Internet

- A global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP) to serve billions of users worldwide.
- It is a *network of networks* that consists of millions of private, public, academic, business, and government networks, of local to global scope, that are linked by a broad array of electronic, wireless and optical networking technologies.

Vinton Cerf

- Father of the internet

World Wide Web

- A global information medium which users can read and write via computers connected to the Internet. The term is often mistakenly used as a synonym for the Internet itself, but the Web is a service that operates over the Internet, as e-mail does.
- In September 1994, Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology with support from the Defense Advanced Research Projects Agency(DARPA) and the European Commission. It comprised various companies that were willing to create standards and recommendations to improve the quality of the Web.

Tim-Berners Lee

- Father of the web

HTTP (hypermedia)

- Application layer used primarily to retrieve hypertext (on hypermedia) documents and resources on the World Wide Web
- Jointly developed by the W3C and the IETF

Protocol

- Set of rules need to be followed.

Fundamentals

- HTTP typically runs on top of TCP/IP, using TCP port 80 by default (TCP port 443 for HTTPS).
- HTTP resources are identified using URIs (specifically, HTTP URLs)
    - Scheme (*http:* or *https:*)
    - (optional) authentication information
    - Host and (optional) port number
    - Path (resolved to the document root on the server) to the resource
    - (optional) scheme-specific parameters
    - (optional) URL-encoded query
    - (optional) bookmark (or fragment identifier)

http://jo:secret@myserver.com:8080/products/price.jsp;sessionid=123456?cat=school&man=xyz#summary

scheme username password host port path scheme specific parameters start of query URL query fragment

- HTTP is based on client-server architecture
- Clients, aka user agents (UA):
    - Web browsers, web crawlers, email clients, other end user tools and applications
- Servers:
    - Origin servers, proxy servers, gateways, tunnels
- HTTP uses a request-response standard protocol
    - The client sends an HTTP request message to the server
    - The server processes the request and replies with an HTTP response message
- HTTP is a stateless communications protocol
    - Servers do not keep information about clients in between requests
    - Web applications effect session tracking using mechanism such as cookies on URL-encoded session information to keep track of related client requests
- HTTP provides support for other functionalities such as cache control, content media type (MIME) specification, language and character set specification, content/transfer coding, client-server protocol negotiations, persistent connections, request pipelining, etc.

Cookie

- A very small text file

Cache

- Local storage/copy of resource that is fetched from a server

HTTP Response Message

- Status line
  - HTTP protocol version
  - Status code
  - Reason phrase
- Response headers
- Empty line
- Message body

HTTP Request Methods

- GET
  - Most commonly used HTTP method
  - Used to request from the server the retrieval of the source identified by the request URI; the retrieved resource is returned in the message body as an entity.
  - Can be combined with conditional and/or range request headers to effect conditional and/or partial resource retrieval
  - Must be supported by all general-purpose servers.
- HEAD
  - Identical to GET, except the message body is not included in the response
- POST
  - Request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line.
- OPTIONS
  - Request for information about the communication options available on the request/response chain identified by the Request-URI. This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating resource retrieval.
- TRACE
  - Request the server to "echo" back to the client the received request
  - Typically used for testing/diagnostics of the request chain
- PUT
  - Request the server to store the enclosed entity in the message under the specified request URI
- DELETE
  - Request the server to delete the resource identified by the request URI
- CONNECT
  - Reserved for use of tunneling proxy servers

Idempotent Methods

- The methods GET, HEAD, PUT and DELETE share this property.

