Lab: Create JSP Home Page Using NetBeans

Table of Contents

1. Overview 1
2. Learning Objectives 1
3. Requirements for your Home Page 2
4. Requirements for your Second Page (labs.jsp) 4
5. Suggested Approach 5

Appendix: How to Move an Internal Style Sheet to an External Style Sheet 6

Appendix: How to Use JSP Include Directives to Reference Common HTML Code 6

1. Overview

In this lab you will:

- Install the NetBeans bundle and create a JSP web application.
- From the pages referenced by our lab page (that summarize information presented by w3schools.com), you'll learn about Basic HTML elements, CSS properties, and web color codes. You will learn how to create a simple layout for your home page.
- Create and publish a two page web site (index.jsp and labs.jsp) that both reference an external style sheet. The topic of your web site will be what you proposed in your previous labs.
- Once you've published your site and tested what you published, you'll attach a zip file of your website into blackboard.

2. Learning Objectives

By the end of this lab, you should

- be able to use basic HTML tags to create a well formed HTML page (tags like: head, title, style, div, p, a, img).
- be able to create an attractively laid out home page using CSS to control:
    - font, font color, link color, background color, background image, alignment
- know how to use NetBeans for its
    - HTML syntax checking editor and
    - source formatting.
- know how to debug HTML layout problems (using colored borders, backtracking and/or divide and conquer).
- know how to select colors using a browser plugin and/or a color blender web site.

3. Requirements for your Home Page

HTML ELEMENTS:

So that we know you are creating your layout from scratch, and so that there is consistency between students, your home page (index.jsp) shall have at least these HTML elements:

- One element with id=”title”
- One element with id=”nav”
- Two or more elements with class = "navLink"
- One element with id=”content”
- One element with id=”footer”

CSS Style Rules:

1. Your style sheet shall have at least these CSS selectors (to style the HTML elements above):
   - #title { … } #nav { … } .navLink #content { … } #footer { … }
2. So that the links in your navigation bar are styled differently than other links in your pages, your style sheet shall have rules something like this:
   - #nav a { … } → add “text-decoration: none;” so that the nav links are not underlined
   - #nav a:link { … } /* style of nav link before it’s clicked */
   - #nav a:visited { … } /* style of nav link after it’s clicked */
3. There shall be some hover behavior for the nav links. For example:
   - #nav div:hover { … } /* it’s usual to specify a background color when the user hovers over a div in your nav bar */

FONTS and COLORS and IMAGES:

1. Your home page shall incorporate at least one image – either as a background image or as a regular <img>.
2. ***All text*** (including link text) shall be readable on its background – no dark text on dark background, no light on light. If you have a background image with text on it, make sure the color of the text is not hidden by the background image.
3. All text shall be large enough to read.
   - Small fonts should use a san-serif font (for readability). Large text can use serif or san-serif. (Serif is French for “tail”. Times New Roman is a serif font because there are “tails” at the edges of the letters. Arial is sans-serif because it has no tails.)
4. Color palette shall be tasteful (e.g., select 2-3 colors from an image, and stick to those colors and variants of those colors (lighter/darker, more/less saturated, blending of the selected colors). Our class’s lab page talks about this, but you can download the “ColorZilla” (color picker) plugin (for firefox or chrome) and you can also visit the “Meyer color blender” web page.
5. If you want some 3D effects, you can consider using text-shadow and/or box-shadow (CSS properties).

WHITE SPACE

1. Your nav links shall be a focal point, nicely styled and noticeable in your layout (not too small/thin etc).
2. Your layout shall use padding and/or margins so that text is never too close to any visible edges.
3. Your page should make effective use of white space – not too cluttered, important things larger etc.

FLUID DESIGN:

All users will not have a screen with the same width and/or resolution as yours. Test that your layout looks good in all browser widths by resizing your browser narrower and narrower. At some small width (e.g., 750 px or less), if the layout starts to do something undesirable (like wrap the nav links), apply min-width to prevent that. min-width will cause the horizontal scroll bar to appear at the width you specify (and wrapping will cease).

