interfaz es todo lo que puede ser accedido remotamente.
    - ▶ add, subtract, y getValue
    - ▶ *no* el atributo value (es un atributo privado)
  - ▶ \$TUTORIAL\_DIR/schema/progtutorial/MathService/Math.gwsdl
  - ▶ Podríamos partir de un interfaz Java, pero está desaconsejado.
    - ▶ La conversión Java a WSDL es propensa a errores.
    - ▶ Con (G)WSDL tenemos más control. Merece la pena aprenderlo. No es complicado.
    - ▶ Pequeña guía en el Tutorial de Programación:  
<http://www.casa-sotomayor.net/gt3-tutorial/howto/gwsdl.html>



## Definir el interfaz (II)

```
<gwsdl:portType name="MathPortType" extends="ogsi:GridService">
  <operation name="add">
    <input message="tns:AddInputMessage" />
    <output message="tns:AddOutputMessage" />
    <fault name="Fault" message="ogsi:FaultMessage" />
  </operation>
  <operation name="subtract">
    <input message="tns:SubtractInputMessage" />
    <output message="tns:SubtractOutputMessage" />
    <fault name="Fault" message="ogsi:FaultMessage" />
  </operation>
  <operation name="getValue">
    <input message="tns:GetValueInputMessage" />
    <output message="tns:GetValueOutputMessage" />
    <fault name="Fault" message="ogsi:FaultMessage" />
  </operation>
</gwsdl:portType>
```



## Implementar el servicio (I)

- ▶ Hay dos enfoques de implementación:
  - ▶ Por herencia
  - ▶ Por delegación
- ▶ Vamos a realizar la implementación por herencia. Más adelante veremos la implementación por delegación.
- ▶ Por herencia: Creamos una clase que hereda de `GridServiceImpl` (clase de GT3) y en la que implementamos los métodos definidos en el interfaz.
- ▶ Podemos incluir todos los métodos y atributos privados que queramos.

\$TUTORIAL\_DIR/org/globus/progtutorial/services/core/first/impl/MathImpl.java

Sistemas Grid Basados en GT3



## Implementar el servicio (II)

```
package org.globus.progtutorial.services.core.first.impl;
```

```
import org.globus.ogsa.impl.ogsi.GridServiceImpl;  
import org.globus.progtutorial.stubs.MathService.MathPortType;  
import java.rmi.RemoteException;
```

```
public class MathImpl extends GridServiceImpl implements MathPortType  
{  
    private int value = 0;  
  
    public MathImpl()  
    {  
        super("Simple MathService");  
    }  
    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;  
    }  
}
```

Interfaz *stub* que se genera a partir del GWSDL

Todos los métodos remotos deben lanzar `RemoteException`

Sistemas Grid Basados en GT3



## Definir despliegue (I)

- ▶ Los detalles de despliegue se definen en un fichero WSDD
  - ▶ Web Services Deployment Descriptor
  - ▶ Formato XML utilizado por Apache Axis