HTTP Message Headers

- General Header Fields
  - `Cache-control`
  - `Connection`
  - `Date`
  - `Pragma`
  - `Trailer`
  - `Transfer-encoding`
  - `Upgrade`
  - `Via`
  - `Warning`
- Request Header Fields
  - `Accept`
  - `Accept-charset`
  - `Accept-encoding`
  - `Accept-Language`
  - `Authorization`
  - `Expect`
  - `From`
  - `Host`
  - `If-Match`
  - `If-Modified-Since`
  - `If-None-Match`
  - `If-Range`
  - `If-Unmodified-Since`
  - `Max-forward`
  - `Proxy-Authorization`
  - `Range`
  - `Referrer`
  - `User-Agent`
- Response Header Fields
  - `Accept-Range`
  - `Age`
  - `E-Tag`
  - `Location`
  - `Proxy-Authenticate`
  - `Retry-After`
  - `Server`
  - `Vary`
  - `WWW-Authenticate`
- Entity Header Fields
  - `Allow`
  - `Content-encoding`
  - `Content-language`

- Content-length
- Content-location
- Content-MD5
- Content-range
- Content-type
- Expires
- Last Modified

HTTP Status Codes

- Informational (1xx)
  - 100 Continue
  - 101 Switching Protocols
- Success (2xx)
  - 200 OK
  - 202 Accepted
  - 203 Non-Authoritative Information
  - 204 No Content
  - 205 Reset Content
  - 206 Partial Content
- Redirection (3xx)
  - 300 Multiple Choices
  - 301 Moved Permanently
  - 302 Found
  - 303 See Other
  - 304 Not Modified
  - 305 Use Proxy
  - 307 Temporary Redirect
- Client Error (4xx)
  - 400 Bad Request
  - 401 Unauthorized
  - 402 Payment Required
  - 403 Forbidden
  - 404 Not Found
  - 405 Method not Allowed
  - 406 Not Acceptable
  - 407 Proxy Authentication Required
  - 408 Request Timeout
  - 409 Conflict
  - 410 Gone
  - 411 Length Required
  - 412 Precondition Failed
  - 413 Request Entity Too Large
  - 414 Request-URI Too Long
  - 415 Unsupported Media Type
  - 416 Request Range Not Satisfiable
  - 417 Expectation Failed
- Server Error (5xx)
  - 500 Internal Server Error
  - 501 Not Implemented
  - 502 Bad Gateway
  - 503 Service Unavailable
  - 504 Gateway Timeout
  - 505 HTTP Version Not Supported

Hypertext Markup Language

- Language used to markup documents (i.e. web pages) in the WWW
  - Structure (and content)
  - Presentation (mostly deprecated in favor of style sheets)
- Versions
  - HTML 2.0, 3.2, 4.0
  - HTML 4.01
    - Strict – deprecated
    - Transitional – still make use the deprecated
    - Frameset – using frames
  - HTML 5 (working draft)
- `Head`
  - `title, base, link, meta, style, script`
- `Body`
  - Grouping elements (`div, span`)
  - Headings (`h1 – h6`)
  - Paragraphs, line breaks, horizontal rules (`p , br , hr`)
  - Lists (`ul, ol, li, dl, dt, dd, dir, menu`)
  - Tables (`table, th, tr, td, thead, tfoot, tbody, colgroup, col`)
  - Structured text
    - Phrase elements (`em, strong, dfn, code, samp, kbd, var, cite, abbr, acronym`)
    - Quotations (`blockquote, q`)
    - Subscripts and superscripts (`sub, sup`)
    - Preformatted texts (`pre`)
  - Font styles and alignments (`tt, i, b, big, small, strike, s, u, font, basefont, center`)
  - Document changes (`ins, del`)
  - Links and anchors (`a`)
  - Objects, images, applets (`object, img, param, applet`)
  - Scripts (`script, noscript`)
  - Miscellaneous (`address, bdo`)
  - Frames, noframes, iframe
  - Frameset (for frameset DTD)

