Web Service

- 1 Web Service概述
  - 1.1 什么是Web Service
  - 1.2 相关标准与技术
    - 1.2.1 SOAP
    - 1.2.2 WSDL
    - 1.2.3 UDDI
    - 1.2.4 技术实现原理
  - 1.3 Web Service中的主要角色
- 2 Web Service的分类
  - 2.1 Big Web Service
  - 2.2 RESTful Web Service
- 3 利用JAX-WS建立Big Web Service
  - 3.1 JAX-WS简述
  - 3.2 建立Big Web Service
  - 3.3 创建客户端
- 4 构建RESTful Web Service
  - 4.1 JAX-RS简述
  - 4.2 创建RESTful Web Service

1 Web Service概述

1.1 什么是Web Service

传统上，我们把计算机后台程序提供的功能称为服务。通俗地说，服务就是计算机可以提供的某种功能。

根据来源的不同，服务可以分成两种：本地服务、网络服务。

Web Service 的定义：Web Service 提供了一种网络服务的定义和获取方式，使得运行在不同的平台和框架下的软件应用程序之间可以通过网络进行互操作。

简单地说，Web Service 也是一种远程调用方式，只是因为它具有自描述、跨平台、开放性等特点，再加上它使用了 Web 上协议进行通信，使得 Web Service 的开发可以使用现有的基础设施，并具有和 Web 一样穿越防火墙的能力，使得 Web Service 得到了发展。

Web 服务实质上是一种部署在 Web 上的对象，采用标准的 XML 格式和开放的 Web 规范技术，向外提供其操作接口。它独立于服务所在的机器硬件、操作系统和开发语言，使得基于 Web 的应用程序达到松散耦合、面向组件和跨技术实现。

1.2 相关标准与技术

需要解决的问题：如何封装、如何描述、如何发现、如何访问。

相关技术：

1. 把功能变成方法——解决封装问题
2. WSDL——解决描述问题
3. UDDI——解决发现问题
4. SOAP——解决访问问题

1.2.1 SOAP

SOAP（Simple Object Access Protocol，简单对象访问协议）是一种轻量的、简单的、基于 XML 的协议，它被设计成在 Web 上交换结构化的信息。

SOAP = HTTP + XML数据。

SOAP 的组成：

1. Envelope——必须的部分，以 XML 的根元素出现
2. Headers——可选的
3. Body——必须的，包含要执行的服务器的方法和发送到服务器的数据

1.2.2 WSDL

WSDL（Web Service Description Language，Web Service 描述语言）是一种基于 XML 的服务描述文档，用于描述 Web Service 及其函数、参数和返回值。因为是基于 XML 的，所以 WSDL 既是机器可阅读的，又是人可阅读的。

WSDL 是 Web Service 客户端和服务器端都能理解的标准格式，其中描述的信息可以分为 what、where、how 等部分。

WSDL 文件保存在 Web 服务器上，通过一个 URL 地址就可以访问到它。客户端要调用一个 Web Service 服务之前，要知道该服务的 WSDL 文件的地址。

Web Service 服务提供商可以通过两种方式来暴露它的 WSDL 文件地址：

1. 将 WSDL 文件注册到 UDDI 服务器，以便被人查找。

2. 直接告诉客户端调用者。

1.2.3 UDDI

UDDI（Universal Description, Discovery, and Integration，统一描述、发现和集成）是一种目录服务，提供了 Web Service 的注册和搜索机制。

通过使用 UDDI 的发现服务，企业可以单独注册那些希望被别的企业发现的自身提供的 Web 服务。

企业可以通过 UDDI 商业注册中心的 Web 界面，将信息加入到 UDDI 的商业注册中心。

当一个企业在 UDDI 商业注册中心的一个实例中实施注册后，其注册信息会被自动复制到其他 UDDI 根节点，于是就能被任何希望发现这些 Web 服务的人所发现。

1.2.4 技术实现原理

Web Service 框架的本质就是一个大大的 Servlet。

从表面上看，Web Service 就是一个应用程序，它向外界暴露出一个能够通过 Web 进行调用的 API。

Web Service 是一种新的 Web 应用程序分支，它们是自包含、自表述、模块化的应用，可以在网络中被描述、发布、查找，以及通过 Web 来调用。

1.3 Web Service中的主要角色

Web Service 与普通 Web 程序的工作过程是不同的，需要三个角色参与该过程：

1. 服务提供者（Service Provider）：实现具体的服务，并通过服务代理将服务进行发布。
2. 服务请求者（Service Requestor）：向服务代理请求特定的服务。服务代理查询已发布的服务，为其寻找满足请求的服务，并向服务请求者返回满足条件的服务描述信息。
3. 服务代理（Service Broker）：提供服务的注册、发布和搜索机制。

2 Web Service的分类

2.1 Big Web Service

Big Web Service 使用遵循 SOAP 标准的 XML 格式的消息，又被称为基于 SOAP 的 Web Service。Big Web Service 通常是面向活动的。

