WEB SERVICES em AXIS e .NET

Tutorial

Sistemas Distribuídos II

(2002/2003)

Introdução

- O trabalho proposto consiste no desenvolvimento de um protótipo de um Web Service. Pretende-se mostrar os diversas fases necessárias para disponibilizar a implementação de um objecto Java sob a forma de Web Service.
- Serão usados dois ambientes de desenvolvimento de Web Services. O ambiente Apache Axis e o ambiente Microsoft .NET.
- No ambiente Axis será descrito passo a passo como criar um Web Service a partir de um objecto Java já existente. Por motivos de completude será mostrada brevemente uma forma de fazer um cliente no mesmo ambiente para consumir o serviço.
- No ambiente .NET serão descritos os passos necessários para fazer um cliente para consumir um serviço, no caso, o que foi criado usando o Axis. Por motivos de completude, será mostrado brevemente como se pode rapidamente fazer um Web Service em .NET.
- Os exemplos e a apresentação oral incidirão principalmente sobre o desenvolvimento do Web Service com o Axis e sobre o desenvolvimento do cliente protótipo em ambiente .NET Compact Framework.

Ambiente Apache AXIS

Ferramentas a usar

- Sun Java Development Kit
- Apache Tomcat – Servlets Engine
- Apache Axis – SOAP Engine

Preparação do ambiente de referência

- Instalar JDK 1.4.1.02
- Instalar Motor de Servlets TomCat 4.1.24
- Instalar Apache Axis 1.1 RC2
- Testar a instalação do ambiente
  - Tomcat: http://localhost:8080/index.jsp
  - Axis: http://localhost:8080/axis/happyaxis.jsp
- Consultar também http://asc.di.fct.unl.pt/sd2/doc

Instalar Tomcat

- Seguir as instruções em

  http://jakarta.apache.org/tomcat/tomcat-4.1-doc/RUNNING.txt

Instalar AXIS

- As instruções de instalação e algumas informações importantes estão em
  - http://cvs.apache.org/viewcvs.cgi/~checkout~/xml-axis/java/docs/install.html

Cenário

- Temos a implementação de um objecto em Java que se pretende tornar acessível na forma de Web Service
- A implementação desse objecto é código Java igual ao de qualquer aplicação desenvolvida nesta linguagem*

* deve-se ter em conta algumas limitações de interoperabilidade

Cenário (exemplo)

```
package pocketbank.pdaservice.api;

public interface PdaService
{
    public RegisterUserResponse registerUser(String challenge, int contractId, String pinA, String subset);

    public AccountsResponse getAccounts(String challenge, int contractId);
    public BalanceResponse getBalance(String challenge, int contractId, AccountInfo account);
    public StatementResponse getStatement(String challenge, int contractId, AccountInfo account);
    public NIBResponse getNIB(String challenge, int contractId, AccountInfo account);

    public Response execPayment(String challenge, int contractId, AccountInfo account, String entity, String

    public String testing();
}
```

Desenvolvimento/Deployment

- 1. Compilação do objecto (JAVA)
- 2. Copiar classes para directoria árvore de directórios do AXIS
- 3. Prototipagem do WebService:

  Registo no ambiente AXIS:
  - Opção 1: método imediato
  - Opção 2: método personalizado

Criar um Web Service - JWS

- Método Imediato:
  - Renomear ficheiro .java para .jws
  - Copiar o ficheiro para o directório do Axis
  - Aceder ao serviço usando http://localhost:8080/axis/ServicoExemplo.jws

  Desvantagens:
  - código fonte necessário
  - compilação em tempo de invocação
    - invocação dinâmica: lento, código não verificável, ...
  - não suporta qualquer tipo de opção de personalização
    (ver método seguinte)

Criar um Web Service – WSDD (1)

- Método personalizado:
  - Criar ficheiro WSDD onde se descreve como o Axis deve disponibilizar o objecto
  - Compilar e copiar as classes para o directório de classes do Axis
  - Instalar esse ficheiro no Axis usando o AdminClient

Criar um Web Service – WSDD (2)