Extensible Hypertext Markup Language

- Reformulation of HTML in XML
- Versions
  - XHTML 1.0
  - XHTML 1.1
  - XHTML 1.2
  - XHTML 2.0
  - XHTML 5 (working draft)
- Basic differences between HTML and XHTML
  - All elements have beginning and ending tags
  - All elements are nested properly
  - Elements and attribute names are case sensitive (lowercase)
  - Attribute values are quoted)
  - Attribute values cannot be minimized
- XHTML Doc. Components
  - Document Type Declaration (DOCTYPES)
    - HTML 4.01 DOCTYPES
    - XHTML 1.0 DOCTYPES
  - Elements
    - Tags
    - Content (elements)
    - Attributes
      - `id, class, title, alt, lang`, intrinsic event attributes (e.g. `onClick`)
    - Character Entity References
      - `< > &   `
- XHTML Elements
  - Block-level and inline (aka text-level) elements
    - Block-level elements represent “larger” document structures and may contain inline & other block level elements
    - Inline elements contain only data and other inline elements
- Author Styles
  - External style sheets (recommended)
  - Embedded styles
  - Inline styles
- User Style
- User Agent Style (example default style sheet for HTML 4)

Quirks

- Older ways of rendering documents

Standards

- *Following the specification by W3C for HTML, CSS*

Cascading Style Sheets

- Language used to specify the presentation aspects (i.e. layout and formatting of structurally marked up (e.g. HTML, XML, XHTML, SVG, etc.) document).
- Developed by *Hakon Wium Lie* (HTML Cascading Style Sheets / CHSS) and *Bert Bos* (Stream-based Style Sheet Proposal / SSP)
- Versions:
  - CSS 1, CSS 2, CSS 2.1, CSS 3 (structural semantic markup / modularizes CSS 2.1)
- CSS Rules
  - Selectors
    - Universal selector
      ```
      *{
      }
      ```
    - Type
      ```
      p{
      }
      ```
    - Grouping
      ```
      h1,h2,h3{
      }
      ```
    - ID
      ```
      #nav{
      }
      ```
    - Class

```
.hide{
}
```

- Descendant
```
div.content  em{
}
```
- Child
```
div.content > em{
}
```
- Adjacent sibling
```
h1 + p{
}
```
- Attribute acronym
```
div[class=note] {
}
```
- Pseudo classes
  - First child pseudo class (:first-child)
```
li:first-child{
}
```
  - Link pseudo class (:link, :visited)
```
a:link, a:visited{
}
```
  - Dynamic pseudo class (:hover, :active, :focus)
```
ul.menu > li{
}
ul.menu > li:hover{
}
```

- Declaration Block
  - Properties
    - Visual, aural, paged, interactive
    - Margins, borders, and padding
    - Colors and background
    - Fonts
    - Text formatting
    - Lists and tables
    - Positioning and dimensions
    - Miscellaneous
  - Values
    - Keywords
    - Numbers
    - Length
      - Length units: `em ex px` (relative) `in an mm pt pc` (absolute)
    - Percentages
    - Colors
    - Strings
    - URLs
    - Counters
    - Initial values, inherited values, !important rules
    - Specified, computed, used, and actual values
      - By origin and importance
        - User important declarations
        - Author important declarations
        - Author normal declarations
      - By specificity
        - Inline styles (1) or not (0)
        - Number of ID attributes in selector
        - Number of other attributes and pseudo-classes in selector
        - Number of element names and pseudo-elements in selector
      - By order
- CSS comments /* */

Client-side web scripting

- Allows programs (i.e. scripts) to be downloaded from a web server and executed in the client environment (e.g. browser)
- Common client-side scripting technologies
  - JavaScript, ECMAScript, Jscript
  - VBScript, ActionScript

- Java Applets, ActiveX Controls, Flash Animations, Microsoft Silverlight, Adobe Integrated Runtime (Adobe AIR)
- Common uses:
  - Dynamic (X)HTML
    - Page embellishments, visual "effects", content generation and manipulation, user interaction, document and page navigation
  - Client-side form data validation
  - Asynchronous content retrieval, RIA Technologies (e.g. AJAX).
- Common issues:
  - Browser support
    - No scripting support
    - Scripting disabled
    - Plugin availability
  - Version incompatibilities, non-standard implementations
  - Capabilities restrictions
    - Scripting languages are not for general-purpose programming
    - E.g. javascript is restricted by the sandbox execution model and the same origin policy
  - Security risks
    - Browser implementation defects (e.g. buffer overflows)
    - E.g. javascript, cross-site scripting (XSS) or cross-site request forgery (XSRF) issues
    - Malicious ActiveX controls