## Definir despliegue (II)

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="progtutorial/core/first/MathService" provider="Handler" style="wrapped">
    <parameter name="name" value="MathService"/>
    <parameter name="baseClassName"
      value="org.globus.progtutorial.services.core.first.impl.MathImpl"/>
    <parameter name="className"
      value="org.globus.progtutorial.stubs.MathService.MathPortType"/>
    <parameter name="schemaPath"
      value="schema/progtutorial/MathService/Math_service.wsdl"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*/>
    <parameter name="persistent" value="true"/>
    <parameter name="handlerClass"
      value="org.globus.ogsa.handlers.RPCURIPProvider"/>
  </service>
</deployment>
```



## Definir despliegue (III)

- ▶ `<deployment>` puede contener 1..N `<service>`
  - ▶ El fichero, de hecho, contiene dos `<service>`. El segundo es un ejemplo que veremos más adelante.
- ▶ Cada `<service>` contiene 0..N `<parameter>`
  - ▶ Los parametros que especifican los detalles del despliegue.
- ▶ `<service name="...">`
  - ▶ Dirección que tendrá el servicio grid.  
`http://localhost/ogsa/services/ + progutorial/core/first/MathService`
- ▶ Parametros
  - ▶ name: Nombre del servicio
  - ▶ baseClass: Clase base del servicio (la implementación)
  - ▶ class: Clase del servicio (el interfaz *stub* del portType)
  - ▶ schemaPath: Interfaz WSDL del servicio (generado por el proceso de compilación; nosotros escribimos un GWSDL)



## Compilar todo (I)

- ▶ La compilación se realiza con Ant
  - ▶ Ant es una herramienta similar al *make* tradicional de UNIX, pero orientado a Java.
  - ▶ El fichero *makefile* está compuesto en XML (en Ant se le suele llamar *buildfile*, o sencillamente fichero Ant)
  - ▶ GT3 incluye ficheros build para las tareas más habituales:
    - ▶ Generar los stubs
    - ▶ Compilar la implementación
    - ▶ etc.



## Compilar todo (II)

- ▶ En proyectos complejos seguramente tendremos que escribir un fichero Ant a medida.
- ▶ En el curso utilizaremos un fichero Ant que sirve para ejemplos sencillos.
  - ▶ No lo utilizaremos directamente, sino a través de un script que facilita su uso.



## Compilar todo (III)

- ▶ Cómo utilizar el script:
  - ▶ Desde `$TUTORIAL_DIR` ejecutar:  
`./tutorial_build.sh <directorio base> <fichero GWSDL>`  
El "directorio base" es la raíz del paquete Java donde ponemos el servicio. Debe contener:
    - ▶ Fichero `server-deploy.wsdd` (descriptor de despliegue)
    - ▶ Directorio `impl` con ficheros Java (implementación)
    - ▶ También puede contener un directorio `config` que veremos más adelante
  - ▶ Para el ejemplo actual:  
`./tutorial_build.sh  
org/globus/progtutorial/services/core/first/  
schema/progtutorial/MathService/Math.gwsdl`



## Compilar todo (IV)

- ▶ El script de compilado genera un fichero GAR en \$TUTORIAL\_DIR/build/lib
- ▶ El fichero GAR contiene:
  - ▶ El descriptor de despliegue
  - ▶ El interfaz convertido a WSDL
  - ▶ La implementación compilada
  - ▶ Los stubs compilados



## Desplegar servicio

- ▶ Para desplegar el GAR en el contenedor, debemos ejecutar lo siguiente desde la cuenta de usuario globus:

```
ant deploy
-Dgar.name=$TUTORIAL_DIR/build/lib/<fichero gar>
org.globus.progtutorial.services.core.first.Math.gar
```



## Iniciar contenedor

- ▶ Para iniciar el contenedor, debemos ejecutar lo siguiente con el usuario globus:

```
globus-start-container
```

- ▶ Al iniciarse, el contenedor muestra una lista de todos los servicios desplegados. Debería aparecer el nuestro:

```
http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/MathService
```



## Probar cliente (I)

- ▶ Para comprobar que el servicio está correctamente instalado, y que funciona bien, vamos a utilizar un sencillo cliente.

```
$TUTORIAL_DIR/org/globus/progtutorial/clients/MathService/Client.java
```



## Probar cliente (II)

Con tan solo dos lineas obtenemos una referencia al interfaz remoto

```
// Get a reference to the MathService instance
MathServiceGridLocator mathServiceLocator = new MathServiceGridLocator();
MathPortType math = mathServiceLocator.getMathServicePort(GSH);

