

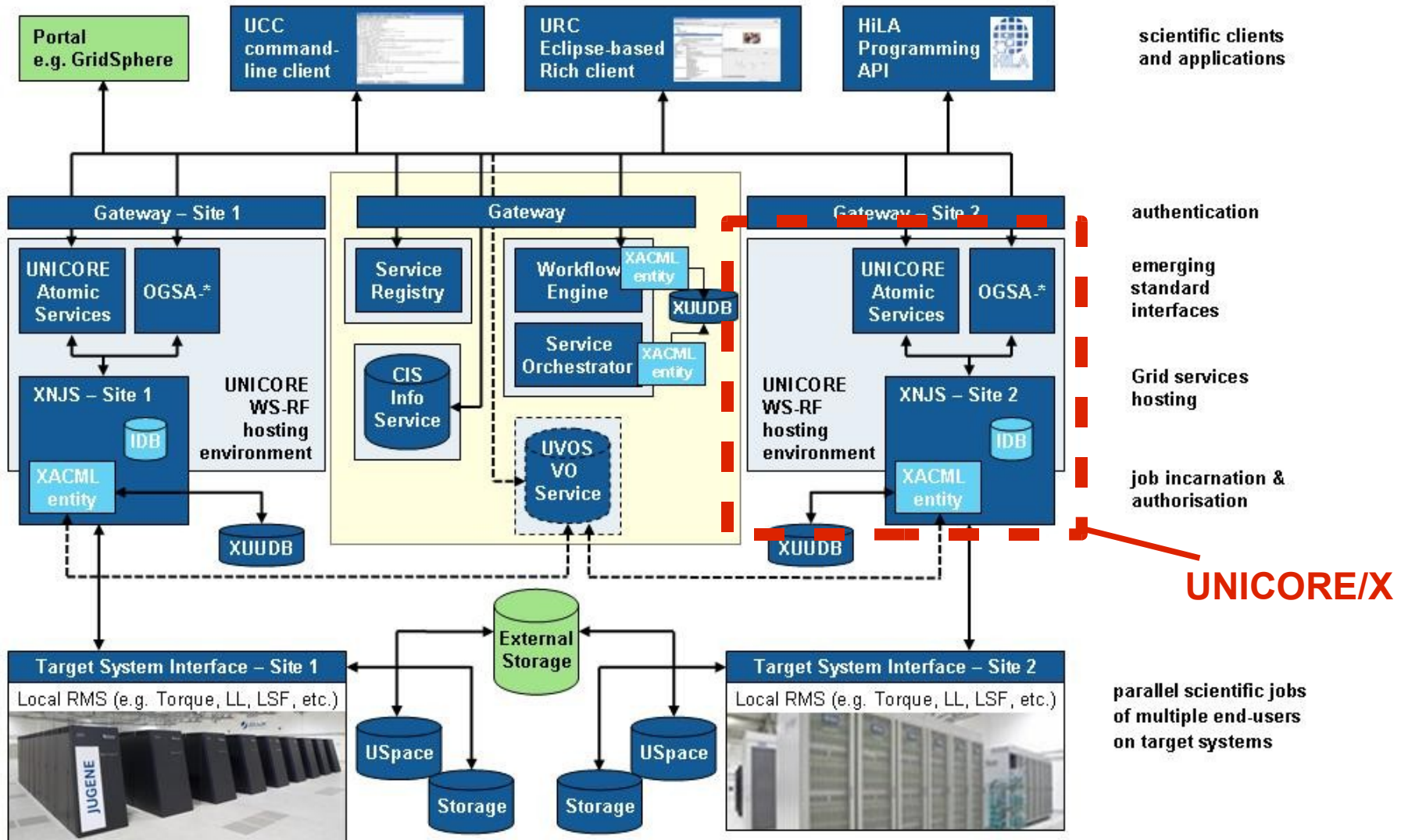
# Programming UNICORE 6 services

Bernd Schuller and the UNICORE team  
Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH  
March 17, 2010  
OGF28 Munich