Javascript

- Developed circa 1995 by *Brendan Eich* at Netscape Communications as the scripting language for the Netscape Navigator Browser
- Formerly called Mocha, then LiveScript, then JavaScript
- Standardized by ECMA International as ECMAScript
- Latest version: JavaScipt 1.8.1, ECMAScript 5 [ECMA-262 5th Edition]
- Common version: JavaScript 1.5, Jscript 5.5, ECMAScript v3 [ECMA-262 3rd Edition]
- JavaScript Frameworks:
  - script.aculo.us, jQuery, MooTools, Prototype, Dojo Toolkit, etc.
- Linked/Embedded in web pages using the <script> element
  - Linked:

```
<script type="text/javascript" src="scripts.js" /> </script>
```

  - Embedded (either in the `<head>` or the `< body>` element)

```
<script type="text/javascript">
        <!--hide script from non-JavaScript browsers…
        /* script code goes here */
        // end of script hiding … -->
</script>
<noscript>
        …content to show when scripting not available
</noscript>
```

- JavaScript + DOM/BOM + CSS + (X)HTML = DHTML
- JavaScript code in (X)HTML pages can be executed "on the fly" as the document is rendered (i.e. code outside of functions executes as the <script> element is encountered); in most cases though, JavaScript code is executed in response to document events (e.g. clicking a page element).
- Basic language features:
  - Paradigm:
    - Object-oriented (prototype-based), functional, imperative scripting language
  - Java-/C-like syntax
    - Implicit semicolon insertion for statement termination
    - Identifiers are alphanumeric, _, and $ characters
    - Single-line (//) or block (/**/) comments
  - Type system and variable scoping rules:
    - Dynamic (aka loose or weak) typing
    - Global (aka top-level) or local scopes
    - Data types:
      - Primitive types
        - `Numbers (decimal, hexadecimal notation)`
        - `Booleans (true, false)`
        - `Strings (Single or Double quote delimited)`
        - `Undefined and Null`
      - Composite (object) types
        - Core JavaScript Objects
          - `Object, Number, Boolean, String, Date, Math, Global, RegExp, Error`
          - `Arrays (Array)`
          - `Functions (Function, Arguments)`

        - DOM Objects
            - `Anchor, Applet, Attr, Comment, DOMException, DOMImplementation, DocumentFragment, Element, Event, Form, Image, Input, Layer, Link, Node, Option, Select, Style, Text, TextArea`
    - Keywords
        - `break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, with`
    - Reserved words (currently unused)
        - `abstract, Boolean, byte, char, class, const, debugger, float`
    - Statements and control structures
        - `var`
            - Used to declare global/local scoped variables
        - `if-else`
            - Condition expressions having values of `0, ""`, `null`, and `undefined` evaluate to `false`
        - `switch-case-default-break`
            - allows any expression type to be used as the switch expression
            - `case` labels may be of different types
            - `case` labels may be expressions
            - `case` execution falls-through, unless terminated by a `break`
        - `while, do-while, for, for-in, break, continue`
            - for while and do-while, false condition expressions similar to if-else
            - for-in used for property enumeration
            - allows labeled break/continue
        - `try-catch-finally, throw`
            - `throw` and `catch` can handle any expression type
        - `function, return`
            - JavaScript functions are similar to Java methods except for the following differences:
                - No return value type is specified, and `return` is optional within the function body
                - Functions may return a value on one invocation and not return a value (i.e. have an `undefined` return value) on another invocation
                - Functions may return different types of values on different invocations
                - Function parameters are dynamically typed
                - Functions can be invoked with an arbitrary number of arguments, regardless of the actual parameters specified in the function definition (the `Arguments` object can be used to access unnamed arguments passed to the function invocation)
                - Functions are first-class objects
            - Functions can be invoked as global functions (i.e. as methods of the `Global` object) as methods of specific objects, or as object constructors
        - `with`
            - Used to access object properties without having to explicitly qualify the property with the object name (serves as shorthand notation for accessing object properties, at the expense of program readability)

