11.1 Web Server Operation

- *Client-server systems*
  - When two computers are connected, either could be the client
  - The client initiates the communication, which the server accepts
  - Generally, clients are human consumers of information, while servers are machine suppliers
  - Client/server systems have an efficient division of work
- All communications between Web clients and servers use HTTP
- When a Web server starts, it tell its OS it is ready to accept communications through a specific port, usually 8080
- All current Web servers are descendents of the first two (CERN and NCSA)
- Most servers are Apache running under UNIX

11.2 General Server Characteristics

- Web servers have two separate directories

  - The *document root* is the root directory of all servable documents (well, not really *all*)

    e.g. Suppose the site name is `www.bloomers.com` and the document root is named `topdocs`, and it is stored in the `/admin/web` directory

    So, `/admin/web/topdocs` is the document directory address

    If a request URL is:

    `http://www.bloomers.com/bulbs/tulips.html`

    The server will search for the file with the given path

    `/admin/web/topdocs/bulbs/tulips.html`

  - The server can have *virtual document trees*

    - Sometimes a different disk, possibly on a different machine, is used after the original disk is filled

11.2 General Server Characteristics

(continued)

- The *server root* is the root directory for all of the code that implements the server
  - The server root usually has four files
    - One is the code for the server itself
    - *Three others are subdirectories*
      - `conf` - for configuration information
      - `logs` - to store what has happened
      - `cgi-bin` - for executable scripts
- *Contemporary servers provide many services*:
  - *Virtual hosts* - multiple sites on the same system
  - *Proxy servers* - to serve documents from the document roots of other sites
  - Besides HTTP, support for FTP, Gopher, News, email
  - Support for database access

11.3 Apache under UNIX

- Apache is available for other platforms, but it is most commonly used under UNIX

- Apache is now a large and complex system

- The configuration file is named `httpd.conf`

- The directives in the configuration file control the operation of the server

- *Configuration file format*:

  - Comments begin with a `#`

  - Blank lines are ignored

  - Non-blank lines that do not begin with `#` must begin with a directive name, which may take parameters, separated by white space

- When Apache begins, it reads the configuration files and sets its parameters according to what it reads

11.3 Apache under UNIX (continued)

- Changes to configuration files affect Apache only if it is forced to reset

  - Use the following UNIX commands to force Apache to reset:

```
cd /usr/local/etc/httpd/logs
kill -HUP `cat httpd.pid`
```

  - This creates a hangup signal

  - It works because Apache writes its process id (pid) into `httpd.pid` when it starts

- *Directives*

  - `ServerName` - default is what is returned by the `hostname` command, but it may be only the first part

```
ServerName    www.bloomers.com
```


11.3 Apache under UNIX (continued)

- `ServerRoot` - to set the server root address
  - Default is `/usr/local/etc/httpd`
    - If it is stored elsewhere, tell Apache with:

```
ServerRoot    /usr/local/httpd
```

- `ServerAdmin` - email address of the site admin

```
ServerAdmin    webguy@www.bloomers.com
```

- `DocumentRoot` - set the document root address
  - Default is `/usr/local/etc/httpd/htdocs`
    - If it is elsewhere, tell Apache with:

```
DocumentRoot    /local/webdocs
```


11.3 Apache under UNIX (continued)

- `Alias` - to specify a virtual document tree

  - Takes two parameters, virtual path for URLs and the actual path

  - Example:

```
Alias   /bushes    /usr/local/plants/bushes
```

  - Now,

```
http://www.bloomers.com/bushes/roses.html
```

  will be mapped to

```
/usr/local/plants/bushes/roses.html
```

- `ScriptAlias` - to create a secure place for CGI scripts

  - Creates a virtual directory

```
ScriptAlias  /cgi-bin/
             /usr/local/etc/httpd/cgi-bin/
```


11.3 Apache under UNIX (continued)

- `Redirect` - to redirect requests to another system

  e.g., To move the `bushes` directory to `www.bloomers2.com`

```
Redirect  /bushes
 http://www.bloomers2.com/local/web/bushes
```

- `DirectoryIndex` - URL-specified directories

  - When a request includes a URL that ends with a slash, Apache searches for a document to return, called the *welcome page*

