Chapter 9: Application Design and Development

Chapter 9: Application Design and Development

- Application Programs and User Interfaces
- Web Fundamentals
- Servlets and JSP
- Application Architectures
- Rapid Application Development
- Application Performance
- Application Security
- Encryption and Its Applications

Application Programs and User Interfaces

- Most database users do *not* use a query language like SQL
- An application program acts as the intermediary between users and the database
  - Applications split into
    - front-end
    - middle layer
    - backend
- Front-end: user interface
  - Forms
  - Graphical user interfaces
  - Many interfaces are Web-based

Application Architecture Evolution

- Three distinct era's of application architecture
  - mainframe (1960's and 70's)
  - personal computer era (1980's)
  - We era (1990's onwards)

(a) Mainframe Era (b) Personal Computer Era (c) Web era

Web Interface

- Web browsers have become the de-facto standard user interface to databases
  - Enable large numbers of users to access databases from anywhere
  - Avoid the need for downloading/installing specialized code, while providing a good graphical user interface
    - Javascript, Flash and other scripting languages run in browser, but are downloaded transparently
  - Examples: banks, airline and rental car reservations, university course registration and grading, an so on.

The World Wide Web

- The Web is a distributed information system based on hypertext.
- Most Web documents are hypertext documents formatted via the HyperText Markup Language (HTML)
- HTML documents contain
  - text along with font specifications, and other formatting instructions
  - hypertext links to other documents, which can be associated with regions of the text.
  - forms, enabling users to enter data which can then be sent back to the Web server

Uniform Resources Locators

- In the Web, functionality of pointers is provided by Uniform Resource Locators (URLs).
- URL example:

  http://www.acm.org/sigmod

  - The first part indicates how the document is to be accessed
    - "http" indicates that the document is to be accessed using the Hyper Text Transfer Protocol.
  - The second part gives the unique name of a machine on the Internet.
  - The rest of the URL identifies the document within the machine.
- The local identification can be:
  - The path name of a file on the machine, or
  - An identifier (path name) of a program, plus arguments to be passed to the program
    - E.g., http://www.google.com/search?q=silberschatz

HTML and HTTP

- HTML provides formatting, hypertext link, and image display features
  - including tables, stylesheets (to alter default formatting), etc.
- HTML also provides input features
    - Select from a set of options
      - Pop-up menus, radio buttons, check lists
    - Enter values
      - Text boxes
  - Filled in input sent back to the server, to be acted upon by an executable at the server
- HyperText Transfer Protocol (HTTP) used for communication with the Web server

Sample HTML Source Text

```
<html>
<body>
  <table border>
     <tr> <th>ID</th> <th>Name</th> <th>Department</th> </tr>
     <tr> <td>00128</td> <td>Zhang</td> <td>Comp. Sci.</td> </tr>
     ….
  </table>
   <form action="PersonQuery" method=get>
     Search for:
       <select name="persontype">
          <option value="student" selected>Student </option>
          <option value="instructor"> Instructor </option>
       </select> <br>
     Name: <input type=text size=20 name="name">
     <input type=submit value="submit">
  </form>
</body> </html>
```

Display of Sample HTML Source

ID	Name	Department
00128	Zhang	Comp. Sci.
12345	Shankar	Comp. Sci.
19991	Brandt	History

Search for: Student

Name:

submit

Web Servers

- A Web server can easily serve as a front end to a variety of information services.
- The document name in a URL may identify an executable program, that, when run, generates a HTML document.
  - When an HTTP server receives a request for such a document, it executes the program, and sends back the HTML document that is generated.
  - The Web client can pass extra arguments with the name of the document.
- To install a new service on the Web, one simply needs to create and install an executable that provides that service.
  - The Web browser provides a graphical user interface to the information service.
- Common Gateway Interface (CGI): a standard interface between web and application server

Three-Layer Web Architecture

Two-Layer Web Architecture

- Multiple levels of indirection have overheads
  - Alternative: two-layer architecture

HTTP and Sessions

- The HTTP protocol is **connectionless**
  - That is, once the server replies to a request, the server closes the connection with the client, and forgets all about the request
  - In contrast, Unix logins, and JDBC/ODBC connections stay connected until the client disconnects
    - retaining user authentication and other information
  - Motivation: reduces load on server
    - operating systems have tight limits on number of open connections on a machine
- Information services need session information
  - E.g., user authentication should be done only once per session
- Solution: use a **cookie**

