Overview of Web Technologies

Course Code: CMPE 296U
Course Name: Web-Based Application Systems and Tools

Instructor: Jerry Gao, Ph.D.

San Jose State University

Fall, 1999

Copyright@1999. Jerry Gao, Ph.D
Introduction to Web Technology

- Common Gateway Interface (CGI)
- JavaScript
- Java Servlet
- JDBC
- RMI
- Swing
About Common Gateway Interface (CGI)

What is the Common Gateway Interface (CGI)?

The CGI connects Web servers to external applications. CGI can do two things:

- It can gather information sent from a web browser to a web server, and make the information available to an external program.

- CGI can send the output of a program to a Web browser that request it.

Advantages of CGI:

- Platform independence: Most web servers support CGI, including
  Unix: Apache, Netscape, NCSA, and CERN
  Windows NT: Netscape, Microsoft IIS, and O’Reilly WebSite
  Macintosh: WebStar

- Language independence: (Perl, TCL, C, C++ Visual Basic, AppletScript, Java)

- Scalability: The simplicity of the CGI interface means that it is extremely scalable.
About Common Gateway Interface (CGI)

How a CGI request is processed?

1. User fills out the form in the browser
2. Form submitted over the Internet
3. Server sends data to CGI application
4. CGI processes the data and generates the HTML page
5. Server sends the page to the browser

Copyright@1999. Jerry Gao, Ph.D
About Common Gateway Interface (CGI)

CGI-based Architecture:

Web browser -> Internet -> Web Server

Web browser

Web browser

Web browser

application

script

script

script

application
About Common Gateway Interface (CGI)

The limitation and problems of CGI

Problems:

- Performance
  Every time a user requests a CGI script, the server must launch the CGI program, which takes processor time.

  When the CGI program is written in an interpreted language like Perl, then the program must run the entire Perl interpreter, and compile the program before it can be run, which takes even more processor time.

  For busy web sites running complex applications, the performance problem become critical issue.

- It is free. --> not easy to sell the web server products with free software.

- Not good to interact with database.(connecting issue)

- Not flexible or powerful to generate dynamic HTML pages.
About Java Servlet

What is Java Servlets?

A servlet is a Java component that can be plugged into a Java-enabled web server to provide custom services. These services include:
- new features
- runtime changes to content
- runtime changes to presentation
- new standard protocols (such as FTP)
- new custom protocols

Objectives:

- Servlets are designed to work within a request/response processing model. The requests can come in the form of an HTTP, URL, FTP or custom protocol.

Protocol Support:

The Servlet API provides a tight link between a server and servlets. This allows servlets to add new protocol support to a server.

Any protocol that follows a request/response computing model can be implemented by a servlet. Including: HTTP, SMTP, POP, FTP
About Java Servlet

Advantages and Benefits:

- It provide a standard approach to extending server functionality without the limitations of CGI-based or server-specific approaches.

- Java Servlets are the perfect replacement for CGI-bin scripts because of:
  - much less resource-intensive
  - more nimble or dynamic
  - better performance on JavaWeb Server
- They are fast, safe, reliable, and 100% pure Java.

Applications:

Java Servlets with Applets:

Work hand in hand with applets, providing a high degree of interactivity and dynamic updating.

E.g. a scheduling applet, could interact with a Java Servlet to provide a common calendar for a group of users. The applet provide interactive features, such as a scrolling calendar view, that are updated by the Java Servlet dynamically.
About Java Servlet

Applications: (Con’t)

Java Servlets with Applets:

Work hand in hand with applets, providing a high degree of interactivity and dynamic updating.

E.g. a scheduling applet, could interact with a Java Servlet to provide a common calendar for a group of users. The applet provide interactive features, such as a scrolling calendar view, that are updated by the Java Servlet dynamically.

Java Servlets for HTML Generation:

- The most common for Java Servlets is to accept form input and generate HTML dynamically.

- Servlets can be written to process HTML pages and customize them as they are sent to a client.
About Java Servlet

Applications: (Con’t)

Java Servlets for Middle Tiers:

- The most common use of Java Servlets --> as part of middle tiers in enterprise networks, connecting SQL DB via JDBC.