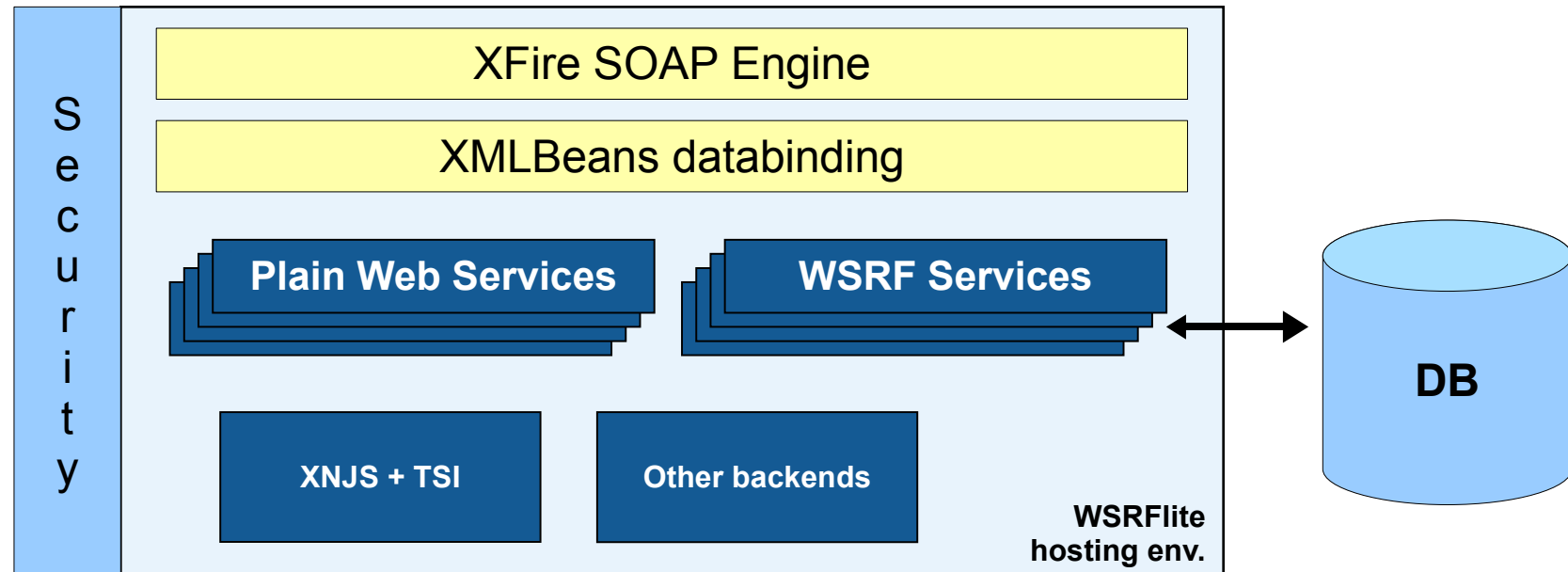
## Outline

- The UNICORE/X service container
- Web services and WSRF
- Tools
- XML Schema
- Java interfaces and Java implementation classes
- Deployment
- Client code