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="PdaService" provider="java:RPC" style="rpc" use="encoded">
      <parameter name="wsdlTargetNamespace" value="http://api.pdaservice.pocketbank"/>
      <parameter name="wsdlServiceElement" value="PdaServiceService"/>
      <parameter name="wsdlServicePort" value="PdaService"/>
      <parameter name="className" value="pocketbank.pdaservice.PdaServiceImpl"/>
      <parameter name="wsdlPortType" value="PdaService"/>
      <parameter name="allowedMethods" value="*"/>
      <parameter name="scope" value="Application"/>
  </service>
</deployment>
```

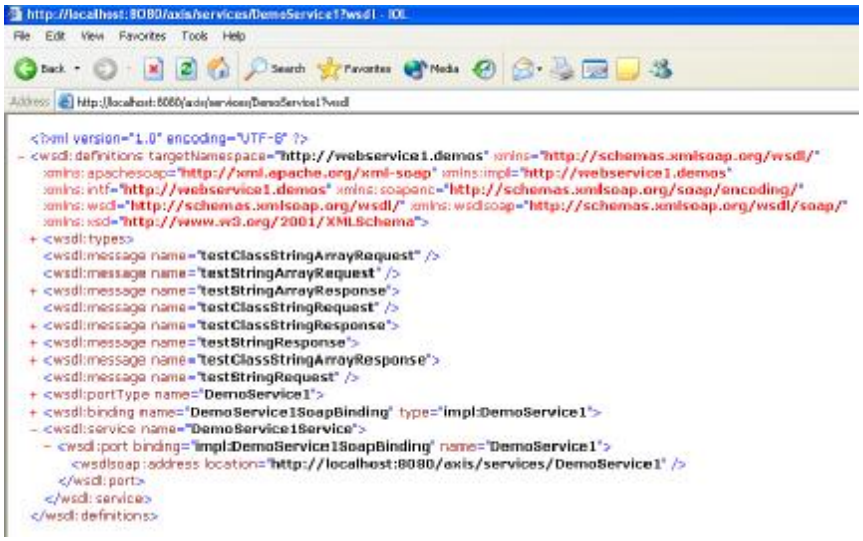
java org.apache.axis.client.AdminClient deploy.wsdd

Criar um Web Service – WSDD (3)

- Se não tiver ocorrido nenhum erro, o serviço estará já disponivel no endereço http://localhost:8080/axis/services/PdaService
- A lista de serviços disponíveis está em http://localhost:8080/axis/servlet/AxisServlet

Lista de serviços disponíveis

WSDL (Definição do Serviço)



Mapeamento de tipos de dados (1)

- Mapeamento de tipos de dados básicos estão normalizados
- O Axis inclui já funcionalidade para *serializar/deserializar* de classes que respeitem as regras do standard JavaBean (constructor por defeito, setters e getters para cada atributo)
- Existe possibilidade especificar mapeamentos personalizados

Mapeamento de tipos de dados (2)

- É preciso ter cuidado!
- Não existem standards para todos os tipos de objectos, pelo que diferentes implementações poderão oferecer mapeamentos incompatíveis
  - Exemplo1: o Axis mapea Hashtables esse mapeamento não é suportado em .NET
  - Exemplo2: o Axis mapea excepções Java em wsdl:faults mas o .NET não.
- Boa prática: tipos correspondentes a esquemas de serialização pré-definidos no ambiente (AXIS, .NET, ...)

Mapeamentos – a prática (AXIS) (1)

- Se tivermos classes do tipo JavaBean que não incluam arrays o Axis *serializa/deserializa-as* automaticamente
- Se houverem atributos que sejam arrays, terá de se especificar no WSDD a classe a usar para *serializar/deserializar* esse objecto
- Noutros ambientes ?

Mapeamentos – a prática (2)