Asynchronous JavaScript and XML (AJAX)

- Synchronous Request
    - Recommended for few small requests. JavaScript will not continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop.

```
var xhr = new XMLHttpRequest();

xhr.open("GET", "somepage.txt", false);

xhr.send(null);

if(xhr.status==200){

        var str = xhr.responseText;

}
```

- Asynchronous Request
    - JavaScript does not have to wait for the server response. It can execute other scripts while waiting for server response, and deal with the response when the response is ready.

```
var xhr = new XMLHttpRequest();
```

```
xhr.open("POST", "products.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onreadystatechange = function(){
        if(this.readyState == 4 && this.status == 200){
                var hdrDate = this.getResponseHeader("Date");
                var xml = this.responseXML;
        }
}

xhr.send("productID=1234&catID=5678");
```

- Cross-Browser and Old Browser Support

```
var xhr;
if(typeof XMLHttpRequest != "undefined"){
        xhr = new XMLHttpRequest();
}else if (window.ActiveXObject){
        var activeXIDs = ["MSXML2.XmlHttp.5.0",
                          "MSXML2.XmlHttp.4.0",
                          "MSXML2.XmlHttp.3.0",
                          "MSXML2.XmlHttp",
                          "Microsoft.XmlHttp"];

        for(var i=0; i<activeXIDs.length;i++){
                try{
                        xhr = new ActiveXObject(activeXIDs[i]);
                }catch(e){}
        }
}
```

- Three important properties of the `XMLHttpRequest` Object
  - `onreadystatechange`
    - Stores a function to be called automatically each time `readyState` changes
  - `readyState`
    - 0 : request not initialized
    - 1 : server connection established
    - 2 : request received
    - 3 : processing request
    - 4 : request finished and response is ready
  - `status`
    - `200` : OK
    - `404` : Not Found

Document Object Model

- Document Object Collections
  - `anchors[]`
  - `forms[]`
  - `images[]`
  - `links[]`
- Document Object Properties
  - `cookie`
  - `documentMode`
  - `domain`
  - `lastModified`
  - `readyState`
  - `referrer`
  - `title`
  - `URL`
- Document Object Methods
  - `close()`
  - `getElementById()`
  - `getElementsByName()`
  - `getElementsByTagName()`
  - `open()`
  - `write()`
  - `writeln()`

Java EE

- Java EE is an open, standards-based development and deployment platform for creating distributed, transactional, reliable, secure, multi-tiered, web-based, server-centric, component-based enterprise applications
- Java EE Application Model
  - Java programming language, Java Virtual Machine (JVM)
  - Java EE Components
    - Java EE Clients
      - Application Clients, Applets (embedded in web clients)
    - Web Components
      - Servlets, JavaServer Pages (JSP), JavaServer Faces (JSF)

        - Enterprise JavaBeans (EJBM)
    - Java EE Containers
        - Client containers, web container, EJB container
    - Java EE Server
- Java EE Web Application
    - Collection of resources installed under a specific subset of the URL namespace of a web application server compliant with the Java EE Specification (e.g. Apache's Tomcat, Apache's Geronimo, Sun Microsystems' Glassfish, IBM's WebSphere, etc.)
    - Resources
        - Static resources: web pages, images, stylesheets, etc. *(serves as is)*
        - Dynamic resources: servlets, JSPs
        - Miscellaneous resources: business object classes (e.g. Java Beans, EJB), support libraries, etc.
    - XML-formatted descriptor and configuration files
        - `web.xml, application.xml, context.xml, etc.`
    - Organized into a standard hierarchical structure and typically packaged and deployed as WAR or EAR files
- Java EE APIs
    - Enterprise JavaBeans Technology
    - Java Servlet Technology
    - Java Server Pages
    - Java Server Pages Standard Tag Library
    - Java Server Faces
    - Java Msg Service API
    - Java Transaction API
    - JavaMail API
    - JavaBeans Activation Framework
    - Java API for XML Processing

