Enabling Grids for E-science in Europe

www.eu-egee.org

15th October 2004

Technologies for Building Grids



EGEE is a project funded by the European Union under contract IST-2003-508833

Objectives

Enabling Grids for E-science in Europe

- To understand the major Java environment packages for writing and deploying Java services and clients
- Basic concepts of containers
- Some JAX-RPC basics.
- Basics of deploying to containers
- An introduction to the ANT tool.

Enabling Grids for E-science in Europe

Making java based web services

Java web services development environment

Enabling Grids for E-science in Europe

- Main components used:
  - J2EE (*Java 2 Enterprise Edition*)
    - Provides basic Java environment: compiler, virtual machine, etc.
  - JWSDP (*Java Web Services Developer Pack*)
    - Provides web services specific java APIs.
    - Automatic code generation to provide web services communication (eg. stubs)
  - Container (Tomcat/Axis)
    - Provides access to low level communications protocols (eg. http through web server. Axis also provides some functions such as XML parsing)

What's in JWSDP

Enabling Grids for E-science in Europe

- (*JAX = Java APIs for XML*)
- JAX-RPC (*Remote Procedure Calls*)
  - APIs for using web services and automatically creating 'glue' code to make it work.
- JAXP (*Processing*)
  - XML parsing components
- JAXR (*Registries*)
  - UDDI etc.
- JAXM (*Messaging*)
  - depricated
- JAXB (*Binding*)
  - Mapping XML to Java
- SAAJ (*SOAP with attachments API for Java*)
  - Produce and consume SOAP messages

The classpath

Enabling Grids for E-science in Europe

- With java based web services development the most common source of problems is with incorrectly set variables in the classpath.
- Similarly with Globus toolkit based grid development.
- The classpath is often the first place to look for compile time problems.

Axis – Tomcat what's the difference?

Enabling Grids for E-science in Europe

- Tomcat was developed as a container to add functionality to the Apache web server.
- Tomcat deals mainly with servlets (non- web services).
- Tomcat was adapted to deal with web services initially
- Axis is the new development specifically for web services – actually is like Tomcat +.
- Can use Tomcat but moving towards Axis.

