SSC - Web applications and development
Introduction and Java Servlet (I)

Shan He

School for Computational Science
University of Birmingham

Module 06-19321: SSC
Outline of Topics

What will we learn

Web development

Java servlet

Java servlet: our first example
What will we learn

What is web applications and development?

- Java Servlet: basic concepts, configure, install and use servlet based web applications, basic implementation.
- Java Serve Pages
- Java Model View Controller (MVC) for Java Servlet
- Advanced topics: Spring, JavaServer Faces, Multithreaded Servlet
What is web applications and development?

- Web-based application: an application that uses a web browser as a client, e.g., google calendar or GMail.
- Web development: work involved in developing a web site for the Internet or intranet, which include:
  - web design
  - web content development
  - client-side/server-side scripting
- Web development scripts:
  - Client side: HTML5, JavaScript, Ajax (Asynchronous JavaScript), Flash, JavaFX, etc.
  - Server side: PHP, Python, Node.js, Java servlet
  - Client-side/Server-side: Google Web Toolkit
What is Java servlet?

▷ Java servlet: a Java platform technology “for extending the functionality of a Web server and for accessing existing business systems”, part of the Java Enterprise Edition

▷ Component-based, server/platform-independent method for building Web-based applications

▷ Can use entire family of Java APIs, also can use existing POJOs (Plain Old Java Objects)

▷ Using Java servlet, you can generate dynamic web pages based on:
  ▶ data submitted by the user, e.g., results pages from search engines and Google calendar
  ▶ the data changes frequent, e.g., news headlines page or weather report
  ▶ information from corporate databases or other such sources, e.g., online store such as Amazon
Why use Java servlet?

- Efficient: the dynamic web pages are easier to write and faster to run.
- Convenient: you can use your familiar language Java.
- Powerful: Java servlets provide all the powerful features of JAVA, such as multi-threads, exception handling and garbage collection.
- Portable: Servlets are written in Java and can run across Web Servers without changes on Apache or Microsoft IIS.
- Versatile: you can communicate with different servlet and servers.
- Inexpensive: Running servlets only requires Java, Apache or Tomcat, which are all free.
Java servlet vs other Web development script

- **Java vs PHP vs Perl vs Python**

- **Note:** Java is statically typed language: types are associated with a variable declaration.

- **Note:** While PHP/Perl/Python are dynamically typed language, e.g., it does not associate values strictly with a specific type.

- **Advantages:**
  - Statically typed language ensures earlier detection of errors during compiling.
  - Many third party libraries for Java Web development.
  - Modern Java Web Development framework such as Spring is robust and convenient.
  - Java provide power multithreaded features.
  - Java is faster than PHP and other languages.
Java servlet life cycle

- Java Servlets are run inside a Servlet compatible "Servlet Container", e.g., Apache Tomcat, JBoss, Jetty, etc.

- A Servlet's life cycle consists of the following steps:
  - Step 1: Load Servlet Class.
  - Step 2: Create Instance of Servlet.
  - Step 3: Call the servlets init() method.
  - Step 4: Call the servlets service() method.
  - Step 5: Call the servlets destroy() method.

- Note 1: By default, a servlet is not loaded until the first request is received for it.

- Note 2: When the servlet container unloads the servlet, destroy() method is called and the container finalises the servlet and collect garbage.
Java servlet life cycle

Servlet Container

Load class

Instantiation

Init()

Service()

Destroy()

Finalisation and garbage collection

Request

Response
How to set up your programming IDE

- What are required?
  - Java EE
  - A Servlet Container, e.g., Apache Tomcat

- You need to read how to run tomcat from school's support web

- I am using Eclipse to show you how to build your first servlet but you can also use NetBean. See a tutorial here
Creating your first servlet

- Create a new “Dynamic Web project”: File menu → New → Dynamic Web Project
- Enter the project name
- Choose “Target Runtime”, e.g., Tomcat X.x
- Choose “Dynamic web module version” depends on your Servlet API version (Servlet 3.0 for Java EE6+ and Tomcat 7)
- Tick “Generate web.xml deployment descriptor”
- Create a new “Sevlet” which will extend HttpServlet class
- Write methods for each HTTP method (GET, POST etc.)
Recalled what we learned: HTTP (Hypertext Transfer Protocol) protocol has the following methods: GET(), POST(), HEAD(), PUT(), DELETE(), etc.

The most useful two:

- **GET()**: Requests data from a specified resource (server)
- **POST()**: Submits data to be processed to a specified resource (server)

Note: In a GET request, query strings (name/value pairs) is sent in the URL, which can be cached, bookmarked, remained in the browser history, have length restrictions.

GET requests should be used only to retrieve non-sensitive data.

Comparison between GET and POST
Java servlet explained: **HttpServlet** class

▶ In Java, your servlet classes extend the HttpServlet abstract class

▶ Two most important methods:
  
  ▶ **doGet()**: to respond HTTP GET requests only
  
  ▶ **doPost()**: to respond HTTP POST requests only

▶ From one method you can call the other method to handle both GET and POST requests, e.g.,

▶ Both methods required two parameters:

  ```java
doGet(HttpHttpServletRequest request,
       HttpServletResponse response)
```

▶ Two parameters are two classes with a lot of methods to process HTTP requests and responses
Java servlet explained: HttpRequest class

► Provides request information for HTTP servlets.
► Request Parameters: sent as part of the URL (in the "query string"), or as part of the body of an HTTP request.

http://georgewbush.com/mytopsecrets.html?
Action=showme&Document=NuclearLaunchCodes

► A query string of two parameters with parameter values:
  ► Action=showme
  ► Document=NuclearLaunchCodes

► getParameterNames : Returns the value of a request parameter as a String, or null if the parameter does not exist.
► getParameter : Returns an Enumeration of String objects containing the names of the parameters in the "query string".
Java servlet explained: **HttpResponse** class

- Provides HTTP-specific functionality in sending a response
- `setHeader`: Set **HTTP Header**, e.g.,
  ```java
  response.setHeader("Header-Name", "Header Value");
  ```
- `getWriter()`: Returns a **PrintWriter** from the **HttpResponse** object, so the server can send character text to the client.
  ```java
  PrintWriter writer = response.getWriter();
  writer.write("<html><body>HTML response</body></html>");
  ```
- `getOutputStream`: Returns a **ServletOutputStream** for writing binary data in the response.
- `sendRedirect`: Sends a temporary redirect response to the client using the specified redirect location URL