Servlets

- Java Object based on the Servlet API
- Runs in a server application to answer client requests; technically, servlets are not tied to a specific client-server protocol, but they are most commonly used with HTTP and the term 'servlet' is often used in the context of an "HTTP Servlet"
- Web-tier components in the Java EE architecture.
- Runs in, and is managed by, a web-tier container called the 'Servlet Container'
- Mapped to URLs to which clients send requests
- Typically asked with (among other things)
    - Processing and/or storing data submitted vial HTML forms
    - Generating dynamic content
- `javax.servlet`
    - `Servlet, GenericServlet`
    - `ServletRequest, ServletResponse`
    - `ServletConfig, ServletContext`
    - `RequestDispatcher`
- `javax.servlet.http`
    - `HttpServlet`
    - `HttpServletRequest`
    - `HttpServletResponse`
    - `HttpSession`
    - `Cookie`
- Servlet Processing
    - Client sends a request to a web server URL that is mapped to a servlet. Web server passes on the request to the servlet container
    - Servlet container checks if servlet is already loaded
    - If it is not yet loaded, servlet container loads the servlet class and instantiates the servlet, and calls its `init` method.
    - Servlet container invokes the servlet's `service` method, passing request and response objects as arguments
    - Servlet processes the request using the response object to create the response, which is returned by the servlet container to the web server, which in turn sends the response to the client
    - Subsequent request to the servlet will not require servlet re-instantiation, unless the servlet has been unloaded; before a servlet is unloaded, the servlet container invokes its `destroy` method.
- `init(config)`
    - Invoked once on the servlet by the servlet container when the servlet is instantiated; can be used by the servlet for one-time startup initialization
- `service(request, response)`
    - Invoked each time the servlet is called upon to process a request (typically on a separate thread for each call)
    - In `HttpServlet`, the default Service implementation maps the call to a specific doXXX() method (e.g. `doGet`, `doPost`) which is typically overridden to affect the servlet's functionality
- `Destroy()`

    - Invoked on the servlet by the servlet container when the servlet is to be unloaded (e.g. when the application is stopped or undeployed); can be used by the servlet for clean-up processing (e.g. resource deallocation)
- Servlet Request Processing (`HttpServletRequest`)
    - Retrieving user-supplied request parameters
    - Retrieving request header values
- Servlet Response Processing (`HttpServletResponse`)
    - Setting response status code
    - Setting response headers
    - Obtaining output object for sending the response
- Servlet Request Dispatching (`RequestDispatcher`)
    - Obtain a `RequestDispatcher` to a resource (static or dynamic) from the request object

```
RequestDispatcher rqstDsp;
rqstDsp = request.getRequestDispatcher(res);
```

    - Include the dispatcher resource (or its output) in the current response; one or more resources can be included (e.g. use for banners, footers, etc.)

```
rqstDsp.include(request, response);
```

    - Forwards the processing of the current request to the dispatcher resource; the servlet processing the current request must not generate a response (e.g. use in MVC “controller” servlets)

```
rqstDsp.forward(request, response);
```

- Session Tracking `(HttpSession)`
    - Session tracking support is implemented either cookies or URL-rewriting
    - Obtaining session object from the current request

```
HttpSession session;

session = request.getSession(createNew);
```

    - Obtaining session information `(HttpSession)`
        - `getCreationTime(), getLastAccessedTime(), getMaxInactiveInternal(), getId(), isNew(), setMaxInactiveInterval(int val)`
    - destroying sessions
        - `invalidate()`
    - URL-rewriting `(HttpServletResponse)`
        - `encodeURL(String url), encodeRedirectURL(String url)`

Web Context (`ServletContext`)

- a web application is associated with a context, which is an object that provides methods that servlets use to communicate with the servlet container
- obtaining the servlet context (`HttpServlet`)

```
ServletContext context;
context = this.getServletContext();
```