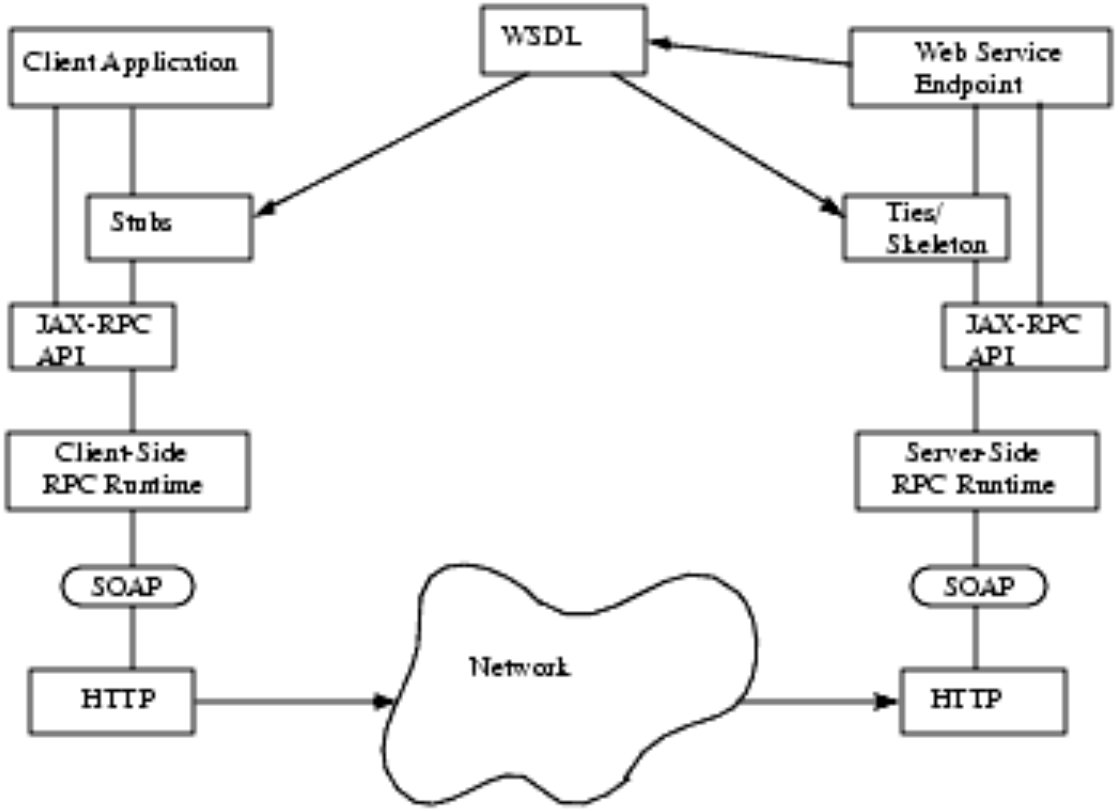
JAX - RPC

JAX-RPC API packages

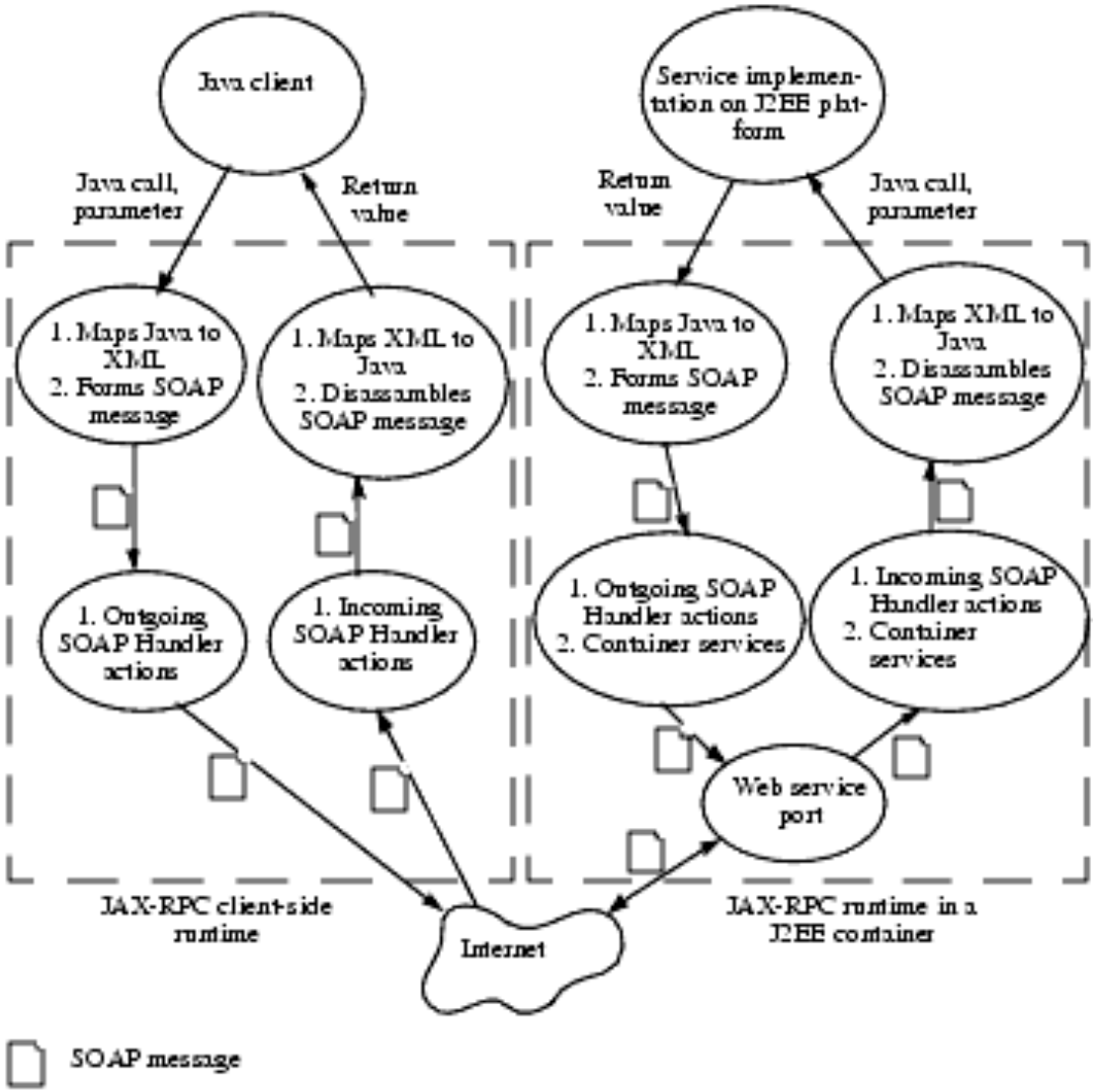
Enabling Grids for E-science in Europe

• javax.xml.rpc	Core classes for the client side programming model
• javax.xml.rpc.encoding	Java primatives <-> XML SOAP messages
• javax.xml.rpc.handler • javax.xml.rpc.handler.soap	processing XML messages
• javax.xml.rpc.holders	support the use of IO parameters
• javax.xml.rpc.server	minimal API for web service inplementation
• Javax.xml.rpc.soap	specific SOAP bindings

JAX-RPC Architecture



Java web service flow



Defining a service

Enabling Grids for E-science in Europe

- A service can be defined starting with:
  - A java interface
  - A WSDL document
- Which to use?
  - If the service end point interface is defined in java it may not be interoperable with services/clients defined in other languages
  - If the service is initially defined in WSDL it will be open

Client side Implementation

wscompile

- Generates
  - Compiled class files + optionally source files for stubs to interface with client side JAX-RPC
  - WSDL file
  - Model file

Example commandline

```
wscompile -gen:client -d output/client -classpath classpath config-file
```

(add `-keep -s` to retain java source files)

config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
    <service name="……..."
    targetNamespace="……"
    typeNamespace="……."
