Chapter 9: Application Design and Development

Chapter 9: Application Design and Development

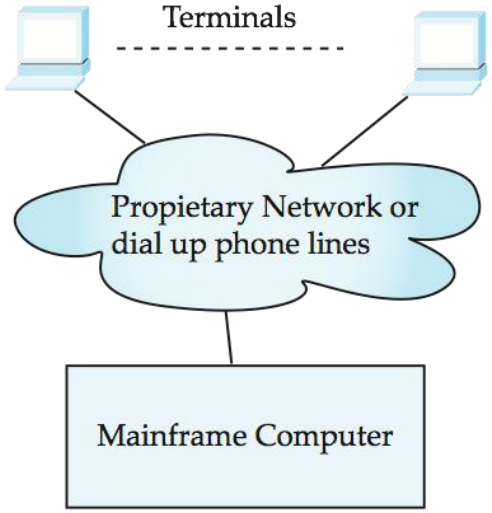
- Application Programs and User Interfaces
- Web Fundamentals
- Servlets and JSP
- Application Architectures
- Rapid Application Development
- Application Performance
- Application Security
- Encryption and Its Applications

Application Programs and User Interfaces

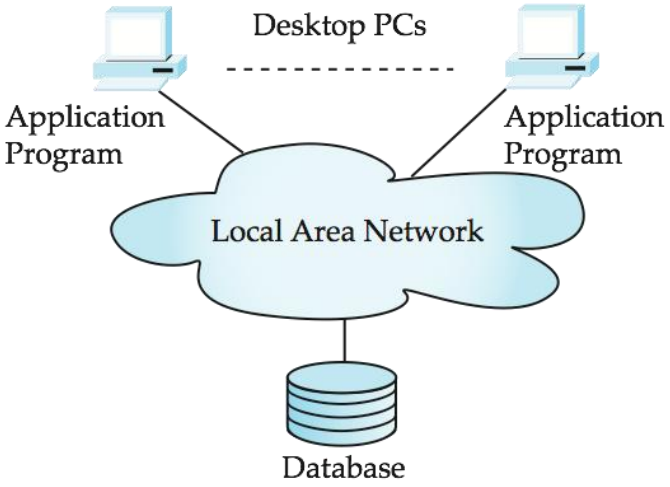
- Most database users do *not* use a query language like SQL
- An application program acts as the intermediary between users and the database
  - Applications split into
    - front-end
    - middle layer
    - backend
- Front-end: user interface
  - Forms
  - Graphical user interfaces
  - Many interfaces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

Application Architecture Evolution

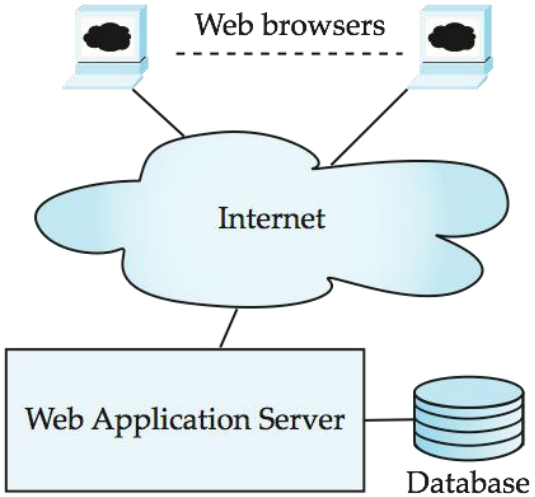
- Three distinct era's of application architecture
  - mainframe (1960's and 70's)
  - personal computer era (1980's)
  - We era (1990's onwards)