// Call remote method 'add'
math.add(a);
```

Trabajamos con él como si fuese un objeto local.



## Probar cliente (III)

- ▶ Incluir librerías de Globus en CLASSPATH:  
`source $GLOBUS_LOCATION/etc/globus-devel-env.sh`
- ▶ Compilar cliente:  
`javac -classpath ./build/classes/:$CLASSPATH  
org/globus/progtutorial/clients/MathService/Client.java`
- ▶ Ejecutar cliente:  
`java -classpath ./build/classes/:$CLASSPATH  
org/globus/progtutorial/clients/MathService/Client  
http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/MathService  
5`
- ▶ Resultado esperado:  
Added 5  
Current value: 5



## Índice

- ▶ Instalación de GT3
- ▶ Un Servicio Sencillo
- ▶ **Enfoques de Implementación**
- ▶ Service Data
- ▶ Notificaciones
- ▶ Servicios Transientes

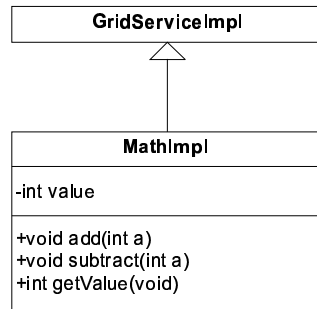


## Implementación por Delegación (I)

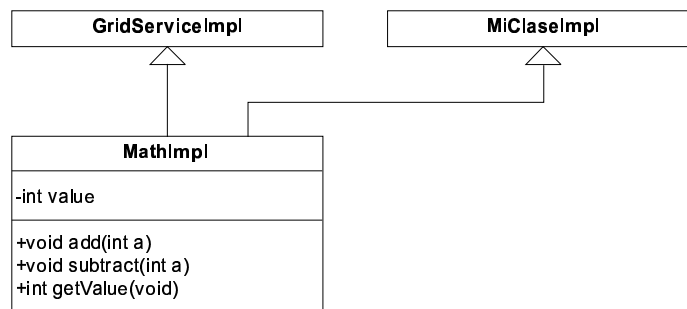
- ▶ En el ejemplo anterior nuestra clase MathImpl hereda de GridServiceImpl.
  - ▶ GridServiceImpl es una clase incluida con GT3 en la que ya está implementada toda la funcionalidad *básica* exigida por OGSI en los servicios grid.
- ▶ Este enfoque tiene inconvenientes



## Implementación por Delegación (II)

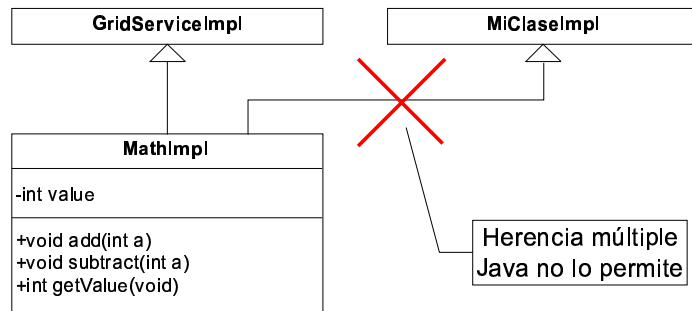


## Implementación por Delegación (III)





## Implementación por Delegación (IV)

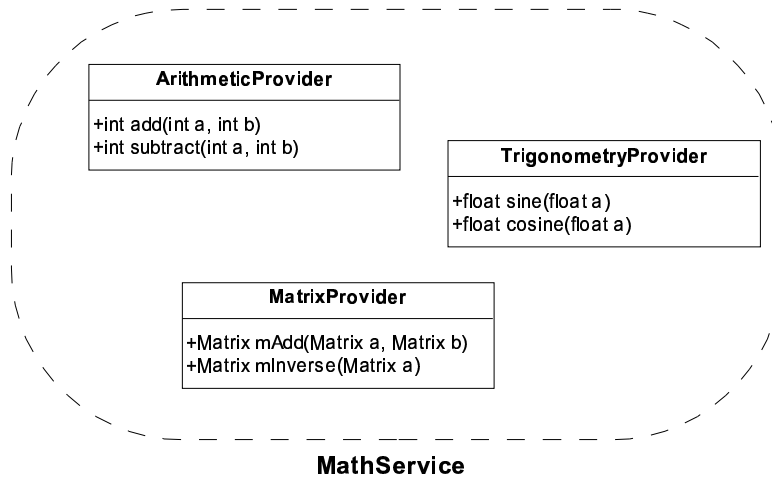


## Implementación por Delegación (V)

- ▶ Solución: Implementación por delegación.
  - ▶ Tenemos un conjunto de clases llamadas *operation providers* que conforman la implementación del servicio.
  - ▶ No tienen que heredar de ninguna clase, sólo implementar el interfaz `OperationProvider`
  - ▶ En realidad, la clase `GridServiceImpl` sigue estando presente como base del servicio, a pesar de no heredar explícitamente de ella.



## Implementación por Delegación (VI)



## Implementación por Delegación (VII)

- ▶ **Ventajas**
  - ▶ Los operation providers pueden heredar de clases propias.
  - ▶ Podemos reutilizar operation providers entre varios servicios.
- ▶ **Desventajas**
  - ▶ Hay que escribir un poco más de código.
- ▶ ¡Ojo! En GT3 no son enfoques mutuamente exclusivos. Puedo implementar un servicio heredando de `GridServiceImpl` y, además, añadir funcionalidad extra con un operation provider.



## Ejemplo

- ▶ Como ejemplo, vamos a reimplementar el ejemplo anterior pero utilizando operation providers.
- ▶ No cambiamos el interfaz, por lo que:
  - ▶ Reutilizamos el GWSDL
  - ▶ Reutilizamos el cliente
- ▶ Cambios:
  - ▶ Implementación
  - ▶ WSDD



## Implementación

- ▶ \$TUTORIAL\_DIR/org/globus/progtutorial/services/core/providers/impl/MathProvider.java