    packageName="……">

        <interface name="……"/>
    </service>
</configuration>
```

name = name of service
targetNamespace = namespace of WSDL for names associated with the service eg. port type
typeNamespace = namespace of WSDL for data types
packageName = name of java package

Generated files

Enabling Grids for E-science in Europe

Some of the client side generated files:

Service	Service.java
	Service_Impl.java
	Service_SerializerRegistry.java
Exception	ServiceException_SOAPSerializer.java
	ServiceException_SOAPBuilder.java
Value type	Info_SOAPSerializer.java
	Info_SOAPBuilder.java
Interface	Interface_Stub.java
	method.java

Service.java file

- The `Service.java` file corresponds to the definition of the interface for the web service, ie it contains the same info as the `<service>` element in the config file.

```
package servicePackage;

import javax.xml.rpc.*;

Public interface Service extends javax.aml.rpc.Service
{
        public servicePackage getServicePort();
}
```

Referencing the stub

- In order to get an object to reference the stub you have to instantiate Service_Impl.
  - (Unfortunately this name is only recommended)

- `Service_Impl service = new Service_Impl ();`

- `value* name = (value)service.getServicePort ();`

- With this reference you can call the methods of the service.

Stub Interface (javax.xml.rpc.Stub)

Enabling Grids for E-science in Europe