(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

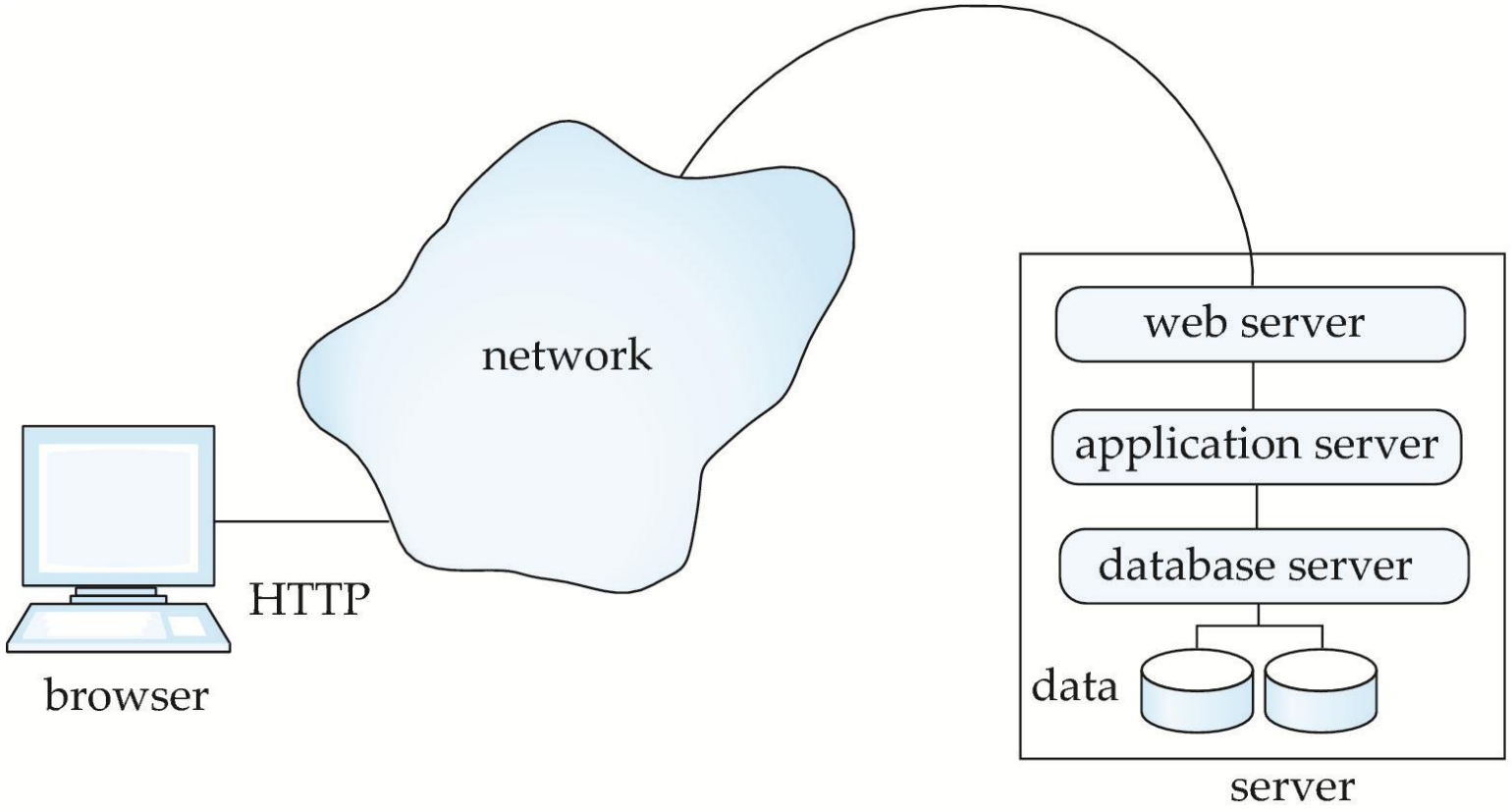
Name:

