

Service Data

CSE 487/587
April 20, 2005

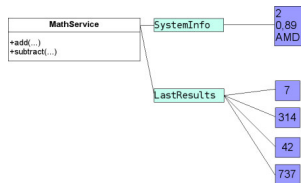
References:
Sotomayor's tutorial on Grid Services

Introduction

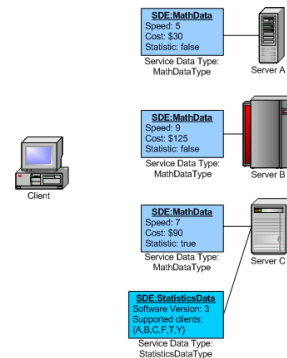
- Service Data is associated with the instance of a Grid Service.
- Data stored in Service Data Elements can be of two types:
 - Service Metadata: System data, supported interfaces, Cost of using the service, etc.
 - State Information: Operation results, intermediate results, runtime information, etc.

Example

- The example below shows the all-famous MathService with two SDEs:
 - SystemInfo: Service Metadata (System Data)
 - LastResults: State Information (Operation Results)



Another Example



Defining Service Data Elements

- The actual structure of the SDE is defined in an XSD file.
- This file is then imported into the service's gwsdl.
- SDEs can be of simple (LastResult) or complex (SystemInfo) type.
- One or more SDEs can be defined in the PortType definition (in the gwsdl).

Defining Service Data Elements

```
<complexType name="MemoryDataType">
  <sequence>
    <element name="freeMemory" type="long"/>
    <element name="maxMemory" type="long"/>
    <element name="totalMemory" type="long"/>
  </sequence>
</complexType>
```

Description of SDE in XML Schema



```
public class MemoryDataType implements java.io.Serializable {
    public long getFreeMemory();
    public void setFreeMemory(long freeMemory);
    public long getMaxMemory();
    public void setMaxMemory(long maxMemory);
    public long getTotalMemory();
    public void setTotalMemory(long totalMemory);
}
```

Java file (stub)

Defining Service Data Elements

- The definition in the PortType looks as follows:

```
<gwsdl:portType name="MemoryMonitorPortType" extends="ogsi:GridService ogsi:NotificationSource">
  <operation name="refresh">
    <input message="tns:RefreshInputMessage"/>
    <output message="tns:RefreshOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <sd:serviceData name="MemoryData"
    type="Data:MemoryDataType"
    minOccurs="1"
    maxOccurs="1"
    mutability="mutable"
    modifiable="false"
    nillable="true">
  </sd:serviceData>
</gwsdl:portType>
```

Defining Service Data Elements

- The following attributes refer to various properties of the SDE:
 - **minOccurs** : The minimum number of values that this SDE can have.
 - **maxOccurs** : The maximum number of values that this SDE can have. The value of this attribute can be unbounded, which indicates an array with no size limit.
 - **modifiable** : True or false. Specifies if the value of this SDE can be changed by a client.
 - **nillable** : True or false. Specifies if the value of this SDE can be NULL.
 - **mutability** : This attribute can have the following values:
 - **static**: The value of the SDE is provided in the GWSDL description.
 - **constant**: The value of the SDE is set when the Grid Service is created, but remains constant after that.
 - **extendable**: New elements can be added to the SDE, but not removed.
 - **mutable**: New elements can be added and removed.

Defining Service Data Elements

- ▣ Besides the above additions to the gwsdl there are a few more (import XSD file and inclusion of additional namespaces). Look at:

```
$TUTORIAL_DIR/schema/<username>_progtutorial/  
MemoryMonitorService/MemoryMonitor.gwsdl
```

- ▣ An additional namespace to package mapping must be setup in the `namespace2package.mappings` file to map the new service data namespace

Service implementation

- ▣ Two new objects are required for dealing with Service Data:
 - `private ServiceData mathDataSDE;`
 - `private MathDataType mathDataValue;`
- ▣ The first one refers to the SDE and the second one refers to the value of the SDE
- ▣ SDEs are created in the `postCreate` method of the service class. This method gets called after the creation of the service instance.

Service Implementation

```
public void postCreate(GridContext context) throws GridServiceException  
{  
    super.postCreate(context);  
  
    // Create Service Data Element  
    memoryDataSDE = this.getServiceDataSet().create("MemoryData");  
  
    // Create a MemoryDataType instance and set initial values  
    memoryDataValue = new MemoryDataType();  
    Timer timer = new Timer();  
    timer.schedule(new TimerTask()  
    {  
        Runtime runtime = Runtime.getRuntime();  
  
        public void run()  
        {  
            memoryDataValue.setFreeMemory(runtime.freeMemory());  
            memoryDataValue.setMaxMemory(runtime.maxMemory());  
            memoryDataValue.setTotalMemory(runtime.totalMemory());  
        }  
    }, 0, 10000);  
  
    // Set the value of the SDE to the MemoryDataType instance  
    memoryDataSDE.setValue(memoryDataValue);  
  
    // Add SDE to Service Data Set  
    this.getServiceDataSet().add(memoryDataSDE);  
}
```

Client

```
// Get a reference to the MemoryMonitor PortType  
MemoryMonitorServiceGridLocator memoryMonitorServiceLocator = new MemoryMonitorServiceGridLocator();  
MemoryMonitorPortType memoryMonitor = memoryMonitorServiceLocator.getMemoryMonitorServicePort(OSB);  
  
// Get Service Data Element "MemoryData"  
ExtensibilityType extensibility = memoryMonitor.findServiceData(QueryHelper.getNameQuery("MemoryData"));  
ServiceDataValuesType serviceData = AnyHelper.getAsServiceDataValues(extensibility);  
MemoryDataType memoryData = (MemoryDataType) AnyHelper.getAsSingleObject(serviceData, MemoryDataType.class);  
  
// Write service data  
System.out.println("Free Memory: " + memoryData.getFreeMemory());  
System.out.println("Total Memory: " + memoryData.getTotalMemory());  
System.out.println("Max Memory: " + memoryData.getMaxMemory());
```

Client

- ▣ We get the SDE called MathData using a GridService method called findServiceData, and a 'helper' class to resolve the name into something the Grid Service can understand.
- ▣ The findServiceData doesn't return a MathDataType class, but an ExtensibilityType class. This needs to be cast into a MathDataType using more helper classes.
- ▣ This returns a MathDataType object. We can use it like any other local object