Big Web Service 使用 WSDL 文档定义并描述特定于服务的操作。操作由特定于服务的消息交换组成，每一个操作都是一个可以执行的活动。那些正在被执行的操作所针对的内容通常是不相关的。

Big Web Service 定义了一整套复杂的标签，以描述调用的远程过程、参数、返回值和出错信息等。

Big Web Service 必须包含下面的组成部分：

1. 一个正式的合约，用于描述 Web Service 所提供的接口，包括消息、操作、绑定、Web Service 位置等细节信息。WSDL 可以完成这项工作，当然，也可以不使用 WSDL 描述服务，而直接处理 SOAP 消息。
2. 一个可以表达非功能需求的架构。通常，可以建立一个通用的词汇表来表述这种非功能需求，如事务、安全、可信、协作等。
3. 一个可以处理异步过程和调用的架构。

2.2 RESTful Web Service

RESTful Web Service 是遵循 REST 架构约束的 Web Service。

RESTful Web Service 是面向资源的服务，它通过 URI（Universal Resource Identifier，统一资源标识符）来识别和定位资源，并且针对这些资源而执行的操作是通过 HTTP 规范定义的。

核心操作：

1. GET：返回已标识资源的状态表示。
2. PUT：执行对已标识资源的一些特定于应用程序形式的更新。
3. POST：在已标识位置（URI）创建新资源。
4. DELETE：销毁已标识位置（URI）的资源。

一个 RESTful Web Service 必须遵循下面约束：

1. 应用程序状态和功能等网络上所有事物都被认为是资源。
2. 每个资源对应一个唯一的统一资源标识符（URI）。
3. 所有资源都共享统一的访问接口，以便在客户端和服务器之间传输状态。统一的访问接口使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。
4. 对资源的操作不会改变资源标识，所有的操作都是无状态的。

3 利用JAX-WS建立Big Web Service

3.1 JAX-WS简述

JAX-WS（Java API for XML Web Services）规范是一组 XML Web Services 的 JAVA API。

JAX-WS 2.0 是对 JAX-RPC 1.0 规范的扩展，是 JAX-RPC 1.1 的后续版本。JAX-WS 2.0 是面向 Java 5 的开发 Web Services 的最新编程标准，它提供了新的编程模型和对以往的 JAX-RPC 方式的 Web Services 进行了增强。

JAX-WS 2.0（JSR 224）是 Sun 新的 Web Services 协议栈，是一个完全基于标准的实现。

JAX-WS 用于建立 Big Web Service，使用 SOAP 协议。JAX-WS 自动处理所有 SOAP 相关的操作，不需要开发者编写任何处理 SOAP 消息的代码。

在服务器端，用户只需要通过 Java 语言定义远程调用所需要实现的接口，并提供相关的实现。通过调用 JAX-WS 的服务发布接口就可以将其发布为 Web Service 接口。

在客户端，用户可以通过 JAX-WS 的 API 创建一个代理来实现对于远程服务器端的调用。

3.2 建立Big Web Service

1. 在 Eclipse 中建立一个“dynamic web project”（动态 Web 工程）。在本例中，工程名为 BigWS_Hello。
2. 建立 Web 服务类。例如：

```
package javaee.bigws;

public class HelloWS
{
    public String sayHelloW(String name)
    {
        return "Hellow WebService! I am " + name + "!";
    }
}
```

3. 发布该类为 Web Service。
   （1）在 Web 服务类上点击右键，在弹出的菜单中选择 Web Services -> Create Web Service。
   （2）在弹出的对话框中 Web Service Type 选择“bottom up Java bean Web Service”，Service Implementation 选择上面编写的 Java 类（如`HelloWS`），并选中“publish the Web Service”，点击 next 。
   （3）在弹出的界面中选择供访问的方法（如`sayHelloW`），点击 next。
4. 查看运行结果：在网址 http://localhost:8080/BigWS_Hello/services 查看已经建立的 Web Service。

3.3 创建客户端

可以在 SE 环境、Web 环境和 EJB 环境中使用 Web Service。使用方法是，首先在需要访问 Big Web Service 的工程中使用 Web Service 描述文件（WSDL 文件）创建一个 Web Service 客户端，然后使用客户端来访问 Web Service。

以 SE 环境中使用 Web Service 为例，在 SE 环境中创建客户端的步骤为：

1. 在 Eclipse 中创建一个“Java Project”（普通 Java 工程）。
2. 在工程中创建一个“Web Service Client”。
3. 在弹出来的对话框中找到 Service definition，填写服务端的 URL 地址。最后点击“finish”会自动导入需要的包和生成代码文件。对于上述`HelloWS`类，生成的代码文件有 HelloWS.java、HelloWSProxy.java、HelloWSService.java、HelloWSServiceLocator.java、HelloWSSoapBindingStub.java。
4. 编写客户端类。例如：

```
public class WSClient
{
    public static void main(String[] args)
    {
        try
        {
            HelloWSProxy proxy = new HelloWSProxy();
            proxy.setEndpoint("http://localhost:8080/BigWS_Hello/services/HelloWS");
            HelloWS service = proxy.getHelloWS();
            System.out.println(service.sayHelloW("a teacher"));
        }
        catch (RemoteException e)
        {
            e.printStackTrace();
        }
    }
}
```