One advantage of middle tier processing is simply connection management. Other roles of middle tier include:

- Business rule enforcement
- Transaction management
- Mapping clients to redundant set of servers
- Supporting different types of clients such as pure HTML and Java capable clients
Introduction to JDBC

Basic Information about JDBC:

1. In the summer of 1996, Sun released the first version of the JDBC kit. This package lets Java programmers connect to a database, query it, or update it using SQL.

2. JDBC is an SQL-level API for Java, which is a universal DB access API. It allows you to embed SQL statements as arguments to methods in JDBC interfaces in a DB-independent fashion.

3. JDBC requires that the database being used support ANSI SQL-2 as the query language.

4. JDBC requires database vendors to furnish a run-time implementation of its interfaces. These implementations route your SQL calls to the DB in the proprietary fashion it recognizes.

Essential advantage over other DB programming environments is ->:

- Programs developed with Java and the JDBC are platform independent and vendor independent.
- The same Java DB program can run on a PC, a workstation, or a Java powered terminal.
- Data in one DB can be moved to another DB easily without changing DB access program..

Experts believe that:
- Java and JDBC will eventually replace proprietary database languages (such as Borland’s PAL) used by various vendors (such as Powersoft, Oracle, and Microsoft) for accessing DBs.

©1999. Jerry Gao, Ph.D
JDBC and Database Technologies

- Database Technologies:

  A Java client/server application can make use of one of three major database architectures:
  - Relational DB, most today’s DB applications use relational DBs.
  - Object database
  - Object-relational database

- JDBC API heavily biased towards relational DBs and their standard query language, SQL. In choosing the object-oriented reality of Java, we need to create a translation layer that maps the relational world into our object world.

- While JDBC provides us with access to relational databases, it leaves the issue of object-to-relational mapping to you.

- Object databases do not attempt to separate object data from behavior. Instead, we think of an object database as a permanent store of objects with which your applications can interface.

- JDBC so tightly bound to SQL, it is difficult to create JDBC drivers to run against an object DB. JavaSoft, in cooperation with the ODMG (Object Database Management Group), is working on a specification for a Java object database API.

©1999. Jerry Gao, Ph.D
What Java Adds to DB Programming?

- While the marriage of Java and database programming is beneficial to Java programmers. Java also helps database programmers.

- Specifically, Java provides database programmers with the following features they have traditionally lacked:
  - Easy object to relational mapping
  - Database independence
  - Distributed computing

- Java provides an alternative to these two tools that frees you from the proprietary interfaces associated with database programming.

- A Java DB application does not care what its database engine is. No matter how many times the DB engine changes, the application itself need never change.
The Design of JDBC

JDBC History:
- in November 1995, JavaSoft started working on extending Java to deal with SQL access to databases --> any random database.
- Later they found that this is an impossible task. Because there are too many databases with different protocols.
- Many DB vendors were all in favor of JavaSoft providing a standard network protocol for database access.

The main design goals of JDBC:
- JDBC should be an SQL-level API.
- JDBC should capitalize on the experience of existing database APIs.
- JDBC should be simple.

- An SQL-level API means that JDBC allows us to construct SQL statements and embed them inside Java API calls. In short, you are basically using SQL. But JDBC lets you smoothly translate between the world of the DB and the world of the Java application.

- JDBC is influenced by ODBC to provide a standard, common, and universal DB access for Java application.

- JDBC is influenced by other DB programming APIs, such as X/OPEN SQL call level interface.
The Structure of JDBC

The JDBC consists of two layers:

- **JDBC API**, which communicates with the JDBC manager driver API, sending it the various SQL statements.

- **JDBC manager driver**, which communicates with the various third-party drivers that actually connect to the database and return the information from the query or perform the action specified by the query.

The ultimate goal of the JDBC is to make possible the following:

- Programmers can write applications in Java to access any DB, using standard SQL statements - or even its extension.