## UNICORE/X service container



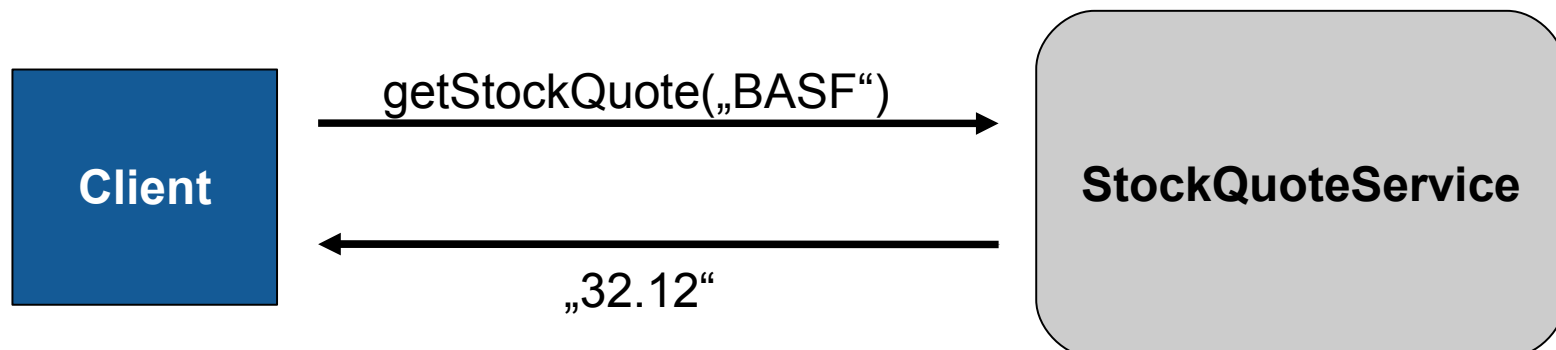
## UNICORE/X service container



- Written in Java (requires Java 5 or later)
- XFire + XMLBeans SOAP stack, Jetty 6 web server, HTTPClient for outgoing calls
- Persistence layer (H2, MySQL)
- Security APIs

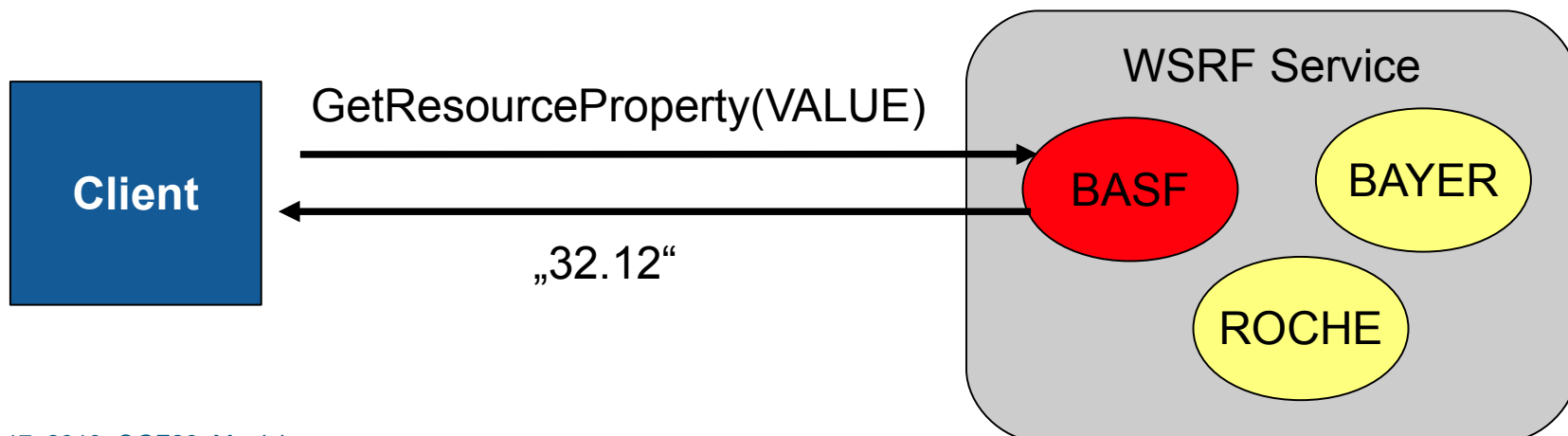
## Web services

- Web services provide basic communication paradigm:
  - exchange of XML messages using the SOAP protocol
- Contact remote service
  - e.g. `http://www.some.com/StockQuoteService`
- Provide parameters as an XML document
- Retrieve results in a XML document



## WSRF services

- Web Service Resource Framework, standardised by OASIS in 2006
- Allows building web services for accessing and managing multiple resources : „WS-Resources“
- WSRF service instances have
  - A finite **lifetime** that can be controlled by clients
  - **Properties** that can be accessed using standard methods