- obtaining context information (`ServletContext`)
    - `getServerInfo(), getContextPath(), getRealPath(), getResource(), getResourceAsStream(), getMimeType, getInitParameter(), getInitParameterNames(), getRequestDispatcher(), getContext()`

Servlet Configuration (`ServletConfig`)

- `getServletName(), getServletContext(), getInitParameter(), getInitParameterNames()`

Information sharing using scope objects

- A request may be processed by several web application components (e.g. through calls to `RequestDispatcher forwad/include`) and there may be a need for one component to communicate information to the other components in the request processing chain.
- A client session typically consists of multiple requests, which due to the stateless nature of HTTP, will appear to the application as being “unrelated” to one another; the `HttpSession` object can be used to “relate” these requests together, but there may still be a need to share information created in one request with a subsequent request
- Different web application components may require access to common resources or information (e.g. page counters, shared database connection).

- Information sharing is accomplished by creating attribute objects and exposing these objects in the appropriate scope.
- Scopes:
    - Request scope (`HttpServletRequest`)
    - Session scope (`HttpSession`)
    - Web Application or Web Context scope (`ServletContext`)
    - Page scope (local objects in a servlet)
- Creating, accessing, and removing attribute objects
    - `setAttribute (String attrName, Object attrValue)`
    - `getAttribute (String attrName)`
    - `getAttributeNames()`
    - `removeAttribute (String attrName)`

Advanced Servlet Topics

Listeners

- java objects used to "subscribe" to application "events" in order to be "notified" when these events occur
    - context-related events
        - context initialized, context destroyed, context attribute changes
    - session-related events
        - session created, session destroyed, session attribute changes
    - request-related events
        - request initialized, request destroyed, request attribute changes

`javax.servlet`

- `ServletContextListener, ServletContextAttributelistener`
- `ServletRequestListener, ServletRequestAttributeListener`

`javax.servlet.http`

- `HttpSessionListener, HttpSessionAttributeListener`

Filters

- Java objects used to intercept incoming requests and outgoing responses in order to perform various tasks such as:
    - Authentication and access control
    - Logging, auditing
    - Caching, data compression
    - Content Transformation
- Filter objects are mapped to the URL patterns they are intended to intercept
- Filter objects can be "chained" together

`javax.servlet`

- `Filter, FilterChain, FilterConfig`

Java Server Pages

- Simply an HTML web page that contains additional bits of code that execute application logic to generate dynamic content.
- Java Server Pages Actions (JSP tags) perform a variety of functions and extend the capabilities of JSP.
- Java Server Pages Actions use XML-like syntax, and are used to manage JavaBeans component.
- Directives are instructions that are processed by the JSP engine when the page is compiled to a servlet.
- Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries
    - `<%@ %>`
- Motivation
    - It is typically a good idea to separate business logic from presentation concern
        - Allows modern web development teams to be divided up into programmers and web page authors / designers
        - Fosters component reuse (e.g. the same data object can be consumed by user agents of varying capabilities and needs)
    - Servlets can be very powerful for programming business logic, but are very awkward to use when generating static (i.e. template) content.

- (X)HTML marked-up documents are very convenient for static content generation but cannot be used to program business logic (or generate dynamic content arising from data produced by the business logic).