TEXT:

1. Your web site title shall be on the page and also in the <title> tag in the <head> (becomes window title).
2. Your navigation bar shall have these links:
   - "Home"- references "index.jsp"
   - "Users" (or substitute the name of your user table) - references "users.jsp"
   - <<Other>> (substitute the name of the table that you created in your database) – references "other.jsp"
   - <<Associative>>> (substitute the name of your associative table in your db) – references "assoc.jsp"
   - "Search" – references "search.jsp"
   - "Members" – references "members.jsp"
   - "Labs" - references "labs.jsp"

   To be clear, all students shall have the same named JSP pages (href attribute), but differently named link text (link text shall align with student's database table names).
3. Your marketing material (that was part of your project proposal) shall be the basis of what you put in your content area.
   - If you choose to do a "hero" style layout (not too much text, large font on a background image), you'll need to have a second, more plain (but similar) style for the non-home pages that will show tabular data extracted from databases. If doing hero, come up with the similar plain style for your labs page.
4. Your name and email address shall be in your footer.

CODE REUSE:

1. The index page shall reference an external style sheet.
2. The index page shall reference jsp include files so that common areas of the page (e.g., title/nav, footer) will be referenced by all pages of your site (so you could add a nav link for example just in one place).

OPTIONAL SPECIAL EFFECTS:

1. You might like to use rounded corners somewhere (border-radius).
2. You might like to use some transparency on the background of some div (called opacity).
3. Gradients are nice.

4. *Requirements for your Second Page (labs.jsp)*

Your labs.jsp page shall be a copy of your index.jsp page (after you moved the style into an external style sheet & after you referenced the JSP include files), except that the content area of the labs page will have a blog for each lab. Each week of the semester, you will add to your labs page. Add a blog for labs 1, 2, and 3. For example:

<h2>Lab 1: Web App Proposal and Data Model </h2>
In this lab, I learned how to create a data model to support my web application. Click **here** for my latest/updated web application proposal. Click **here** to see my data model**.**

<h2>Lab 2: Database</h2>
In this lab, I populated my database with realistic data. I wrote these **SQL select statements** that demonstrate how to join data from several tables

<h2>Lab 3: JSP Web Site</h2>
In this lab, I learned how to use basic HTML and CSS to create a simple layout for a web site. Click **here** to see my home page.

…

5. *Suggested Approach*

- The first suggestion is to **start early** so that you have time to **ask for help** if you need to. Work with classmates (but don't share code). Ask the TA or me for help. You can email me a zip file, tell me what your problem is and I can provide feedback, but it won't be immediate (so start early).
- If you are new to HTML and CSS, use my HTML/CSS tutorial page to get started, **learn from the tutorials** and **study the sample code**. Run the code and observe its functionality. Change the code and see if you get what you anticipated. Otherwise, you will waste a lot of time on "trial and error" (and you will not do well in the weekly quiz). My page references w3schools.com, but the "value added" is that it identifies which are the most important HTML elements and CSS properties you need to get started. Take the time to understand each example --
- If you haven't already **install the Netbeans Bundle** at home, do that first… There is a separate document (see link from the lab page) that tells you how to install the Netbeans Bundle and how to use it to create and run your first JSP Web Application.
- If you didn't already come up with your **project proposal** (see the "project ideas" document in a previous lab), do that first, too…
- **Read about web design** (see link from our course's lab page). This page explains:
    - how to get started with web design, showing you the most commonly used HTML elements and the most commonly used CSS properties and how to combine them to make a nice looking but simple layout,
    - web color codes and how to select colors, and
    - image file formats (and how to reduce their size if they are too big, so your page does not load too slowly).
    - How to use w3schools.com as a tutorial and for reference.
- One of the first things to do when you work on your layout is to **select a color palette**. I recommend that you select an image that is related to your topic and that has colors that you would like to incorporate into your layout. Select 2-3 colors from this image and stick to them (or some variation, for example lighter/darker, more/less saturated, blended together). To get colors from an image, download and install ColorZilla (plugin for firefox, also one available for chrome). Once you have this installed, you'll see an eyedropper icon in your browser window – this is what you use to grab the colors out of the page. Visit a page like meyer color blender to make variations of your selected color palette.
- Using what you learned by reading about HTML and CSS, start with one of the sample layouts (from my "getting started with web design" page) and begin **modifying the layout for your own web site**. Incorporate your picture and be consistent with your color usage.
    - How to debug layout - Add a thin border to areas where the spacing is not what you want or expect. This can show you where you need to add or subtract space. Then you can remove the border.
    - Make frequent backups so that you do not lose any work.
- Once you are totally happy with your index page, make sure that it is syntactically correct by checking the NetBeans editor that there are no red errors flagged. Also view your page in firefox and check the "View Source" to be sure there are no red syntax errors identified. Then, **back up your index page**.
- Move the internal style sheet from your index page to an **external style sheet** (see appendix for example). View the index page (and it's new external style sheet) from firefox, View Source and check for syntax errors.
- Make your home page reference common code using **JSP include statements**. See appendix for example.
    - Now, it is really important to View Source from firefox. Netbeans can't give you syntax errors that arise out of faulty JSP include statements.
- Make a copy of your index.jsp → **labs.jsp**. Delete the contents of the labs page and put the blogs in. You will upload your **word documents from labs one and two** and link to them just as you would any other internal link.
- Test locally, then publish, then submit zip file of web root folder into blackboard.