    - Default welcome page is `index.html`

    - If there is no `index.html`, Apache may try to build a directory listing for the home directory (unless automatic directory listings are turned off)

    - To avoid this, provide more than one welcome page names

```
DirectoryIndex  index.html  contents.html
```


11.3 Apache under UNIX (continued)

- `UserDir` - to specify whether local users can or cannot add or delete documents; default is:

```
UserDir  public_html
```

  - Now, if user `bob` stores `stuff.html` in his `public_html` directory, the URL

```
http://site-name/~bob/stuff.html
```

    will work

  - To make a subdirectory of `public_html` available, include it in the parameter

```
UserDir   public_html/special_stuff
```

  - To disallow additions and deletions:

```
UserDir    disabled
```

- *Logs*

  - Access logs record all accesses (time, date, HTTP command, URL, status, etc.)

  - Error logs have the form:

    [date/time] The error message

11.4 Overview of Servlets

- A servlet is a compiled Java class

- Servlets are executed on the server system under the control of the Web server

- Servlets are managed by the *servlet container*, or *servlet engine*

- Servlets are called through HTML

- Servlets receive requests and return responses, both of which are supported by the HTTP protocol

- When the Web server receives a request that is for a servlet, the request is passed to the servlet container

  - The container makes sure the servlet is loaded and calls it

  - The servlet call has two parameter objects, one with the request and one for the response

  - When the servlet is finished, the container reinitializes itself and returns control to the Web server

11.4 Overview of Servlets (continued)

- Servlets are used 1) as alternatives to CGI, and 2) as alternatives to Apache modules

- *Servlet Advantages*:

  - Can be faster than CGI, because they are run in the server process

  - Have direct access to Java APIs

  - Because they continue to run (unlike CGI programs), they can save state information

  - Have the usual benefits of being written in Java (platform independence, ease of programming)

- *Java Server Pages* (JSP)

  - Provide processing and dynamic content to HTML documents (similar to servlets)

  - Can be used as a server-side scripting language (scriplets)

    - Scriplets are translated by the JSP container into servlets

11.5 Servlet Details

- All servlets are classes that either implement the `Servlet` interface or extend a class that implements the `Servlet` interface

- The `Servlet` interface provides the interfaces for the methods that manage servlets and their interactions with clients

- The Servlet interface declares three methods that are called by the servlet container (the *life-cycle methods)*

  - `init` - initializes the servlet and prepares it to respond to client requests

  - `service` - controls how the servlet responds to requests

  - `destroy` - takes the servlet out of service

11.5 Servlet Details (continued)

- The `Servlet` interface declares two methods that are used by the servlet:

  - `getServletConfig` - to get initialization and startup parameters for itself

  - `getServletInfo` - to allow the servlet to return info about itself (author, version #, etc.) to clients

- Most user-written servlet classes are extensions to `HttpServlet` (which is an extension of `GenericServlet`, which implements the `Servlet` Interface)

- Two other necessary interfaces:

  - `ServletResponse` – to encapsulate the communications, client to server

  - `ServletRequest` – to encapsulate the communications, server to client
    - Provides servlet access to `ServletOutputStream`

11.5 Servlet Details (continued)

- `HttpServlet` – an abstract class
  - Extends `GenericServlet`
  - Implements `java.io.Serializable`
  - Every subclass of `HttpServlet` MUST override at least one of the methods of `HttpServlet`

```
doGet*
doPost*
doPut*
doDelete*
init
destroy
getServletInfo
```

* Called by the server

11.5 Servlet Details (continued)

- The protocol of `doGet` is:

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
  throws ServletException, java.io.IOException
```

  - `ServletException` is thrown if the `GET` request could not be handled

- The protocol of `doPost` is the similar

- *Servlet output* – HTML

  1. Use the `setContentType` method of the response object to set the content type to `text/html`

```
response.setContentType("text/html");
```

  2. Create a `PrintWriter` object with the `getWriter` method of the response object

```
PrintWriter servletOut =
                        response.getWriter();
```

- Example – Respond to a `GET` request with no data

→Show `tst_greet.html` and `Greeting.java`

11.6 A Survey Example

--> Show `conelec2.html` and Figure 11.3

- *The servlet*:

- To accumulate voting totals, it must write a file on the server

- The file will be read and written as an object (the array of vote totals) using `ObjectInputStream`

- An object of this class is created with its constructor, passing an object of class `FileInputStream`, whose constructor is called with the file variable name as a parameter