```
package org.globus.progtutorial.services.core.providers.impl;

// imports

public class MathProvider implements OperationProvider
{
    // Operation provider properties
    private static final QName[] operations = new QName[]{new QName("", "*")};
    private GridServiceBase base;

    // Operation Provider methods
    public void initialize(GridServiceBase base) throws GridServiceException
    {
        this.base = base;
    }

    public QName[] getOperations()
    {
        return operations;
    }
    // Implementacion de metodos remotos. Exactamente igual que en ejemplo anterior.
}
```



## Descriptor de Despliegue (I)

- ▶ En el descriptor de despliegue (WSDD) hay un nuevo parámetro:
  - ▶ `operationProviders`: clases que van a aportar funcionalidad a nuestro servicio.



## Descriptor de Despliegue (II)

▶ \$TUTORIAL\_DIR/org/globus/progtutorial/services/core/providers/server-config.wsdd

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="progtutorial/core/providers/MathService" provider="Handler" style="wrapped">
    <parameter name="name" value="MathService (with Operation Provider)"/>
    <parameter name="className"
      value="org.globus.progtutorial.stubs.MathService.MathPortType"/>
    <parameter name="baseClassName"
      value="org.globus.ogsa.impl.ogsi.GridServiceImpl"/>
    <parameter name="schemaPath"
      value="schema/progtutorial/MathService/Math_service.wsdl"/>
    <parameter name="operationProviders"
      value="org.globus.progtutorial.services.core.providers.impl.MathProvider"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*/>
    <parameter name="persistent" value="true"/>
    <parameter name="handlerClass"
      value="org.globus.ogsa.handlers.RPCURIPProvider"/>
  </service>
</deployment>
```



## Compilar, Desplegar, Ejecutar

### ▶ Como usuario

```
▶ ./tutorial_build.sh  
org/globus/progtutorial/services/core/providers/  
schema/progtutorial/MathService/Math.gwsdl
```

### ▶ Como globus

```
▶ ant deploy -Dgar.name=$TUTORIAL_DIR/build/lib/<gar>  
▶ globus-start-container
```

### ▶ Como usuario

```
▶ java -classpath ./build/classes/:$CLASSPATH  
org/globus/progtutorial/clients/MathService/Client  
http://127.0.0.1:8080/ogsa/services/progtutorial/core/providers/MathService 5
```



