Due: Saturday March 1 2008, 12:00 midnight

PROJECT 1: DESIGNING AND DEPLOYING A SERVICE-BASED DISTRIBUTED SYSTEMS

1. Purpose:

1. To understand the fundamentals of a distributed systems.
2. To understand the components, core technologies, architecture and protocols that enable a Web Services-based distributed system.
3. To design and implement a Web Service.
4. To understand the process of preparing and deploying a remote service.

2. Preparation before lab:

1. Read and study the Web Services architecture and associated protocols. Go through the Web Services tutorial offered by Sun Microsystems. Understand RMI. Web Resources:
    a. http://www.w3.org/2002/ws/
    b. http://java.sun.com/webservices/docs/2.0/tutorial/doc/
    c. http://java.sun.com/j2se/1.5.0/docs/guide/rmi/index.html

2. Learn how to use the XML-based build tool Ant.
    a. http://ant.apache.org/manual/index.html

3. Learn the fundamental of a relational database and designing relational tables. You may use MS Access, Oracle 9i (UB CSE Department's) or any small footprint file-based databases that come bundled with the IDE you may use. Learn to use the application interface to a relational database using embedded SQL and JDBC.
    a. http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html
    b. http://www.cse.buffalo.edu/local/Consulting/Software/Databases/Oracle/

4. Choose an IDE (Integrated Development Environment) to work with. You may choose to work with Eclispse or Netbeans. Instructions and demos during lecture and recitations will be on Netbeans IDE.
    a. http://www.netbeans.org (choose version 6 and 5.5.1). This is available for most operating systems including Mac OS and Linux.
    b. You may work on your lap top or on project space that will be allocated for you on CSE machines. This project space is accessible from *pollux* (Solaris) as well as *nickelback* (Linux) CSE machines.

5. Finally, you must have a clear understanding of a client-server system operation.

3. Web Services Technology:

Web Services technology provides a standard means of building a distributed system over the Internet. In simple terms, it provides a means for a sophisticated remote procedure call. The sophistication arises out of the elegant mechanisms it supports for enabling:

1. Various transparencies (platform, language, and hardware) using open standards,
2. Application to application data exchange and interoperability, and
3. Creation of composite web services from a set of simple web services.

The significant difference between regular HTTP-based technologies and Web Services is the standardization realized through the XML and SOAP (SOAP and XML over HTTP). Web Services Definition Language (WSDL) is an important standard that allows for consistent definition of services. All these advantages make Web Services technology ideally suited for large-scale enterprise level application integration.
Web Services specifications are defined by the World Wide Web Consortium (W3C). Many vendors including Sun Microsystems, Oracle and Microsoft (.net) have frameworks for building and deploying Web Services.

4. Assignment:

Build a multi-tier distributed system comprising two major sub-systems

1. An **RMI** and **simple data acquisition system** and
2. A **Web Services** based **web application** that processes and serves the data collected.

The two sub-systems are *loosely coupled* via a database. The block diagram of the system you will implement is given in Figure 1. The RMI part of the project is adapted from the Weather service problem described in the Fourth Edition of '*Java: How to Program?*' *by Deitel and Deitel*.