```
ObjectInputStream indat =
  new ObjectInputStream(
    new FileInputStream(File_variable_name));
```

- On input, the contents of the file will be cast to integer array

- For output, the file is written as a single object

11.6 A Survey Example (continued)

- The servlet must access the form data from the client

  - This is done with the `getParameter` method of the request object, passing a literal string with the name of the form element

    e.g., if the form has an element named `zip`

```
zip = request.getParameter("zip");
```

  - If an element has no value and its value is requested by `getParameter`, the returned value is `null`

- If a form value is not a string, the returned string must be parsed to get the value

  - e.g., suppose the value is an integer literal

    - A string that contains an integer literal can be converted to an integer with the `parseInt` method of the wrapper class for `int`, `Integer`

```
price = Integer.parseInt(
          request.getParameter("price"));
```


11.6 A Survey Example (continued)

- The file structure is an array of 14 integers, 7 votes for females and 7 votes for males

- *Servlet actions*:

  If the votes data array exists
    read the votes array from the data file
  else
    create the votes array

  Get the gender form value
  Get the form value for the new vote and convert it to an integer
  Add the vote to the votes array
  Write the votes array to the votes file
  Produce the return HTML document that shows the current results of the survey

- Every voter will get the current totals

--> Show the servlet, `Survey.java`

--> Show Figure 11.4

11.7 Storing Information about Clients

- A *session* is the collection of all of the requests made by a particular browser from the time the browser is started until the user exits the browser

- The HTTP protocol is stateless

- But, there are several reasons why it is useful for the server to relate a request to a session

  - Shopping carts for many different simultaneous customers

  - Customer profiling for advertising

  - Customized interfaces for specific clients

- *Approaches to storing client information:*

  - Store it on the server – too much to store!

  - Store it on the client machine - this works

- *Cookies*

  - A cookie is an object sent by the server to the client

11.7 Storing Information about Clients

(continued)

- Every HTTP communication between the browser and the server includes information in its header about the message

- At the time a cookie is created, it is given a lifetime

- Every time the browser sends a request to the server that created the cookie, while the cookie is still alive, the cookie is included

- A browser can be set to reject all cookies

- A cookie object has data members and methods

  - Data members to store lifetime, name, and a value (the cookies' value)

  - Methods: `setComment`, `setMaxAge`, `setValue`, `getMaxAge`, `getName`, and `getValue`

- Cookies are created with the `Cookie` constructor

```
Cookie newCookie =
                    new Cookie(gender, vote);
```


11.7 Storing Information about Clients (continued)

- By default, a cookie's lifetime is the current session
  - If you want it to be longer, use `setMaxAge`
- A cookie is attached to the response with `addCookie`
- *Order in which the response must be built*:
  1. Add cookies
  2. Set content type
  3. Get response output stream
  4. Place info in the response
- The browser does nothing with cookies, other than storing them and passing them back
- A servlet gets a cookie from the browser with the `getCookies` method

```
Cookie theCookies [];
  …
  theCookies = request.getCookies();
```

- A Vote Counting Example

→ Show `ballot.html` and Figure 11.5

11.7 Storing Information about Clients (continued)

- *Vote counting servlet activities:*
  - See if a vote was cast
  - Make sure the voter hasn't voted before
  - Tally real votes and give the client the totals
  - Store votes in a file

→Show the `VoteCounter` algorithm

→ Show `VoteCounter`

→ Show Figures 11.6, 11.7, and 11.8

- *Session Tracking*
  - An alternative to cookies
  - Use the `HttpSession` object, which can store a list of names and values

11.7 Storing Information about Clients (continued)

- Create a `Session` object

- Put value in the session object with `putValue`

```
mySession.putValue("iVoted", "true");
```

- A session can be killed with the `invalidate` method

- A value can be removed with `removeValue`

- A value can be gotten with `getValue(name)`

- All names of values can be gotten with `getValueNames`

→ SHOW `VoteCounter2.java`