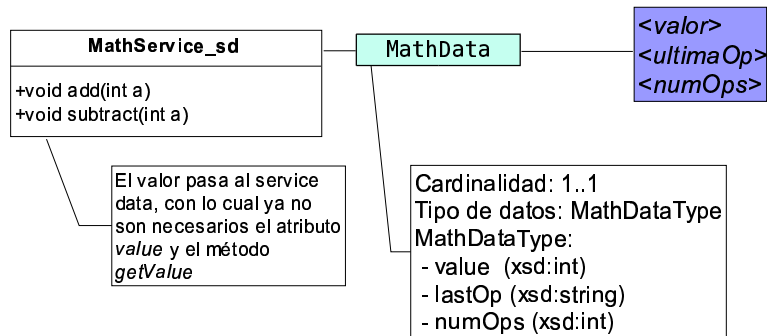
## Índice

- ▶ Instalación de GT3
- ▶ Un Servicio Sencillo
- ▶ Enfoques de Implementación
- ▶ **Service Data**
- ▶ Notificaciones
- ▶ Servicios Transientes



## Ejemplo

- ▶ Vamos a añadirle *service data* a MathService.



## Interfaz (I)

- ▶ Nuevo interfaz
  - ▶ \$TUTORIAL\_DIR/schema/progtutorial/MathService\_sd/Math.gwsdl
- ▶ Cambios con respecto al interfaz anterior:

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



## Interfaz (II)

- ▶ El tipo de datos MathDataType está definido en un fichero XML Schema aparte. También puede definirse dentro del <types> del fichero GWSDL.
- ▶ \$TUTORIAL\_DIR/schema/progtutorial/MathService\_sd/MathSDE.xsd

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



## Implementación y Descip.Despliegue

- ▶ La implementación del servicio cambia para trabajar con el service data.
  - ▶ \$TUTORIAL\_DIR/org/globus/progtutorial/services/core/servicedata/imp/MathImpl.java
  - ▶ Inicialización de MathData cuando se crea el servicio.
  - ▶ Actualización de MathData cada vez que se invoca una operación.
    - ▶ Almacenar el valor en MathData
    - ▶ Incrementar el numero de operaciones
    - ▶ Guardar nombre de última operación
- ▶ No hay parámetros nuevos en el WSDD, aunque los valores ahora corresponden a las nuevas clases.
  - ▶ \$TUTORIAL\_DIR/org/globus/progtutorial/core/servicedata/server-config.wsdd



## Cliente

- ▶ El cliente, además de invocar el método add, consulta el SDE MathData.

- ▶ `$TUTORIAL_DIR/org/globus/progtutorial/clients/MathService_sd/Client.java`



## Compilar, Desplegar, Ejecutar

- ▶ Como usuario

- ▶ `./tutorial_build.sh`  
`org/globus/progtutorial/services/core/servicedata/`  
`schema/progtutorial/MathService_sd/Math.gwsdl`

- ▶ Como globus

- ▶ `ant deploy -Dgar.name=$TUTORIAL_DIR/build/lib/<gar>`
  - ▶ `globus-start-container`

- ▶ Como usuario

- ▶ `javac -classpath ./build/classes/:$CLASSPATH`  
`org/globus/progtutorial/clients/MathService_sd/Client.java`
  - ▶ `java -classpath ./build/classes/:$CLASSPATH`  
`org/globus/progtutorial/clients/MathService_sd/Client`  
`http://127.0.0.1:8080/ogsa/services/progtutorial/core/servicedata/MathService 5`



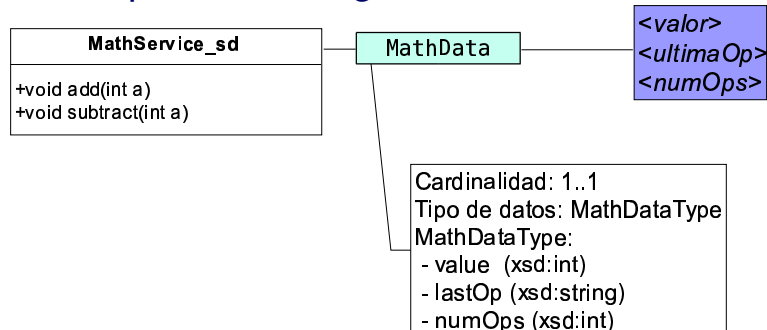
## Índice

- ▶ Instalación de GT3
- ▶ Un Servicio Sencillo
- ▶ Enfoques de Implementación
- ▶ Service Data
- ▶ **Notificaciones**
- ▶ Servicios Transientes