Appendix: How to Move an Internal Style Sheet to an External Style Sheet

Eventually, we will have more pages than just the home page and so we want all pages to reference a single style sheet (for ease of web site maintenance). When all files reference a single style sheet, then you can change the style of ALL pages by making one change to the style sheet.

index.jsp BEFORE (style in the head)

```
<html>
 <head>
  <style>
   body {
    background-color:pink;
    color:green;
   }
  </style>
 </head>
 <body>
  Hello World
 </body>
</html>
```

That HTML page with its styles moved to an external style sheet.

index.jsp AFTER (references external style sheet)	**mystyle.css (NEW external style sheet)**

<html>
  <head>
  <link href="mystyle.css" rel="stylesheet" type="text/css" />
 </head>
  <body>
  Hello World
  </body>
</html>
	body {
	  background-color:pink;
	  color:green;
	 }

After making these changes, save all your files and make sure index.jsp still renders properly.

Appendix: How to Use JSP Include Directives to Reference Common HTML Code

Because you will eventually have many pages in your site, you do not want, for example, to have your nav bar copy/pasted into each of all of those pages. If you did, you'd have all of those pages to change if someone wants to add a link to the nav bar. So, I recommend that you have three include files:

- toHead.jsp: code in common, starts at the top of the page and leaves you in the head (in case a page might want, for example, to add an internal style sheet into its head section).
- headToContent.jsp: the code that finishes off the head, starts the body, then has the common code at the top of the page (title, nav), starts the content.
- postContent.jsp: the code that finishes off the content div, then the footer and ends the body and html tags.

index.jsp BEFORE using JSP includes

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>JSP Page</title>
    <link href="mystyle.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div id="title">
      My Title
      <div id="nav">
        links
      </div>
    </div>
    <div id="content">
      My Content
    </div>
    <div id="footer">
      My Footer
    </div>
  </body>
</html>
```

After making these changes, remember to save all files, then test that index.jsp still renders properly.

index.jsp AFTER using JSP includes

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

  <jsp:include page="toHead.jsp" />

  <style>
    /* override any style, as needed, for this page */
  </style>

  <jsp:include page="headToContent.jsp" />

   My Content

  <jsp:include page="post-content.jsp" />
```

toHead.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
     <link href="mystyle.css" rel="stylesheet" type="text/css" />
```

headToContent.jsp

```
  </head>
  <body>
    <div id="title">
      My Title
      <div id="nav">
        links
      </div>
    </div>
    <div id="content">
```

postContent.jsp

```
    </div>
    <div id="footer">
      My Footer
    </div>
  </body>
</html>
```