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

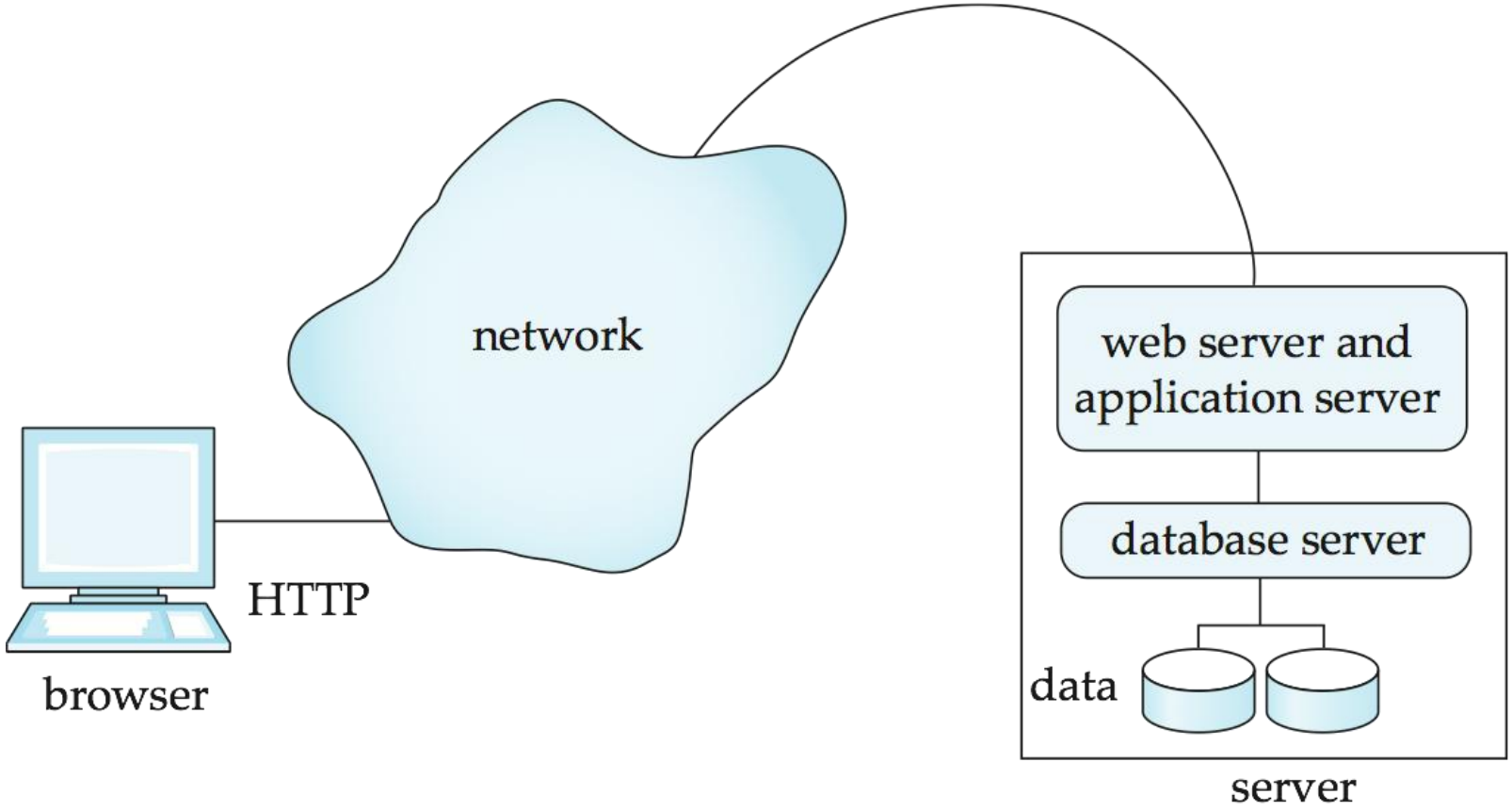
Three-Layer Web Architecture



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 name = 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