Sessions and Cookies

- A **cookie** is a small piece of text containing identifying information
  - Sent by server to browser
    - Sent on first interaction, to identify session
  - Sent by browser to the server that created the cookie on further interactions
    - part of the HTTP protocol
  - Server saves information about cookies it issued, and can use it when serving a request
    - E.g., authentication information, and user preferences
- Cookies can be stored permanently or for a limited time

Servlets

- Java Servlet specification defines an API for communication between the Web/application server and application program running in the server
  - E.g., methods to get parameter values from Web forms, and to send HTML text back to client
- Application program (also called a servlet) is loaded into the server
  - Each request spawns a new thread in the server
    - thread is closed once the request is serviced

Example Servlet Code

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PersonQueryServlet extends HttpServlet {
  public void doGet (HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HEAD><TITLE> Query Result</TITLE></HEAD>");
    out.println("<BODY>");
      ….. BODY OF SERVLET (next slide) …
    out.println("</BODY>");
    out.close();
  }
}
```

Example Servlet Code

```
String persontype = request.getParameter("persontype");
String number = request.getParameter("name");
if(persontype.equals("student")) {
    ... code to find students with the specified name ...
    ... using JDBC to communicate with the database ..
    out.println("<table BORDER COLS=3>");
    out.println(" <tr> <td>ID</td> <td>Name: </td>" + " <td>Department</td> </tr>");
    for(... each result ...){
        ... retrieve ID, name and dept name
        ... into variables ID, name and deptname
        out.println("<tr> <td>" + ID + "</td>" + "<td>" + name + "</td>" + "<td>" + deptname
                + "</td></tr>");
    };
    out.println("</table>");
}
else {
    ... as above, but for instructors ...
}
```

Servlet Sessions

- Servlet API supports handling of sessions
  - Sets a cookie on first interaction with browser, and uses it to identify session on further interactions
- To check if session is already active:
  - if (request.getSession(false) == true)
    - .. then existing session
    - else .. redirect to authentication page
  - authentication page
    - check login/password
    - request.getSession(true): creates new session
- Store/retrieve attribute value pairs for a particular session
  - session.setAttribute("userid", userid)
  - session.getAttribute("userid")

Servlet Support

- Servlets run inside application servers such as
  - Apache Tomcat, Glassfish, JBoss
  - BEA Weblogic, IBM WebSphere and Oracle Application Servers
- Application servers support
  - deployment and monitoring of servlets
  - Java 2 Enterprise Edition (J2EE) platform supporting objects, parallel processing across multiple application servers, etc

Server-Side Scripting

- Server-side scripting simplifies the task of connecting a database to the Web
  - Define an HTML document with embedded executable code/SQL queries.
  - Input values from HTML forms can be used directly in the embedded code/SQL queries.
  - When the document is requested, the Web server executes the embedded code/SQL queries to generate the actual HTML document.
- Numerous server-side scripting languages
  - JSP, PHP
  - General purpose scripting languages: VBScript, Perl, Python

Java Server Pages (JSP)

- A JSP page with embedded Java code

```
<html>
<head> <title> Hello </title> </head>
<body>
<% if (request.getParameter("name") == null)
{ out.println("Hello World"); }
else { out.println("Hello, " + request.getParameter("name")); }
%>
</body>
</html>
```

- JSP is compiled into Java + Servlets
- JSP allows new tags to be defined, in tag libraries
  - such tags are like library functions, can are used for example to build rich user interfaces such as paginated display of large datasets

PHP

- PHP is widely used for Web server scripting
- Extensive libaries including for database access using ODBC