- Features
  - Text-based document capable of generating both static and dynamic content (typically intermixed)
  - Mark-up based document syntax (JSP-style or XML-style), combining (X) HTML elements as well as standard and custom JSP elements; thus, web page authors can feel right "at home" with the mark-up syntax.
  - Embedded Java Coding support via "scriptlets"
    - `<%  %>`
  - Template text are converted into JSPWriter
- Components
  - Template (i.e. static) text
  - JSP elements
    - Directives
      - `<%@ page ContentType="text/html" pageEncoding="UTF-8" %>`
      - `<%@ page import="java.util.Random" %>`
        - autoFlush
        - buffer
        - contentType
        - errorPage
        - extends
        - import
        - info
        - isELIgnored
        - isErrorPage
        - isThreadSafe
        - language
        - pageEncoding
        - session
      - `<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>`
        - prefix
        - taglib
        - uri
    - Scripting Elements
      - Declarations
        - `<%! int a = 100; %>`
        - `<%! int square(int n) { return n*n ; } %>`
      - Expressions
        - `<% String s = new java.util.Date().toString(); %>`
      - Scriptlets
        - `<% for(int i = 0; i < 10 ; i++) { out.println(i); } %>`
    - Actions
      - Standard actions
        - `<jsp:directive.include>, <jsp:directive.page>`
        - `<jsp:declarations>`
        - `<jsp:scriptlet>`
        - `<jsp:expression>`
        - `<jsp:include>, <jsp:forward>`
        - `<jsp:useBean>, <jsp:setProperty>, <jsp:getProperty>`
        - `<jsp:plugin>, <jsp:param>, <jsp:params>, <jsp:fallback>`
        - `<jsp:element>, <jsp:attribute>, <jsp:body>`
        - `<jsp:text>`
      - Custom Actions (JSTL)
        - JSTL, user-written custom tag libraries
      - Expression Language (EL)
        - `${  }`
      - Implicit Scripting Objects
        - `request, response, out, pageContext`
        - `session, pageContext, application`
        - `config, page, exception`
      - Implicit EL Objects
        - `pageContext`
        - `pageScope`
        - `requestScope`
        - `sessionScope`
        - `applicationScope`
        - `param, paramValues`
        - `header, headervalues`

    - cookie
    - initparam
- Comments
    - <%-- this is a JSP comment --%>

Servlets or JSPs?

- The common practice is to leverage both technologies to implement the MVC design pattern

Model-View-Controller (MVC) Design Pattern

- Model
    - Represents business objects (logic and state)
- View
    - Presentation of the model in some appropriate way
- Controller
    - Mediates application flow

A sample web MVC framework can use

- JavaBeans for the model
- JSPs for the View
- Servlets for the controller

JSP Standard Tag Library (JSTL)

- Set of custom JSP elements that provide various programmatic functionality via markup syntax
    - Core Tag Library
        - variable support, flow control, URL management
    - SQL Tag Library
        - Database connections, queries, updates
    - Internationalization Tag Library
        - Locate setting, message bundling, number formatting, date formatting
    - XML
        - Core XML processing, flow control, transformation
    - JSTL Function
        - String functions, collection lengths
- In addition to the JSTL, developers can also create their own tag libraries for commonly occurring tasks

ACRONYMS:

Adobe AIR (Adobe Integrated Runtime)

AJAX (Asynchronous JavaScript and XML)

ANSI (American National Standards Institute)

ASP (Active Server Pages)

BOM (Browser Object Model)

CFML (ColdFusion Markup Language)

CGI (Common Gateway Interface)

DHTML (Dynamic HTML)

DOM (Document Object Model)

DTD (Document Type Definition)

EAR (Enterprise Archive)

EIS (Enterprise Information Systems)

EJB (Enterprise JavaBeans)

EL (Expression Language)

IIS (Internet Information Services)

JAR (Java ARchive)

JSF (Java Server Faces)

JSON (JavaScript Object Notation)

JSP (Java Server Pages)

JSTL (JSP Standard Tag Libraries)

PERL (Practical Extraction and Reporting Language)

PHP (Hypertext Preprocessor)

RDF (Resource Description Framework)

RIA (Rich Internet Application)

RSS (Really Simple Syndication)

SMIL (Synchronized Multimedia Integration Language)

SMX (Server Macro Expansion)

SOAP (Simple Object Access Protocol)

SVG (Scalable Vector Graphics)

WAP (Wireless Application Protocol)

WAR (Web Application Archive)

WSDL (Web Services Description Language)

XForms (XML Forms)

XLink (XML Linking Language)

XPointer (XML Pointer Language)

XQuery (XML Query Language)

XSD (XML Schema)

XSL (Extensible Style Sheet Language)

XSL-FO (Extensible Style Sheet Language Formatting Objects)

XSLT (XSL Transformations)

XSRF (Cross-site request forgery)

XSS (Cross-site scripting)