## Ejemplo

- ▶ Vamos a permitir que MathService sea un *origen de notificaciones*. El service data permanece igual.





## Interfaz

- ▶ Nuevo interfaz

- ▶ \$TUTORIAL\_DIR/schema/progtutorial/MathService\_sd\_notif/Math.gwsdl

- ▶ Cambios con respecto al interfaz anterior:

```
<gwsdl:portType name="MathPortType"
  extends="ogsi:GridService oksi:NotificationSource">

  <!-- operaciones -->

  <!-- service data -->

</gwsdl:portType>
```



## Implementación y Descip.Despliegue

- ▶ La implementación del servicio cambia para lanzar una notificación cada vez que hay un cambio.

- ▶ \$TUTORIAL\_DIR  
/org/globus/progtutorial/services/core/notifications/imp/MathProvider.java

- ▶ En el WSDD añadimos un nuevo operation provider

- ▶ org.globus.ogsa.impl.ogsi.NotificationSourceProvider
  - ▶ Nos proporciona toda la funcionalidad necesaria para que nuestro servicio sea una fuente de notificaciones (no tenemos que implementar esa funcionalidad nosotros)
  - ▶ \$TUTORIAL\_DIR/org/globus/progtutorial/core/notifications/server-config.wsdd



## Compilar y Desplegar

- ▶ Como usuario
  - ▶ `./tutorial_build.sh`  
`org.globus/progtutorial/services/core/notifications/schema/progtutorial/MathService_sd_notif/Math.gwsdl`
- ▶ Como globus
  - ▶ `ant deploy -Dgar.name=$TUTORIAL_DIR/build/lib/<gar>`
  - ▶ `globus-start-container`



## Cientes (I)

- ▶ En este ejemplo vamos a utilizar dos clientes.
  - ▶ `$TUTORIAL_DIR`  
`/org.globus/progtutorial/clients/MathService_sd_notif/ClientAdder.java`
    - ▶ Invoca el método `add()`
  - ▶ `$TUTORIAL_DIR`  
`/org.globus/progtutorial/clients/MathService_sd_notif/ClientListener.java`
    - ▶ Se suscribe al SDE MathData y recibe notificaciones.



## Cientes (II)

### ▶ Como usuario

- ▶ `javac -classpath ./build/classes/:$CLASSPATH org/globus/progtutorial/clients/MathService_sd_notif/ClientAdder.java`
- ▶ `javac -classpath ./build/classes/:$CLASSPATH org/globus/progtutorial/clients/MathService_sd_notif/ClientListener.java`
- ▶ En un terminal:  
`java -classpath ./build/classes/:$CLASSPATH -Dorg.globus.ogsa.schema.root=http://localhost:8080/ org/globus/progtutorial/clients/MathService_sd_notif/ClientListener http://127.0.0.1:8080/ogsa/services/progtutorial/core/notifications/MathService`
- ▶ En otro terminal:  
`java -classpath ./build/classes/:$CLASSPATH org/globus/progtutorial/clients/MathService_sd_notif/ClientListener http://127.0.0.1:8080/ogsa/services/progtutorial/core/notifications/MathService 5`



## Índice

- ▶ Instalación de GT3
- ▶ Un Servicio Sencillo
- ▶ Enfoques de Implementación
- ▶ Service Data
- ▶ Notificaciones
- ▶ Servicios Transientes



## Ejemplo

- ▶ Vamos a volver a utilizar el interfaz del primer ejemplo.
- ▶ La novedad es que ahora vamos a poder crear *instancias* de MathService.
- ▶ Reutilizamos:
  - ▶ Interfaz (GWSDL)
  - ▶ Implementación (Java)
  - ▶ Cliente (Java)
  - ▶ ¡Lo único que cambia es el WSDD!
- ▶ Ni siquiera tenemos que compilar y desplegar. La versión factoría/instancia se desplegó con el primer ejemplo.

