**Prelim Exam**

**Web Development (Lab)**

**Part 1 (65 points) HTML**

Create your own blog site with the following rubrics and validate your source code using W3C Markup Validation Service ([http://validator.w3.org](http://validator.w3.org/))

**Attachments:** Screenshot of your sample web page, source code and W3C validation.

Table

Description automatically generated

**Part 2 (50 points) Tomcat Server  
STEP 0: Create a Directory to Keep all your Works**

Create a directory called "c:\myWebProject" (for Windows) or "~\myWebProject" (for macOS).

**STEP 1: Take note of Your Tomcat Installed Directory**. Hereafter refer to the Tomcat installed directory as <TOMCAT\_HOME>.

##### **Tomcat's Sub-Directories**

Take a quick look at the Tomcat installed directory. It contains these sub-directories:

* **bin**: contains the binaries and scripts (e.g., startup.bat and shutdown.bat for Windows; startup.sh and shutdown.sh for Unixes and macOS).
* **conf**: contains the system-wide configuration files, such as server.xml, web.xml, and context.xml.
* **webapps**: contains the webapps to be deployed. You can also place the WAR (Webapp Archive) file for deployment here.
* **lib**: contains the Tomcat's system-wide library JAR files, accessible by all webapps. You could also place external JAR file (such as MySQL JDBC Driver) here.
* **logs**: contains Tomcat's log files. You may need to check for error messages here.
* **work**: Tomcat's working directory used by JSP, for JSP-to-Servlet conversion.

**STEP 2: Create an Environment Variable JAVA\_HOME**

You need to create an *environment variable* (system variable available to all applications) called "JAVA\_HOME” and set it to your JDK installed directory.

**Follow the steps:**

First, check if JAVA\_HOME is already set by start a CMD and issue: (cmd > set JAVA\_HOME)

If JAVA\_HOME is not set, you will receive "Environment variable JAVA\_HOME not defined", please install the latest JAVA SDK. Otherwise, set up your JAVA\_HOME

To set/change JAVA\_HOME in Windows:

1. Launch "Control Panel"
2. "System"
3. "Advanced system settings"
4. Switch to "Advanced" tab
5. "Environment variables"
6. Choose "System Variables" (for all users)
7. To *add* a new environment variable "JAVA\_HOME":
   * 1. Choose "New"
     2. In "Variable Name", enter "JAVA\_HOME".
     3. In "Variable Value", click "Browse Directory..." and navigate to the JDK installed directory (e.g., "C:\Program Files\Java\jdk-15.0.xx").
     4. OK ⇒ OK ⇒ OK.
8. To *change* the existing "JAVA\_HOME" setting:
   * 1. Select "JAVA\_HOME" then "Edit"
     2. In "Variable Value", click "Browse Directory..." and navigate to the JDK installed directory (e.g., "C:\Program Files\Java\jdk-15.0.xx").
     3. OK ⇒ OK ⇒ OK.

You need to **RE-START** CMD for the new setting to take effect!

Text

Description automatically generatedTo verify the new setting, re-start CMD:

**STEP 3: Configure the Tomcat Server**

The Tomcat configuration files, in XML format, are located in the "conf" sub-directory of your Tomcat installed directory, e.g. "c:\myWebProject\tomcat\conf" (for Windows) or "~/myWebProject/tomcat/conf" (for macOS). The important configuration files are:

1. server.xml
2. web.xml
3. context.xml

**Step 3(a) "conf\server.xml" - Set the TCP Port Number**

Download programming text editor (Sublime Text or Atom) to open the configuration file “server.xml”. The default TCP port number configured in Tomcat is 8080, you may choose any number between 1024 and 65535, which is not used by existing applications. We shall choose 9999 in this exam. (For production server, you should use port 80, which is pre-assigned to HTTP server as the default port number.) Locate the following lines (around Line 69) that define the HTTP connector, and change port="8080" to port="9999".

##### **Step 3(b) "conf\web.xml" - Enable Directory Listing**

Again, use a programming text editor to open the configuration file "web.xml".

We shall enable directory listing by changing "listings" from "false" to "true" for the "default" servlet. This is handy for test system, but not for production system for security. Locate the following lines (around Line 122) that define the "default" servlet; and change the "listings" from **"false" to "true".**

##### **Step 3(c) "conf\context.xml" - Enabling Automatic Reload**

We shall add the attribute reloadable="true" to the <Context> element to enable automatic reload after code changes. Again, this is handy for test system but not recommended for production, due to the overhead of detecting changes.

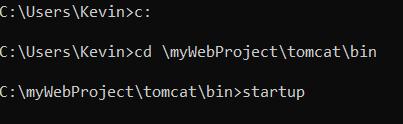
Locate the <Context> start element (around Line 19), and change it to: <Context **crossContext="true" reloadable="true"**>

**STEP 4: Start Tomcat Server**

The Tomcat's executable programs and scripts are kept in the "bin" sub-directory of the Tomcat installed directory.

**For Windows**

I shall assume that Tomcat is installed in "c:\myWebProject\tomcat". Launch a CMD shell and issue:



A new Tomcat console window appears (with Java's coffee-cup logo as icon). Study the messages on the console. Look out for the Tomcat's port number. Double check that Tomcat is running on port 9999 as configured.