```
<html>
  <head> <title> Hello </title> </head>
  <body>
  <?php if (!isset($_REQUEST[‘name’]))
  { echo “Hello World”; }
  else { echo “Hello, ” + $_REQUEST[‘name’]; }
  ?>
  </body>
  </html>
```

Client Side Scripting

- Browsers can fetch certain scripts (client-side scripts) or programs along with documents, and execute them in "safe mode" at the client site
  - Javascript
  - Macromedia Flash and Shockwave for animation/games
  - VRML
  - Applets
- Client-side scripts/programs allow documents to be active
  - E.g., animation by executing programs at the local site
  - E.g., ensure that values entered by users satisfy some correctness checks
  - Permit flexible interaction with the user.
    - Executing programs at the client site speeds up interaction by avoiding many round trips to server

Client Side Scripting and Security

- Security mechanisms needed to ensure that malicious scripts do not cause damage to the client machine
  - Easy for limited capability scripting languages, harder for general purpose programming languages like Java
- E.g., Java's security system ensures that the Java applet code does not make any system calls directly
  - Disallows dangerous actions such as file writes
  - Notifies the user about potentially dangerous actions, and allows the option to abort the program or to continue execution.

Javascript

- Javascript very widely used
  - forms basis of new generation of Web applications (called Web 2.0 applications) offering rich user interfaces
- Javascript functions can
  - check input for validity
  - modify the displayed Web page, by altering the underling **document object model (DOM)** tree representation of the displayed HTML text
  - communicate with a Web server to fetch data and modify the current page using fetched data, without needing to reload/refresh the page
    - forms basis of AJAX technology used widely in Web 2.0 applications
    - E.g. on selecting a country in a drop-down menu, the list of states in that country is automatically populated in a linked drop-down menu

Javascript

- Example of Javascript used to validate form input