Sistemas Grid Basados en GT3



## Descriptor de Despliegue (I)

- ▶ \$TUTORIAL\_DIR/org/globus/progtutorial/services/core/first/server-config.wsdd
- Primer ejemplo:**

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="progtutorial/core/first/MathService" provider="Handler" style="wrapped">
    <parameter name="name" value="MathService"/>
    <parameter name="baseClassName"
      value="org.globus.progtutorial.services.core.first.impl.MathImpl"/>
    <parameter name="className"
      value="org.globus.progtutorial.stubs.MathService.MathPortType"/>
    <parameter name="schemaPath"
      value="schema/progtutorial/MathService/Math_service.wsdl"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*" />
    <parameter name="persistent" value="true" />
    <parameter name="handlerClass"
      value="org.globus.ogsa.handlers.RPCURIPProvider"/>
  </service>
</deployment>
```

Sistemas Grid Basados en GT3



## Descriptor de Despliegue (II)

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="progtutorial/core/first/MathServiceFactory" provider="Handler" style="wrapped">
    <parameter name="instance-name" value="MathService"/>
    <parameter name="instance-baseClassName"
      value="org.globus.progtutorial.services.core.first.impl.MathImpl"/>
    <parameter name="instance-className"
      value="org.globus.progtutorial.stubs.MathService.MathPortType"/>
    <parameter name="instance-schemaPath"
      value="schema/progtutorial/MathService/Math_service.wsdl"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*/>
    <parameter name="persistent" value="true"/>
    <parameter name="handlerClass"
      value="org.globus.ogsa.handlers.RPCURIPProvider"/>
  </service>
</deployment>
```



## Descriptor de Despliegue (III)

```
<!-- Start common parameters -->
<parameter name="allowedMethods" value="*/>
<parameter name="persistent" value="true"/>
<parameter name="handlerClass" value="org.globus.ogsa.handlers.RPCURIPProvider"/>

<parameter name="className"
  value="org.gridforum.ogsi.Factory"/>
<parameter name="baseClassName"
  value="org.globus.ogsa.impl.ogsi.PersistentGridServiceImpl"/>
<parameter name="schemaPath"
  value="schema/ogsi/ogsi_factory_service.wsdl"/>
<parameter name="factoryCallback"
  value="org.globus.ogsa.impl.ogsi.DynamicFactoryCallbackImpl"/>
<parameter name="operationProviders"
  value="org.globus.ogsa.impl.ogsi.FactoryProvider"/>
```



## Descriptor de Despliegue (IV)

- ▶ Hemos configurado el WSDD de tal manera que:
  - ▶ Habrá una factoría persistente
  - ▶ Esta factoría creará instancias de MathService, que serán transientes.



## Crear instancia, ejecutar cliente

- ▶ Para crear una instancia:
  - ▶ `ogsi-create-service`  
`http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/MathFactoryService`
- ▶ Deberíamos ver:
  - ▶ Service successfully created:  
Handle:  
`http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/MathFactoryService/hash-24981262-1078167170769`  
Termination Time: infinity
- ▶ Este es el GSH de la instancia. Es un MathService, así que podemos acceder con el cliente del primer ejemplo.
  - ▶ `java -classpath ./build/classes/:$CLASSPATH org/globus/progtutorial/clients/MathService/Client http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/MathFactoryService/hash-24981262-1078167170769`  
5



## ¿Preguntas?



**Borja Sotomayor**

Facultad de Ingeniería - ESIDE

Universidad de Deusto

[bsotomay@eside.deusto.es](mailto:bsotomay@eside.deusto.es)