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

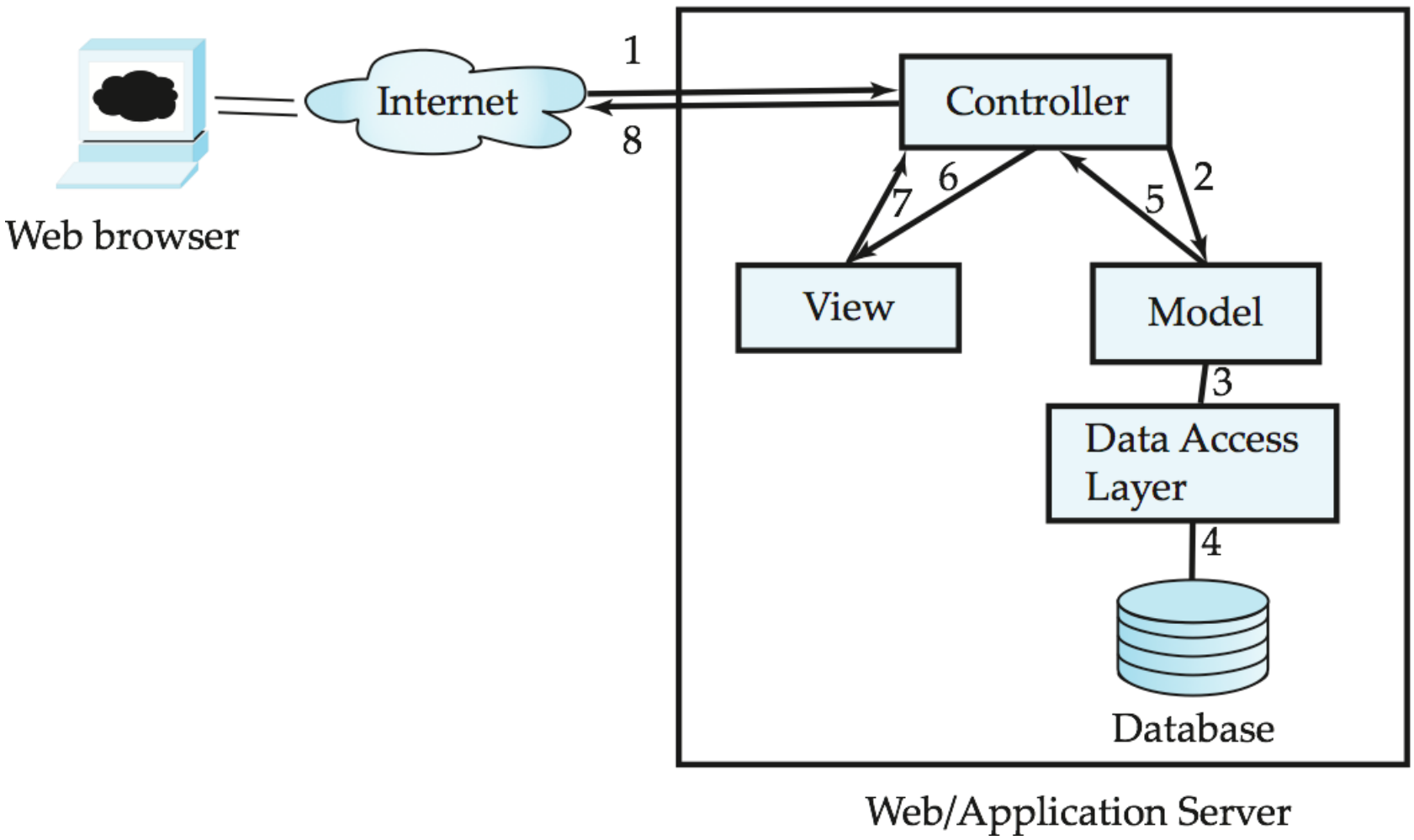
```
<html> <head>
    <script type="text/javascript">
        function validate() {
            var credits=document.getElementById("credits").value;
            if (isNaN(credits)|| credits<=0 || credits>=16) {
                alert("Credits must be a number greater than 0 and less than 16");
                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



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

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