```
Public interface Stub
{
   public abstract Object _getProperty (String name) throws
   JAXRPCException;
   public abstract Iterator  _getPropertyNames ();
   public abstract void _setProperty(String name, Object
   value) throws     JAXRPCException;
}
```

These methods allow the stub to be configured by setting various properties.

Deploying with JWSDP - Tomcat

Deployment: Making the container aware of a servlet

Enabling Grids for E-science in Europe

- The web server has to be aware of the interface and exposed methods of a servlet in order to use it.
- Using Tomcat as an example this can be done in a number of ways.

1. Enter the values manually into the SOAP admin page from a Deployment descriptor.
2. You can use the SOAP manager application from the command line
3. You can manually edit Tomcat's WEB-INFO/web.xml file
4. You can create a WAR file and place it in Tomcat's webapps folder
5. You can use ANT

Using a WAR file

Enabling Grids for E-science in Europe

- A WAR file is basically an archive description of a servlet installation

  (*JAR and WAR naming derives from UNIX TAR – java archive, web archive, tape archive*).

- Example: placed in Tomcat's webapps folder it can be interpreted by the container.

Deployment Descriptor

Enabling Grids for E-science in Europe

A SOAP manager file

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
   id="urn:stock-onhand">
   <isd:provider type="java" scope="Application" methods="getQty">
        <isd:java class="StockQuantity"/>
   </isd:provider>
   <isd:faultListener>org.apache.soap.sever.DOMFaultListener</isd:faultListener>
</isd:service>
```

Some containers (Tomcat) provide GUIs for deployment

Additional WAR files required for JWSDP

Enabling Grids for E-science in Europe

`WEB-INF/web.xml`	Web application deployment descriptor
`WEB-INF/jaxrpc-ri.xml`	JWSDP-specific deployment information
`WEB-INF/model`	Model file generated by `wscompile`

web.xml file

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
   2.3//EN"
   "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">