- DB vendors and DB tool vendors can supply the low-level drivers. Thus, they can optimize their drivers for their specific situation.
The Structure of JDBC

The classes and interfaces of java.sql, the JDBC API package:

- Driver
- Statement
- Connection
- ResultSet
- ResultSetMetaData
- DatabaseMetaData
  - PreparedStatement
  - CallableStatement
  - Java.lang.Object
    - java.util.Date
      - Date
    - DriverManager
    - DriverPropertyInfo
    - Types
      - Time
      - Timestamp

Copyright@1999. Jerry Gao, Ph.D
Applications of JDBC

We use the JDBC-enhanced version of Java in both applications and applets.

- When the version is used in an applet, all the normal security restrictions apply.
- The JDBC continues to assume that all Java applets are untrusted.

1. Applets use JDBC to open a DB connection from the server from which they are downloaded.
   - Applets can not use local information.
   - Although the JDBC extensions of the Java security model allow one to download a JDBC driver and register it with JDBC device manager on the server, that driver can be used only for connections from the same server the applet came from.
   - When signed Java applets become possible, this restriction could be loosened.
   - You can use JDBC with applets, but you must manage the server carefully.

2. Applications have complete freedom, They can give the application total access to files and remote servers.

3. Three-tier model on JDBC + RMI.
   - Use Java (application or applet) as the client software
   - Use Java to generate a middleware.
   - Use RMI to communicate between the client and the middle layer.
   - Use JDBC as an interface between a DB server and the middleware.
Topic: Overview of Web Technologies

Browser

Java Application

RMI

Application/Object Server

ObjectPersistence

OP to table/column

OP to file

OP to OODBMS

JDBC

O/S

undefined interface

ODMG working on spec.

Application/Object Server

JDBC Connection Server

mSQL

Oracle

Informix

OODBMS

Database Server

Class Library

Copyright@1999. Jerry Gao, Ph.D
**Introductions to Remote Objects**

**Basic Concepts: Remote Objects**

Locally collecting information on a client computer and sending the request information across the Net to a server. The server processes the request, and will in turn to send back the response, or provide the service, and return the results.

**RMI** - Remote Method Invocation for communicating between two machines running Java Virtual Machines.

Two ways to transmit information between a client and a server:

- Use a socket connection to send byte streams between the customer and the vendor computers.
- Use JDBC to make DB queries and updates.

---

**Transmitting objects between client and server**

Copyright@1999. Jerry Gao, Ph.D
What is Remote Method Invocation?

*RMI* is the best solution for communication between Java programs on different hosts.

RMI is a facility which allows Java programs to call certain methods on a remote server.

RMI does not require either the client or the server to run an HTTP server.

We can look at RMI as a mechanism for defining custom protocols and servers with minimal effort.

RMI lets Java objects on different hosts communicate with each other.

RMI mechanism was invented by JavaSoft for Java-to-Java communication.

RMI is useful only for communication between Java objects.

It does not currently use a standard transmission protocol such as IIOP.

Its basic concepts of distributed computing are similar for RMI and CORBA.

Java implements remote objects by supplying a transport layer that handles the data encoding and the transmission and call protocols. The major benefits:

- You need not worry about managing streams of bytes, you can send messages to Java objects of any type.
- You can transport objects across a network.
- You can invoke method calls on objects that reside on another computer without having to move those objects to the machine making the method call.

Limitation of RMI:

- It only can be used to communicate with Java programs. In other words, call methods must written in Java.

A Programmer’s view:

From the programmer’s perspective, remote objects and methods work just like the local objects and methods you’re accustomed to.
Stubs and Skeletons

Stub:

When a client code wants to invoke a remote method on a remote object, it actually calls a regular Java method that is encapsulated in a surrogate object called, a stub.

The stub resides on the client machine, not the server machine. It packages the parameters used in the remote method as a block of bytes in a device-independent encoding.

The process of encoding the parameters into a format suitable for transporting them between objects running different processes or across the Net is called parameter marshalling.

To sum up: the stub method on the client builds an information block that consists of:
- An identifier of the remote object to be used.
- An operation number, describing the method to be called.
- The marshalled parameters.

The stub then sends the information to the server.
**Stubs and Skeletons**

**Skeleton:**

On the server side, a skeleton object makes sense out of the information contained in the packet and passes that information to the actual object executing the remote method.