A picture containing text

Description automatically generated**(Paste here the screenshot that your tomcat server is running on Port9999 and that the server is starting up.)**

**Step 4(b) Start a Client to Access the Server**

Start a browser as an HTTP client. Issue URL **"http://localhost:9999"** to access the Tomcat server's welcome page. The hostname "localhost" (with IP address of 127.0.0.1) is meant for local loop-back testing within the same machine.

**WARNING**: You MUST properly shutdown the Tomcat. DO NOT kill the CAT by pushing the window's "CLOSE" button.

**STEP 5: Develop and Deploy a "Hello-world" WebApp**

**Step 5(a) Create the Directory Structure for your WebApp**

Goto Tomcat's "webapps" sub-directory and create the following directory structure for your webapp "hello" The directory names are case-sensitive.

1. Under Tomcat's "webapps", create your webapp's *root* directory "hello" (i.e., "<TOMCAT\_HOME>\webapps\hello").
2. Under "hello", create a sub-directory "WEB-INF" (case sensitive, a "dash" not an underscore) (i.e., "<TOMCAT\_HOME>\webapps\hello\WEB-INF").
3. Under "WEB-INF", create a sub-sub-directory "classes" (case sensitive, plural) (i.e., "<TOMCAT\_HOME>\webapps\hello\WEB-INF\classes").

You need to keep your web resources (e.g., HTMLs, CSSs, images, scripts, servlets, JSPs) in the proper directories:

* "hello": This called the *context root* (or *document base directory*) of your webapp. You should keep all your HTML files and resources visible to the web users (e.g., HTMLs, CSSs, images, scripts, JSPs) under this *context root*.
* "hello/WEB-INF": This directory, although under the context root, is *not visible* to the web users. This is where you keep your application's web descriptor file "web.xml".
* "hello/WEB-INF/classes": This is where you keep all the Java classes such as servlet class-files.

You need to RE-START your Tomcat server to pick up the hello webapp. Check the Tomcat's console to confirm that "hello" application has been properly deployed. After confirming, access it using http://localhost:9999/hello

**PASTE HERE THE SCREENSHOT OF YOUR TOMCAT CONSOLE TO CONFIRM THAT “HELLO” WEBAPP HAS BEEN PROPERLY DEPLOYED AND THE SCREENSHOT OF YOUR DIRECTORY LISTING.**

**Example:**

A picture containing graphical user interface

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Step 5(a) Write a Welcome Page**

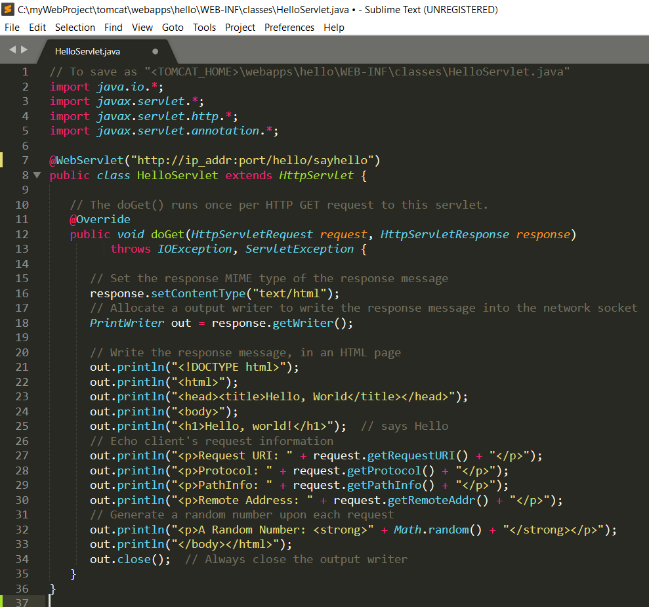
Saved your created HTML document in your webapps root directory “hello” and access it thru [http://localhost:9999/hello/[your\_file\_name].html](http://localhost:9999/hello/%5byour_file_name%5d.html) and <http://localhost:9999/hello>.

**PASTE HERE THE SCREENSHOT OF USING BOTH ADDRESS.**

**STEP 6: Write a "Hello-world" Java Servlet using programming text editor (ex.: Sublime or Atom)**

**Code is as follows:**

Text

Description automatically generated

##### **Step 6(b) Compiling the Servlet – CMD. Go to your folder and the output of the compilation is "HelloServlet.class".**

Text

Description automatically generated**PASTE HERE THE FILES INSIDE YOUR CLASSES DIRECTORY INCLUDING THE TYPE FILE. It should has a .java and .class file type.**

##### **Step 6(c) Invoke the Servlet**

To invoke this servlet, start a browser, and issue the request URL configured as follows:

**http://localhost:9999/hello/sayhello**

You shall see the output of the servlet displayed in your web browser.

Refresh the browser, you shall see a new random number upon each refresh. In other word, the doGet() method of the servlet runs once per request.

Then. Go to View Page Source and inspect the HTTP Requests and Response Messages

**PASTE HERE THE SCREENSHOT OF REQUEST MESSAGE HEADER AND IDENTIFY THE BODY.**

**PASTE HERE THE SCREENSHOT OF RESPONSE MESSAGE HEADER AND IDENTIFY THE BODY.**