<web-app>
   <display-name>Service Name</display-name>
   <description>A web service application</description>
</web-app>
```

Creating a deployable WAR file

Enabling Grids for E-science in Europe

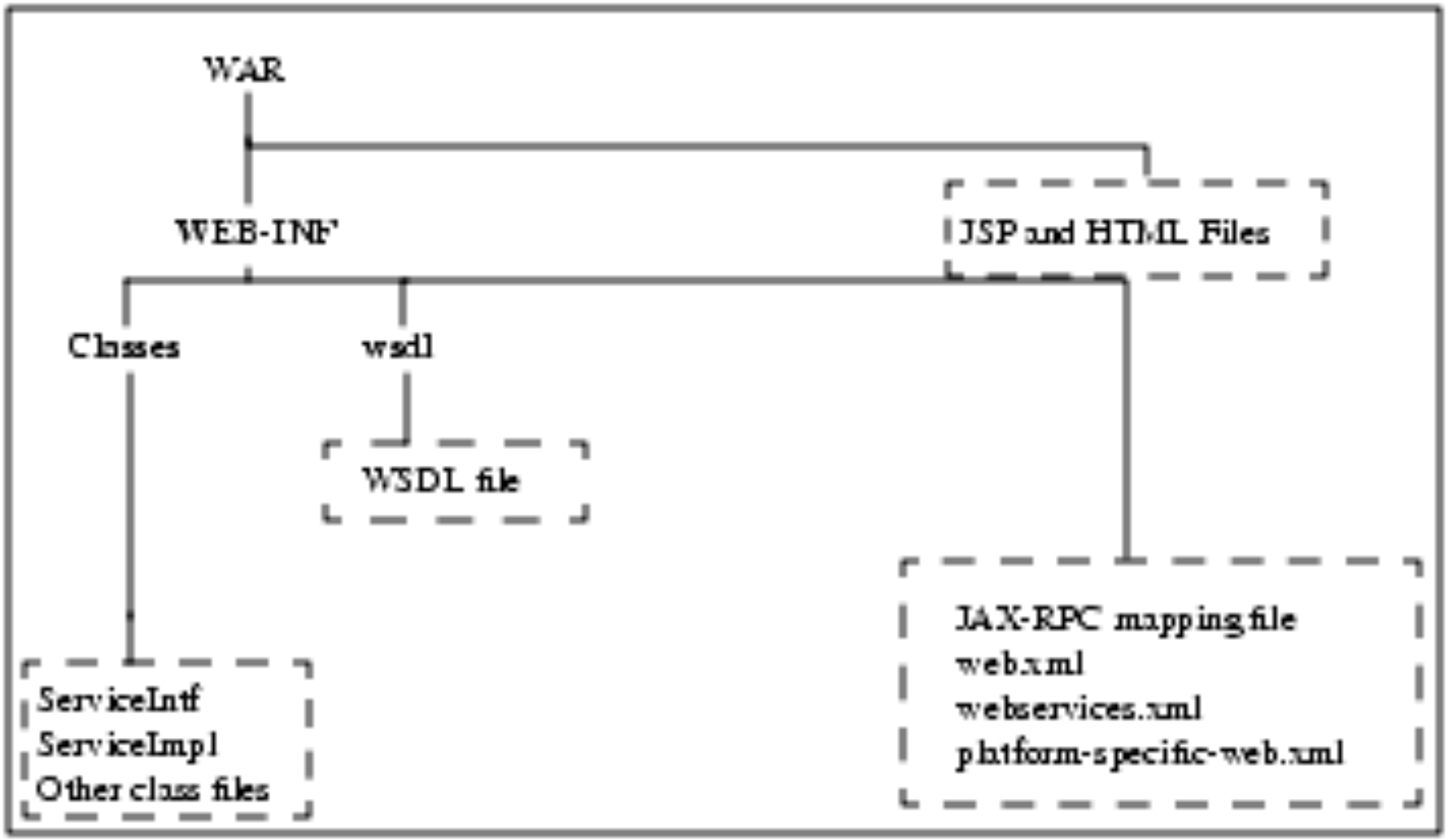
```
wsdeploy -o targetFileName portableWarFileName
```

The process is informed by the content of the `jaxrpc-ri.xml` file.

The archive contains:

- class files and resources
- compiled class files for the ties
- compiled class files for serializers
- WSDL (in WEB-INF directory)
- model file for the service (in WEB-INF)
- modified `web.xml` file
- `jaxrpc-ri-runtime.xml` (based on `jaxrpc-ri.xml`)

Package Structure for JAX-RPC Service Endpoint



Introducing the ANT tool

Ant

Enabling Grids for E-science in Europe

Ant – Another Neat Tool

- Ant is a tool for building projects
- It uses a build file, called build.xml, to determine how to build a particular project

ANT vs MAKE

Enabling Grids for E-science in Europe

- ANT performs similar functions to `make` as a software project build tool.
- `Build.xml` is similar to a `Makefile`
- It is written in Java and is therefore entirely platform independent

Understanding Ant

Enabling Grids for E-science in Europe

- The key to understanding Ant is to understand the build.xml file structure