4 构建RESTful Web Service

4.1 JAX-RS简述

REST 的全称为 Representational State Transfer，翻译为“表现层状态转化”。它不是一种框架，也不是一种规范，而是一种网络应用程序的设计风格和开发方式，用来降低开发的复杂性，提高系统的可伸缩性。

所谓 RESTful 资源指的是网络上的一个实体，或者说是网络上的一个具体信息。可以用一个 URI 指向它，每种资源对应一个特定的 URI。要获取这个资源，访问它的 URI 就可以，因此 URI 就成了每

一个资源的地址或独一无二的识别符。

在基于 REST 的系统中，对资源的管理、定位或者其他操作，URI 是不变的。

资源是一种信息实体，它可以有多种外在表现形式。我们把资源具体呈现出来的形式，叫做它的“表现层”。URI 只代表资源的实体，不代表它的形式。

访问一个网站，就代表了客户端和服务器的一个互动过程。在这个过程中，势必涉及到数据和状态的变化。HTTP 协议是一个无状态协议，这意味着所有的状态都保存在服务器端。因此，如果客户端想要操作服务器，必须通过某种手段，让服务器端发生“状态转化”（State Transfer）。而这种转化是建立在表现层之上的，所以就是“表现层状态转化”。

REST 的设计原则：

1. 网络上的所有事物都被抽象为资源。
2. 网络上的资源都有一个唯一标识符，即 URI。
3. 对资源的操作通过统一的通用接口规范来访问。
4. 对资源的访问不会改变它的唯一标识符，即 URI 不变。
5. 所有的操作都是无状态的。

JAX-RS 是一套 Java API，用于开发 REST 架构的应用。JAX-RS 使用标注来简化 RESTful Web Service 的开发。开发者可以通过标注来定义资源，以及作用于资源上的操作。

JAX-RS 标注是运行时的标注，会使用反射机制在运行时创建一些辅助类和软件部件（Artifacts）。

JAX-RS的标注有：

1. `@Path`：标注资源类或方法的相对路径
2. `@GET`、`@PUT`、`@POST`、`@DELETE`：标注方法处理的 HTTP 请求的类型
3. `@Produces`：标注返回的 MIME 媒体类型
4. `@Consumes`：标注可接受请求的 MIME 媒体类型
5. `@PathParam`、`@QueryParam`、`@HeaderParam`、`@CookieParam`、`@MatrixParam`、`@FormParam`：分别标注方法的参数来自于 HTTP 请求的不同位置。例如`@PathParam`来自于 URL 的路径，`@QueryParam`来自于 URL 的查询参数，`@HeaderParam`来自于 HTTP 请求的头信息，`@CookieParam`来自于 HTTP 请求的 Cookie。

4.2 创建RESTful Web Service

1. 在 Eclipse 中创建一个“dynamic web project”（动态 Web 工程）。在本例中，工程名为 RESTWS_Hello。
2. 创建 Web 服务类。例如：

```java
package javaee.restws.hello;

@Path("/helloworld") // @Path 表示访问这个资源的路径
public class HelloRESTWS
{
    @GET // @GET 表示响应 HTTP 的 get 方法
    @Produces("text/plain") // @Produces 标注返回的 MIME 媒体类型
    public String sayHello()
    {
        return "Hello World";
    }
}
```

3. 发布。在 JBoss AS7 中，发布 RESTful Web Service 需要重新编写一个`javax.ws.rs.core.Application`的子类，如下所示：

```java
package javaee.restws.hello;

public class MyRESTApplication extends Application
{
    private Set<Object> singletons = new HashSet<Object>();
    private Set<Class<?>> empty = new HashSet<Class<?>>();

    public MyRESTApplication()
    {
        singletons.add(new HelloRESTWS());
    }

    @Override
    public Set<Class<?>> getClasses()
    {
        return empty;
    }

    @Override
    public Set<Object> getSingletons()
    {
        return singletons;
    }
}
```

4. 配置 web.xml 文件，如下所示：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         id="WebApp_ID" version="3.0">
    <display-name>RESTWS_Hello</display-name>
    <context-param>
        <param-name>javax.ws.rs.Application</param-name>
        <param-value>javaee.restws.hello.MyRESTApplication</param-value>
    </context-param>

    <listener>
        <listener-class>
            org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>Resteasy</servlet-name>
        <servlet-class>
            org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
        </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Resteasy</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

创建完成后，将工程部署到 JBoss 上，即可通过 http://localhost:8080/RESTWS_Hello/helloworld 访问该服务。