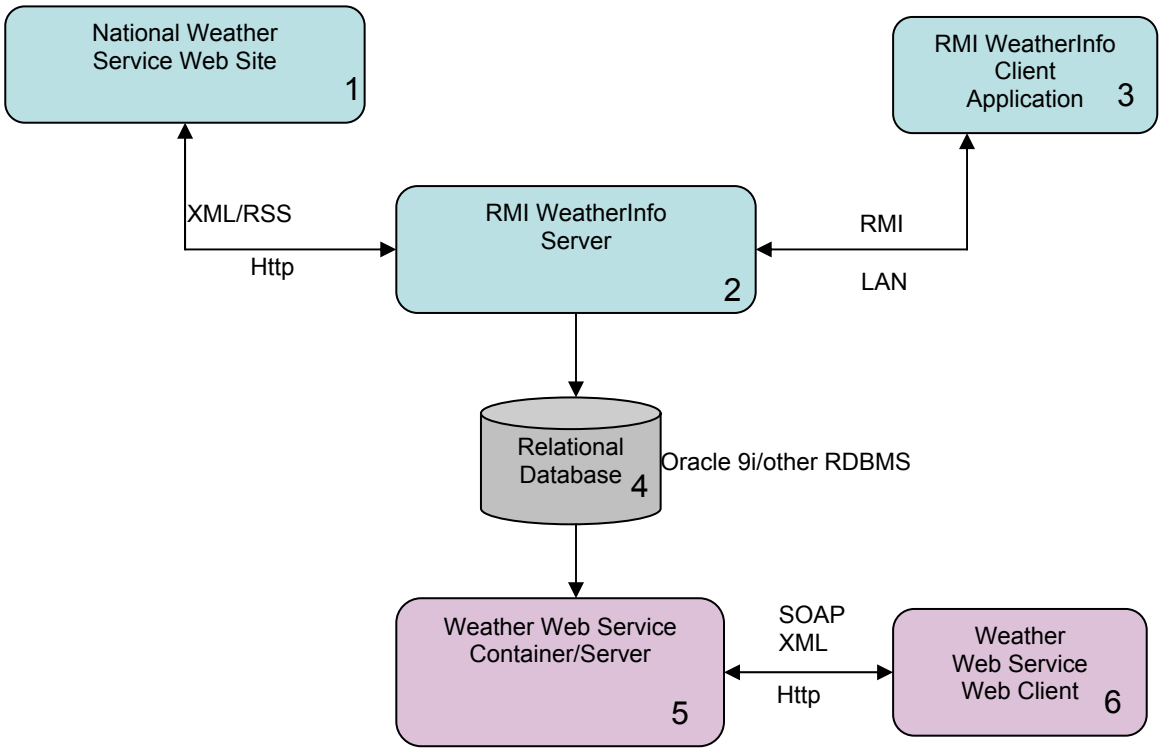

Figure 1: System Model of the Weather Service Distributed System

The national weather bureau updates the weather conditions at various cities on its web site (*box 1 in Figure 1*). You may either use XML feed from weather.com or RSS (XML) feed from weather.gov.

The RMI Server (*box 2*) streams in the page and parses it for the relevant data and stores it in a persistent storage. (The details of the RMI and existing code base and a simple frame work

were discussed during the lecture and are available in the course web page. You will have to port that code to Java 5 version.

The persistent storage in the sample code is a simple file and the data stored is just the weather data for one day. You are required to update the code to accumulate the data for a period of over **at least 1 week** (or *any 7 days*). You may design the server to be a persistent one that automatically collects the data once every day and persists the data in a database. The data collected should be stored in a **relational database** (*box 4*) on *Oracle 9i*. The daily data is served to an RMI client (*box 3*) which simply renders the ASCII data provided by the weather bureau by transforming it using visually appealing graphics [The RMI client functionality is already provided with the sample code and was demonstrated in class].

In the **Web Services** part of the system, the data collected in the database will be processed by the server (*box 5*) for such information as average temperature for a given city, and the temperature on a particular day for a city. The city can be specified by either the text name or zip code. The Web Services client (*box 6*) will be able to query the server for various information related to the data collected. Your task is to design and implement the complete Web Services-based system indicated by *boxes 4, 5* and *6* of *Figure 1*, and study the operation of the integrated system depicted in Figure 1.

5. Project Implementation Details and Steps:

1. **Get used to building *client-server* systems**:
   - When you implement a simple *client side* application program there are just two steps involved: compile and execute the code.
   - In a *client-server* system, you will have to take care of the *server side* as well as the *client side*. On the *server side*, you will compile the code, generate stubs or proxies using special compilers, deploy the service, register and publicize the service for the clients to use. On the *client side* you will prepare the client code with appropriate stubs, and during execution lookup the service needed and use it.
   - To understand the processes study the RMI-based system code and implementation. Deploy it and make sure it works and you understand the various operations. You will notice that besides simple compile and execute, configuration and deployment of a service are important issues to be reckoned with. Use the latest version of RMI.

2. **Working with the *relational database* and embedded *SQL*:**
   - In this project you will store the data in a relational table and access it using SQL statements embedded in the Java programming language (using JDBC). Work on a simple java program to refresh your knowledge about accessing the Oracle/other relational database.

3. **Building systems using build tools such as Ant:**
   - In order to tackle complexities in configuration and deploying server-side applications, you will need to use special build tools. Apache Ant is an XML-based build tool similar to "make" utility that most of you are familiar with. Work on simple files to familiarize yourself with the Ant build tool. If you use an IDE the build is automatically configured for you. Even in this case it is good to understand the nuances of build operation.

4. **Study and understand the Web Services building and deployment details:**
   - For the Web Services part of this project, we will use the Sun Microsystems implementation of the Web Services specification.
   - Most IDEs come with servers to deploy the web services. **Please try the tutorials on RMI and Web services on the IDE you have chosen before you start working on the project.** If you are using Solaris or Linux you will have to set up Tomcat server in your project spaces.

5. **Design, implement and test your weather Web Service:**
   - Using the example given in the Step 4 above design the Web Service for dispensing and answering user queries about the weather information of various cities. This is expected to be the most time consuming part of the project due to the novelty of the topic. (Note: Explore using JUnit for testing the various modules).

6. **Deploy the integrated system:**
   - The various components listed above should have been deployed and tested individually. In this step you will run the entire integrated system. The RMI part can be scheduled to acquire data once a day to update the database that will be used by the Web Service part.

6. Project Deliverables:

1. A modified **RMI Server** that periodically schedules and updates a relational database with weather information for a set of cities.

2. An **RMI Client** (essentially the same client that was given to you) that can still connect to the RMI Server, obtain, and display latest weather information for a set of cities.

3. A **Web Service** that is able to query the relational database for different information. Requirements include (but need not be limited to):
   a. Weather for a particular city on a particular day
   b. Average weather for a particular city over a given period
   c. Ability to convert between English and Metric weather units
   d. Ability to directly invoke the RMI Server from Web Service to get **current** weather (as opposed to weather stored in the database)

4. A **Web** based **client** created using JSP pages that allows a user to use the various services offered by your web service.

7. Submission Details:

Create a compressed deployable distribution of your project. Submit it online. You will have demo the code to the TAs after the due date. Demo is mandatory.
More details will be provided as the deadline near. There is a lot of new stuff you will have to pick up. You will definitely need the whole month. So,

- Start Early,
- Read-up and make sure you understand what your code does, and
- Do not forget to have fun!