```
<project ..>
 <property .../>
 <target .../>
</project>
```

- The root element is `project`
- There are then `properties` elements, which can be set externally or internally to the file
- There are also `target` elements which specify what actions are to be taken

Target in ANT

Enabling Grids for E-science in Europe

The use of the word target can be confusing.

In ANT a target is an action

It is not a destination – as the word is used in some other development environments.

(There is also a *task* in ANT – this is a finer grained action than a target.)

Example Project (1)

Enabling Grids for E-science in Europe

- The next few slides will demonstrate the use of Ant to build a very simple project

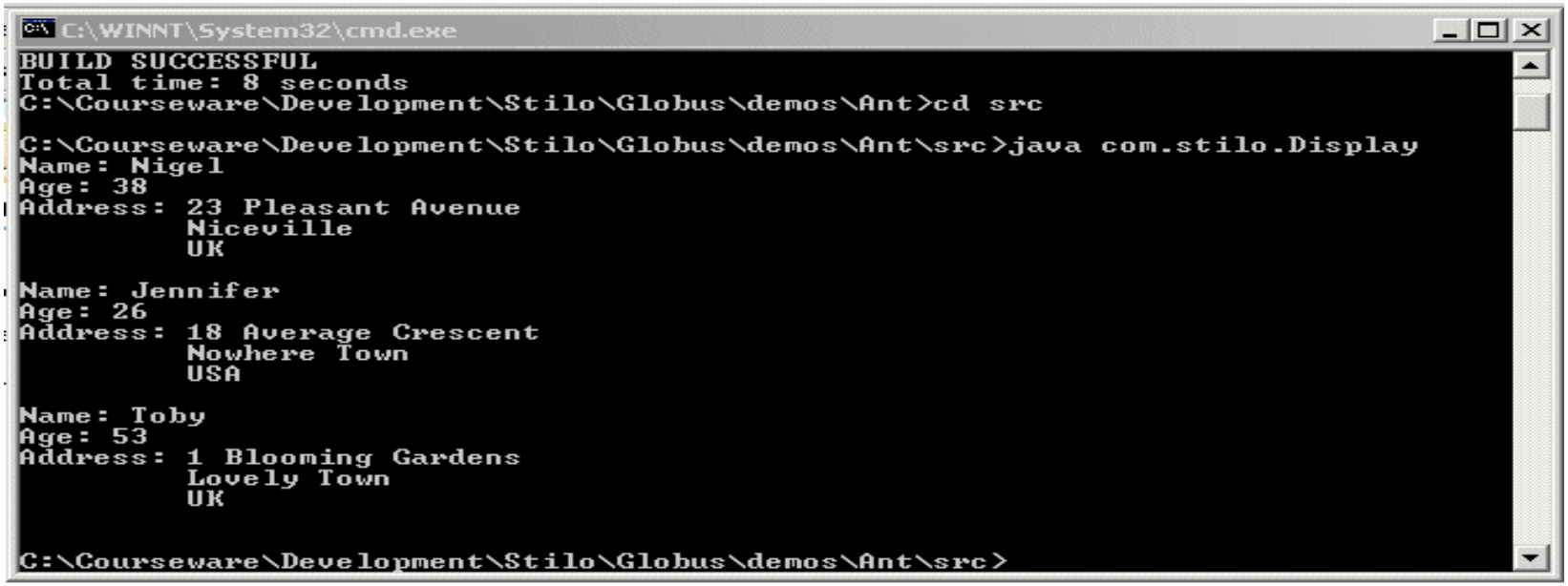
- There are three classes in the Project

  - Person
    - Contains a name, age and an Address
  - Address
    - Contains street, town and country information
  - Display
    - Instantiates 3 Person classes and displays their details on the command line

Example Project (2)

- All the source files are in a package structure in a directory called `src`

- An example output would be:



Compile Example

- This example will perform a simple compile

```
<project name="PeopleProject" default="compile" basedir=".">
    <!-- set global properties for this build -->
    <property name="src" location="src"/>
    <target name="compile" >
        <!-- Compile the java code  -->
        <javac srcdir="${src}"/>
    </target>
</project>
```

- There is only one target, which will compile all classes in the same location as the source files

More Complex Example

Enabling Grids for E-science in Europe

- This example will now create a directory called `build` and put the compiled code in there, preserving package structure

```
<property name="src" location="src"/>
<property name="build" location="build"/>

<target name="init">
    <mkdir dir="${build}"/>
</target>

<target name="compile" depends="init" >
    <!-- Compile the java code  -->
    <javac srcdir="${src}" destdir="${build}" />
</target>
```

Creating JAR files (1)

- This `build.xml` will require two runs:
  - One to compile and package in to a JAR
  - One to clean up unnecessary files

```
<project name="PeopleProject" default="dist" basedir=".">

<target name="init" description="prepare the environment">
    <mkdir dir="${build}"/>
    <mkdir dir="lib"/>
</target>

...
```

Creating JAR files (2)

Enabling Grids for E-science in Europe

```
...