```
<html> <head>
    <script type="text/javascript">
        function validate() {
            var credits=document.getElementById("credits").value;
            if (isNaN(credits)|| credits<=0 || credits>=16) {
                alert("Credits must be a number greater than 0 and less than 16");
                return false
            }
        }
    </script>
</head> <body>
    <form action="createCourse" onsubmit="return validate()">
        Title: <input type="text" id="title" size="20"><br />
        Credits: <input type="text" id="credits" size="2"><br />
        <Input type="submit" value="Submit">
    </form>
</body> </html>
```

Application Architectures

Application Architectures

- Application layers
  - Presentation or user interface
    - **model-view-controller (MVC)** architecture
      - **model**: business logic
      - **view**: presentation of data, depends on display device
      - **controller**: receives events, executes actions, and returns a view to the user
  - **business-logic** layer
    - provides high level view of data and actions on data
      - often using an object data model
    - hides details of data storage schema
  - **data access** layer
    - interfaces between business logic layer and the underlying database
    - provides mapping from object model of business layer to relational model of database

Application Architecture

Business Logic Layer

- Provides abstractions of entities
  - e.g. students, instructors, courses, etc
- Enforces **business rules** for carrying out actions
  - E.g. student can enroll in a class only if she has completed prerequsites, and has paid her tuition fees
- Supports **workflows** which define how a task involving multiple participants is to be carried out
  - E.g. how to process application by a student applying to a university
  - Sequence of steps to carry out task
  - Error handling
    - e.g. what to do if recommendation letters not received on time
  - Workflows discussed in Section 26.2

Object-Relational Mapping

- Allows application code to be written on top of object-oriented data model, while storing data in a traditional relational database
  - alternative: implement object-oriented or object-relational database to store object model
    - has not been commercially successful
- Schema designer has to provide a mapping between object data and relational schema
  - e.g. Java class *Student* mapped to relation *student,* with corresponding mapping of attributes
  - An object can map to multiple tuples in multiple relations
- Application opens a session, which connects to the database
- Objects can be created and saved to the database using
  session.save(object)
  - mapping used to create appropriate tuples in the database
- Query can be run to retrieve objects satisfying specified predicates

Object-Relational Mapping and Hibernate (Cont.)

- The **Hibernate** object-relational mapping system is widely used
  - public domain system, runs on a variety of database systems
  - supports a query language that can express complex queries involving joins
    - translates queries into SQL queries
  - allows relationships to be mapped to sets associated with objects
    - e.g. courses taken by a student can be a set in Student object
  - See book for Hibernate code example
- The **Entity Data Model** developed by Microsoft
  - provides an entity-relationship model directly to application
  - maps data between entity data model and underlying storage, which can be relational
  - Entity SQL language operates directly on Entity Data Model

Web Services

- Allow data on Web to be accessed using remote procedure call mechanism
- Two approaches are widely used
  - **Representation State Transfer (REST)**: allows use of standard HTTP request to a URL to execute a request and return data
    - returned data is encoded either in XML, or in **JavaScript Object Notation (JSON)**
  - **Big Web Services:**
    - uses XML representation for sending request data, as well as for returning results
    - standard protocol layer built on top of HTTP
    - See Section 23.7.3

Disconnected Operations

- Tools for applications to use the Web when connected, but operate locally when disconnected from the Web
  - E.g. Google Gears browser plugin
    - Provide a local database, a local Web server and support for execution of JavaScript at the client
    - JavaScript code using Gears can function identically on any OS/browser platform
  - Adobe AIR software provides similar functionality outside of Web browser

Rapid Application Development

- A lot of effort is required to develop Web application interfaces
  - more so, to support rich interaction functionality associated with Web 2.0 applications
- Several approaches to speed up application development
  - Function library to generate user-interface elements
  - Drag-and-drop features in an IDE to create user-interface elements
  - Automatically generate code for user interface from a declarative specification
- Above features have been in used as part of **rapid application development (RAD**) tools even before advent of Web
- Web application development frameworks
  - Java Server Faces (JSF) includes JSP tag library
  - Ruby on Rails
    - Allows easy creation of simple **CRUD** (create, read, update and delete) interfaces by code generation from database schema or object model

ASP.NET and Visual Studio

- ASP.NET provides a variety of controls that are interpreted at server, and generate HTML code
- Visual Studio provides drag-and-drop development using these controls
  - E.g. menus and list boxes can be associated with DataSet object
  - Validator controls (constraints) can be added to form input fields
    - JavaScript to enforce constraints at client, and separately enforced at server
  - User actions such as selecting a value from a menu can be associated with actions at server