```
<typeMapping
  xmlns:ns="http://api.pdaservice.pocketbank"
  qname="ns:Balance"
  type="java:pocketbank.pdaservice.api.Balance"
  serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
  deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
<typeMapping
  xmlns:ns="http://api.pdaservice.pocketbank"
  qname="ns:ArrayOfBalance"
  type="java:pocketbank.pdaservice.api.Balance[]"
  serializer="org.apache.axis.encoding.ser.ArraySerializerFactory"
  deserializer="org.apache.axis.encoding.ser.ArrayDeserializerFactory"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
```

Mapeamento – a prática (3)

- Quando existem muitas classes, torna-se inconveniente escrever o WSDD manualmente e aumentam as probabilidades de inconsistencias
- Existe uma forma de produzir o WSDD automaticamente usando Java2WSDL e WSDL2Java

Criar WSDD automaticamente (1)

- java org.apache.axis.wsdl.Java2WSDL
  -o deploy.wsdl
  -lhttp://maquina:porta/axis/services/NomeServiço
  nome_classe
- java org.apache.axis.wsdl.WSDL2Java
  -o directorio_destino
  -d Session -s -S true
  deploy.wsdl

- Este processo é usado para criar os stubs e skeletons, por isso o WSDD está definido para uma classe que implementaria o objecto a disponibilizar. É preciso mudar o nome dessa classe para o nome da nossa

Criar WSDD automaticamente (2)

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="PdaService" provider="java:RPC" style="rpc" use="encoded">
      <parameter name="wsdlTargetNamespace" value="http://api.pdaservice.pocketbank"/>
      <parameter name="wsdlServiceElement" value="PdaServiceService"/>
      <parameter name="wsdlServicePort" value="PdaService"/>
      <parameter name="className" value="pocketbank.pdaservice.PdaServiceImpl"/>
      <parameter name="wsdlPortType" value="PdaService"/>
      <parameter name="allowedMethods" value="*"/>
      <parameter name="scope" value="Application"/>

      <typeMapping
        xmlns:ns="http://api.pdaservice.pocketbank"
        qname="ns:Balance"
        type="java:pocketbank.pdaservice.api.Balance"
        serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
        deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
      ...
  </service>
</deployment>
```

Interface do Web Service - WSDL

- http://localhost:8080/axis/services/PdaService?WSDL

Consumir o Web Service c/ AXIS

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;

public class TestClient {
   public static void main(String [] args) {
       try {
           String endpoint =
                   "http://nagoya.apache.org:5049/axis/services/echo";

           Service  service = new Service();
           Call     call    = (Call) service.createCall();

           call.setTargetEndpointAddress( new java.net.URL(endpoint) );
           call.setOperationName(new QName("http://soapinterop.org/", "echoString"));

           String ret = (String) call.invoke( new Object[] { "Hello!" } );

           System.out.println("Sent 'Hello!', got '" + ret + "'");
       } catch (Exception e) {
           System.err.println(e.toString());
       }
   }
}
```

Ambiente .NET

Ferramentas

- Microsoft IIS (WinXP)
- Microsoft Visual Studio 2003

Criar um Web Service

- Instalar IIS
- Novo Projecto – ASP.NET Web Service
- Editar a classe Service1.asmx incluindo o código pretendido
- Os métodos a disponibilizar são precedidos pelo atributo [WebMethod]
- Build Solution

Interface do Web Service - WSDL

- **http://Host/Namespace/NomeServiço.asmx?WSDL**

Testar o Web Service

- Este ambiente permite testar imediatamente o serviço através de uma interface web disponibilizada directamente no endereço do serviço

  http://maquina:porta/Namespace/NomeServiço.asmx

Consumir o Web Service (cliente .NET)

- Novo Projecto – Windows Application
- Add Web Reference
  - Especificar endereço do WSDL do serviço, por ex:
    http://maquina:porta/Namespace/NomeServiço?WSDL
  - Serão criadas as classes os proxies locais que mapeam o serviço localmente
- Criar objecto do tipo ClasseService que é o proxy local para o serviço remoto
- Invocar os métodos desse objecto

Conclusões

- Os prototipos desenvolvidos nos vários ambientes permitiram testar a interoperabilidade na prática. Conclui-se que já é possível um bom nível de interoperabilidade se se tiver algum cuidado nos tipos de dados (objectos) usados.
- Existem funcionalidades que ainda não são standard não se garantindo portanto o correcto funcionamento entre diferentes implementações (ainda que diferentes implementações refiram implementar as mesmas versões desses standards)
- O ambiente Apache Axis dá ao programador um melhor controlo sobre algumas operações.
  - Ex: mapeamento de namespaces em packages. O ambiente .NET não permite isso.
  - EVentualmente mais flexibilidade no detalhe do processo de desenvolvimento
- O suporte em .NET visa uma integração mais homogénea no ambiente de programação WIndows .NET
  - Mais simplicidade para o desenvolvimento de serviços simples, imediatos
- O mapeamento dos dados é fulcral. No momento só é possível mapear tipos básicos e composiçoes de tipos básicos. Comportamentos dos objectos não são mapeaveis.
- ...