The skeleton performs five actions for every remote method call:
- It unmarshals the parameters.
- It calls the desirable method on the remote object that lies on the server.
- It captures the return or exception of the call on the server.
- It marshals that value.
- It sends a package consisting of the value in the marshalled from back to the stub on the client.
RMI and Others

**Traditional Networking:**
Networking has been concerned with two fundamental applications:

- Moving files and data between hosts. This is handled by FTP, SMTP, HTTP, NFS, and many others.
- Allowing one host to run programs on another host. This refers to telnet, rlogin, remote procedure calls (RPC) and a lot of middleware.

**RPC (Remote Procedure Call) VS. RMI:**
- An older technology Sun developed that does much the same thing as RMI.
- RPC is language and processor independent. RMI is processor independent, but Java-based.
- RPC has to convert arguments between architectures, so that each computer can use its native data types, while RMI can send objects.

**RMI VS. CORBA:**
- CORBA is the most general solution for distributed objects. Because CORBA lets objects written in different languages communicate with each other.
- RMI only can be used to communicate between Java distributed objects. If you have some programs written in C++, then RMI is not the solution for you. However, CORBA is.

**Server’s Remote objects:**
- A remote object lives on a server. Each remote object implements a remote interface that specifies which of its methods can be invoked by clients.
- Clients can invoke the methods of the remote object almost exactly as they invoke local methods.

Copyright ©1999. Jerry Gao, Ph.D
RMI and Security

The prospect of remote clients invoking methods in your host’s objects raises a lot of security issues.

Just as an applet host can limit the activities of an applet, a host that allows RMI can limit what the remote clients can do:
- The activities that a remote object can perform are limited in much the same way as an applet’s activity is limited.
- A SecurityManager object checks all operations to make sure they’re allowed by the server.
- Custom security managers can be defined for specific applications.
- Public key authentication can be used to verify a user’s identity, and allow different users different levels of access to a remote object.

For example, the general public may be allowed to query a database, but not update it. Users from inside a company might be allowed to query and update the database.
The RMI Layer Model

Object parameter passing between distributed objects are different from objects in the same machine. There are three ways to pass or return results from remote methods, depending on the type of data being passed.

- **Primitive types (int) are passed by value.**
- **References to remote objects are passed as a remote reference that allows the recipient to invoke methods on the remote object.**
- **Objects that do not implement the Remote interface are passed by values.**

Objects that don’t allow themselves to be serialized can not be passed to remote methods.

Remote objects run on the server but can be called by objects running on the client. Non-remote, serializable objects run on the client system.

**Stubs:**
The stub doesn’t need to be compiled into the client, it can be downloaded at runtime. A client invokes a remote method using a stub. The stub is a special object that implements the remote interfaces of the remote object. When the stub is invoked, it passes the invocation into the remote reference layer.
The RMI Layer Model

- Server Program
- Skeleton
- Remote Reference Layer
- Transport Layer

Logic Path

- Client Program
- Stub
- Remote Reference Layer
- Transport Layer

Server

Client

Parameters

Client Object

Method invocation on client

Marshalled parameters

Return value

Server Object

Marshalled return value

Method invocation on Server

Copyright@1999. Jerry Gao, Ph.D
The RMI Layer Model

The remote reference layer:
It carries out a specific remote reference protocol that is independent of the specific client stubs and server skeletons. The remote reference layer is responsible for understanding what a particular remote reference means.

In essence, the remote reference layer translates the local reference to the stub into a remote reference to the object on the server. Then it passes the data to the transport layer.

The transport layer:
The transport layer handles the actual movement of data across the Internet. It provides: connection setup, connection management, and tracking and dispatching of remote objects.

On the server side, the transport layer listens for incoming connect or data. When it receives an invocation, it forwards the invocation to the remote references sent by the client into references for the local virtual machine. Then, it passes the request to the skeleton.

The skeleton layer:
It reads the arguments, and passes the data to the server program, which makes the actual call. If the method call returns a value, that value is sent down through the skeleton, remote reference, and transport layers on the server side, across the Internet, and then up through the transport, remote reference, and stub layers on the client side.