   <target name="compile" depends="init" >
      <!-- Compile the java code  -->
      <javac srcdir="${src}" destdir="${build}"/>
   </target>

   <target name="dist" depends="compile" >
      <jar jarfile="lib/PeopleProject.jar" basedir="${build}"/>
   </target>

   <target name="clean" description="clean up">
       <delete dir="${build}"/>
   </target>

</project>
```

Invoking targets

- Here `dist` invokes `clean` so we now only require one run of `ant`

```
<target name="dist" depends="compile" >
    <jar jarfile="lib/PeopleProject.jar" basedir="${build}"/>
    <antcall target="clean"/>
</target>

<target name="clean" description="clean up" >
    <delete dir="${build}"/>
</target>
```

Setting the classpath

Enabling Grids for E-science in Europe

- To set the classpath, use the `classpath` task and specify the appropriate `pathelement` tags

```
<javac srcdir="${src}" destdir="${build}">
    <classpath>
        <pathelement path="${basedir}/lib/Jar1.jar"/>
        <pathelement path="${basedir}/lib/Jar2.jar"/>
        <pathelement path="${basedir}/lib/Jar2.jar"/>
    </classpath>
</javac>
```

- All of the above JAR files will now be on the class path when the source is compiled

Copying Files

Enabling Grids for E-science in Europe

- The copy task in ant allow you to copy files and/or directories

```
<copy file="Example.txt" toDir="MyFiles/"/>
```

- This example will copy the file `Example.txt` to the `MyFiles` directory.

Looking ahead to Globus and Ant

Enabling Grids for E-science in Europe

- Much of the work in building and deploying Grid Services for Globus can be done using Ant build files
- The next few slides will go through some of the build files that are provided with Globus
- You can also write your own build files for use with Globus



startContainer(1)

- The main build file that is provided with Globus contains a target called `startContainer`

```
<target name="startContainer">
  <ant antfile="${build.services}" target="startContainer"/>
</target>
```

- This is invoking a target with the same name in another build file called `build.services`
- We will see this target on the next slide

startContainer(2)

Enabling Grids for E-science in Europe

- Here is the target in `build.services`

```
<target name="startContainer" depends="setJbossClientClasspath,
setWebSphereClientClasspath">

    ...

</target>
```

- The dependencies add JAR files to the classpath

Starting a Java VM

Enabling Grids for E-science in Europe

- The `java` task runs the specified Java class

```
<target name="startContainer" depends="setJbossClientClasspath,
setWebSphereClientClasspath">

 <java
classname="org.globus.ogsa.server.ServiceContainer"
        fork="yes">
    …

 </java>

</target>
```

- The `fork` attribute indicates that this class should be run in a new VM

Specifying Java Options (1)

- The `arg` task allows you to specify arguments for the class

```
<java classname="org.globus.ogsa.server.ServiceContainer"
        fork="yes">
  ...
  <arg line="-p ${service.port}"/>
  <arg line="-${container.initialization}"/>
  ...
</java>
```

Specifying Java Options (2)

Enabling Grids for E-science in Europe

- The arguments being used would, so far, be equivalent to:

```
java org.globus.ogsa.server.ServiceContainer -p 8080 -eager
```

- The `-p` switch specifies the port
- `container.initializion` has evaluated to `eager`

Classpath

Enabling Grids for E-science in Europe

- The classpath is set using properties specified in an external properties file

```
<project ... >
  <property file="ogsa.properties"/>

<arg line="-p ${service.port}"/>
<arg line="-${container.initialization}"/>

...

<classpath>
  <pathelement location="${ogsa.jar}"/>
  <pathelement location="${samples.lib}/${samples.jar}"/>
  <path refid="classpath"/>
</classpath>
```