  - DataGrid provides convenient way of displaying SQL query results in tabular format

Application Performance

Improving Web Server Performance

- Performance is an issue for popular Web sites
  - May be accessed by millions of users every day, thousands of requests per second at peak time
- Caching techniques used to reduce cost of serving pages by exploiting commonalities between requests
  - At the server site:
    - Caching of JDBC connections between servlet requests
      - a.k.a. **connection pooling**
    - Caching results of database queries
      - Cached results must be updated if underlying database changes
    - Caching of generated HTML
  - At the client's network
    - Caching of pages by Web proxy

Application Security

SQL Injection

- Suppose query is constructed using
  - "select * from instructor where name = '" + name + "'"
- Suppose the user, instead of entering a name, enters:
  - X' or 'Y' = 'Y
- then the resulting statement becomes:
  - "select * from instructor where name = '" + "X' or 'Y' = 'Y" + "'"
  - which is:
    - select * from instructor where name = 'X' or 'Y' = 'Y'
  - User could have even used
    - X'; update instructor set salary = salary + 10000; --
- Prepared statement internally uses:
  "select * from instructor where name = 'X\' or \'Y\' = \'Y'
- **Always use prepared statements, with user inputs as parameters**
- Is the following prepared statemen secure?
  - conn.prepareStatement("select * from instructor where name = '" + name + "'“)

Cross Site Scripting

- HTML code on one page executes action on another page
  - E.g. <img src = http://mybank.com/transfermoney?amount=1000&toaccount=14523>
  - Risk: if user viewing page with above code is currently logged into mybank, the transfer may succeed
  - Above example simplistic, since GET method is normally not used for updates, but if the code were instead a script, it could execute POST methods
- Above vulnerability called **cross-site scripting (XSS)** or **cross-site request forgery (XSRF or CSRF)**
- **Prevent your web site from being used to launch XSS or XSRF attacks**
  - Disallow HTML tags in text input provided by users, using functions to detect and strip such tags
- **Protect your web site from XSS/XSRF attacks launched from other sites**
  - ..next slide

Cross Site Scripting

- **Protect your web site from XSS/XSRF attacks launched from other sites**
  - Use **referer** value (URL of page from where a link was clicked) provided by the HTTP protocol, to check that the link was followed from a valid page served from same site, not another site
  - Ensure IP of request is same as IP from where the user was authenticated
    - prevents hijacking of cookie by malicious user
  - Never use a GET method to perform any updates
    - This is actually recommended by HTTP standard

Password Leakage

- Never store passwords, such as database passwords, in clear text in scripts that may be accessible to users
  - E.g. in files in a directory accessible to a web server
    - Normally, web server will execute, but not provide source of script files such as file.jsp or file.php, but source of editor backup files such as file.jsp~, or .file.jsp.swp may be served
- Restrict access to database server from IPs of machines running application servers
  - Most databases allow restriction of access by source IP address

Application Authentication

- Single factor authentication such as passwords too risky for critical applications
  - guessing of passwords, sniffing of packets if passwords are not encrypted
  - passwords reused by user across sites
  - spyware which captures password
- Two-factor authentication
  - e.g. password plus one-time password sent by SMS
  - e.g. password plus one-time password devices
    - device generates a new pseudo-random number every minute, and displays to user
    - user enters the current number as password
    - application server generates same sequence of pseudo-random numbers to check that the number is correct.

Application Authentication

- **Man-in-the-middle** attack
  - E.g. web site that pretends to be mybank.com, and passes on requests from user to mybank.com, and passes results back to user
  - Even two-factor authentication cannot prevent such attacks
- Solution: authenticate Web site to user, using digital certificates, along with secure http protocol
- **Central authentication** within an organization
  - application redirects to central authentication service for authentication
  - avoids multiplicity of sites having access to user's password
  - LDAP or Active Directory used for authentication

Single Sign-On

- **Single sign-on** allows user to be authenticated once, and applications can communicate with authentication service to verify user's identity without repeatedly entering passwords
- **Security Assertion Markup Language (SAML)** standard for exchanging authentication and authorization information across security domains
  - e.g. user from Yale signs on to external application such as acm.org using userid joe@yale.edu
  - application communicates with Web-based authentication service at Yale to authenticate user, and find what the user is authorized to do by Yale (e.g. access certain journals)
- **OpenID** standard allows sharing of authentication across organizations
  - e.g. application allows user to choose Yahoo! as OpenID authentication provider, and redirects user to Yahoo! for authentication

Application-Level Authorization

- Current SQL standard does not allow fine-grained authorization such as “students can see their own grades, but not other’s grades”
  - Problem 1: Database has no idea who are application users
  - Problem 2: SQL authorization is at the level of tables, or columns of tables, but not to specific rows of a table
- One workaround: use views such as