# Example: UNICORE 6 Job properties

```
<jms:JobProperties xmlns:jms="http://unigrids.org/2006/04/services/jms"
                  xmlns:rl="http://docs.oasis-open.org/wsrp/1.2">
  <jms:SubmissionTime>2009-02-25T15:23:29.784+02:00</jms:SubmissionTime>
  <jms:OriginalJSDL...</jms:OriginalJSDL>
  <jms:ExecutionJSDL>...</jms:ExecutionJSDL>
  <jms:Log>...</jms:Log>
  <jms:TargetSystemReference>...</jms:TargetSystemReference>
  <jms:WorkingDirectoryReference>...</jms:WorkingDirectoryReference>
  <jms:StdOut>stdout</jms:StdOut>
  <jms:StdErr>stderr</jms:StdErr>
  <typ:StatusInfo xmlns:typ="http://unigrids.org/2006/04/types">
    <typ:Status>SUCCESSFUL</typ:Status>
  </typ:StatusInfo>
  ...
  <rl:CurrentTime>2009-02-25T21:11:55.300+02:00</rl:CurrentTime>
  <rl:TerminationTime>2009-02-26T15:23:29.771+02:00</rl:TerminationTime>
</jms:JobProperties>
```

## Full example

- Requires Apache Maven (<http://maven.apache.org>)
- Checkout from SVN

```
svn co http://unicore.svn.sourceforge.net/svnroot/unicore/SampleWSRFService
```

- Optional:
  - Create Eclipse project files

```
cd SampleWSRFService  
mvn eclipse:eclipse
```

- Import project into Eclipse

## XML Schema

- XML Schema is used to define the XML part of the service interface
  - Input and output messages
  - Resource properties for WSRF services
- Apache XMLBeans
  - Generate Java classes from XML schema
  - Using Maven:

```
mvn xmlbeans:xmlbeans
```

## Messages

```
<xsd:element name="AddRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Integer"
        type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="AddResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="NewSum"
        type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

## Properties

```
<xsd:element name="AdderProperties">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="test:Sum"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Sum">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Sum"
        type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

**src/main/schema/example.xsd**

# Java interface

```
package org.example;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.xml.namespace.QName;
import org.example.xmlbeans.AddRequestDocument;
import org.example.xmlbeans.AddResponseDocument;
import de.fzj.unicore.wsrflite.xmlbeans.WSResource;

@WebService(targetNamespace="http://www.example.org", portName="Adder")
public interface AdderIFace extends WSResource {

    /**
     * method to add an item to the registry
     */
    @WebMethod(action = "http://www.example.org/AddRequest")
    public AddResponseDocument add(AddRequestDocument request);

}
```

**src/main/java/org/example/AdderIFace.java**

# Service implementation

```
public class AdderImpl extends WSRResourceImpl implements AdderIFace {

    @Persist
    private int sum=0;

    /** which XML element holds the service's resource properties */
    public QName getResourcePropertyDocumentQName() {
        return AdderPropertiesDocument.type.getDocumentElementName();
    }

    /** initialise() is called when a new service instance is created */
    public void initialise(String serviceName, Map<String, Object> init)
        throws Exception{
        super.initialise(serviceName, init);
        // add java code for holding the resource properties
        properties.put(RPSumQName, new SumProperty(this));
    }

    public AddResponseDocument add(AddRequestDocument request) {
        //...implement addition
    }
}
```

**src/main/java/org/example/AdderImpl.java**

## Further required Java classes

- Home class (usually trivial subclass of a WSRFlite base class)
  - Instance creation
- Resource properties
  - Simple, if content is immutable (e.g. creation time of some resource)
  - May be complex and requiring some coding
  - Many examples can be found in UNICORE source code

## Deployment

- Compile, run tests and build jar file

```
mvn clean install
```

- Copy jar file into UNICORE/X lib directory
- Edit conf/wsrf-lite.xml and add service declaration

```
<service name="Adder" wsrf="true">  
  <interface class="org.example.AdderIFace"/>  
  <implementation class="org.example.AdderHomeImpl"/>  
  <initTask class="org.example.SampleInitTask"/>  
</service>
```

- Service will be deployed on server restart

## Client code

- Use proxy client
  - Java interface which performs web service calls behind the scenes
- UNICORE provides convenient helper methods and base classes

```
//create a client for communicating with the adder service
EndpointReferenceType epr=Utilities.makeEPR("Adder", "default_adder");
BaseWSRFClient base=new BaseWSRFClient(epr);
```

```
//create a proxy client with the correct interface
AdderIFace adder=base.makeProxy(AdderIFace.class);
```

```
//now we can add
AddRequestDocument request=AddRequestDocument.Factory.newInstance();
request.addNewAddRequest().setInteger(10);
adder.add(request);
```

**src/test/java/org/example/TestAdderBasic.java**