```
create view  studentTakes  as
select *
from   takes
where takes.ID = syscontext.user_id()
```

  - where syscontext.user_id() provides end user identity
    - end user identity must be provided to the database by the application
  - Having multiple such views is cumbersome

Application-Level Authorization (Cont.)

- Currently, authorization is done entirely in application
- Entire application code has access to entire database
  - large surface area, making protection harder
- Alternative: **fine-grained (row-level) authorization** schemes
  - extensions to SQL authorization proposed but not currently implemented
  - Oracle Virtual Private Database (VPD) allows predicates to be added transparently to all SQL queries, to enforce fine-grained authorization
    - e.g. add *ID= sys_context.user_id()* to all queries on student relation if user is a student

Audit Trails

- Applications must log actions to an audit trail, to detect who carried out an update, or accessed some sensitive data
- Audit trails used after-the-fact to
  - detect security breaches
  - repair damage caused by security breach
  - trace who carried out the breach
- Audit trails needed at
  - Database level, and at
  - Application level

Encyption

Encryption

- Data may be *encrypted* when database authorization provisions do not offer sufficient protection.
- Properties of good encryption technique:
  - Relatively simple for authorized users to encrypt and decrypt data.
  - Encryption scheme depends not on the secrecy of the algorithm but on the secrecy of a parameter of the algorithm called the encryption key.
  - Extremely difficult for an intruder to determine the encryption key.
- **Symmetric-key encryption**: same key used for encryption and for decryption
- **Public-key encryption** (a.k.a. **asymmentric-key encryption**): use different keys for encryption and decryption
  - encryption key can be public, decryption key secret

Encryption (Cont.)

- *Data Encryption Standard* (DES) substitutes characters and rearranges their order on the basis of an encryption key which is provided to authorized users via a secure mechanism. Scheme is no more secure than the key transmission mechanism since the key has to be shared.
- Advanced Encryption Standard (AES) is a new standard replacing DES, and is based on the Rijndael algorithm, but is also dependent on shared secret keys.
- *Public-key encryption* is based on each user having two keys:
  - *public key* – publicly published key used to encrypt data, but cannot be used to decrypt data
  - *private key* -- key known only to individual user, and used to decrypt data. Need not be transmitted to the site doing encryption.

  Encryption scheme is such that it is impossible or extremely hard to decrypt data given only the public key.
- The RSA public-key encryption scheme is based on the hardness of factoring a very large number (100's of digits) into its prime components

Encryption (Cont.)

- **Hybrid schemes** combining public key and private key encryption for efficient encryption of large amounts of data
- Encryption of small values such as identifiers or names vulnerable to **dictionary attacks**
  - especially if encryption key is publicly available
  - but even otherwise, statistical information such as frequency of occurrence can be used to reveal content of encrypted data
  - Can be deterred by adding extra random bits to the end of the value, before encryption, and removing them after decryption
    - same value will have different encrypted forms each time it is encrypted, preventing both above attacks
    - extra bits are called **salt bits**

Encryption in Databases

- Database widely support encryption
- Different levels of encryption:
  - **disk block**
    - every disk block encrypted using key available in database-system software.
    - Even if attacker gets access to database data, decryption cannot be done without access to the key.
  - **Entire relations, or specific attributes of relations**
    - non-sensitive relations, or non-sensitive attributes of relations need not be encrypted
    - however, attributes involved in primary/foreign key constraints cannot be encrypted.
- Storage of encryption or decryption keys
  - typically, single master key used to protect multiple encryption/decryption keys stored in database
- Alternative: encryption/decryption is done in application, before sending values to the database

Encryption and Authentication

- Password based authentication is widely used, but is susceptible to sniffing on a network.
- **Challenge-response** systems avoid transmission of passwords
  - DB sends a (randomly generated) challenge string to user.
  - User encrypts string and returns result.
  - DB verifies identity by decrypting result
  - Can use public-key encryption system by DB sending a message encrypted using user's public key, and user decrypting and sending the message back.
- **Digital signatures** are used to verify authenticity of data
  - E.g., use private key (in reverse) to encrypt data, and anyone can verify authenticity by using public key (in reverse) to decrypt data. Only holder of private key could have created the encrypted data.
  - Digital signatures also help ensure **nonrepudiation:** sender cannot later claim to have not created the data

End of Chapter

Digital Certificates

- **Digital certificates** are used to verify authenticity of public keys.
- Problem: when you communicate with a web site, how do you know if you are talking with the genuine web site or an imposter?
  - Solution: use the public key of the web site
  - Problem: how to verify if the public key itself is genuine?
- Solution:
  - Every client (e.g., browser) has public keys of a few root-level **certification authorities**
  - A site can get its name/URL and public key signed by a certification authority: signed document is called a **certificate**
  - Client can use public key of certification authority to verify certificate
  - Multiple levels of certification authorities can exist. Each certification authority
    - presents its own public-key certificate signed by a higher level authority, and
    - uses its private key to sign the certificate of other web sites/authorities

A formatted report

Acme Supply Company, Inc.
Quarterly Sales Report

Period: Jan. 1 to March 31, 2009

Region	Category	Sales	Subtotal
North	Computer Hardware	1,000,000	
	Computer Software	500,000	
	All categories		1,500,000
South	Computer Hardware	200,000	
	Computer Software	400,000	
	All categories		600,000
		Total Sales	2,100,000

Figure 9.11

Web browser
Internet
1
8
Controller
6
7
5
2
View
Model
3
Data Access Layer
4
Database
Web/Application Server

Web Interfaces to Database (Cont.)

2. Dynamic generation of documents
   - Limitations of static HTML documents
     - Cannot customize fixed Web documents for individual users.
     - Problematic to update Web documents, especially if multiple Web documents replicate data.
   - Solution: Generate Web documents dynamically from data stored in a database.
     - Can tailor the display based on user information stored in the database.
       - E.g., tailored ads, tailored weather and local news, …
     - Displayed information is up-to-date, unlike the static Web pages
       - E.g., stock market information, ..