

WEB PROGRAMMING LANGUAGES

**LABORATORY WORKBOOK**

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TABLE OF CONTENT**

Hyper Text Markup Language 3

Cascading Style Sheets 24

HTML Form Design 37

CSS Identifiers and Classes 60

JQuery 68

ASP.NET Validation Controls 79

Master Pages 87

ASP.NET Themes 94

ASP.NET Custom Controls 99

Language Integrated Query 107

Entity Framework 113

Localization / Globalization 122

ASP.NET Identity 128

ASP.NET MVC 138



WEB PROGRAMMING LANGUAGES

**LAB-1**

# Hyper Text Markup Language

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Getting to Know HTML

With our introduction to HTML and CSS complete, it’s time to dig a little deeper into HTML and examine the different components that make up this language.

In order to start building websites, we need to learn a little about which HTML elements are best used to display different types of content. It’s also important to understand how elements are visually displayed on a web page, as well as what different elements mean semantically.

Using the proper element for the job goes a long way, and we’ll want to make well-informed decisions in the process.

## Semantics Overview

So what exactly are semantics? [Semantics within HTML](http://boagworld.com/dev/semantic-code-what-why-how/) is the practice of giving content on the page meaning and structure by using the proper element. Semantic code describes the value of content on a page, regardless of the style or appearance of that content. There are several benefits to using semantic elements, including enabling computers, screen readers, search engines, and other devices to adequately read and understand the content on a web page. Additionally, semantic HTML is easier to manage and work with, as it shows clearly what each piece of content is about.

Moving forward, as new elements are introduced, we’ll talk about what those elements actually mean and the type of content they best represent. Before we do that, though, let’s look at two elements—<div>s and <span>s—that actually don’t hold any semantic value. They exist for styling purposes only.

## Identifying Divisions & Spans

Divisions, or <div>s, and <span>s are HTML elements that act as containers solely for styling purposes. As generic containers, they do not come with any overarching meaning or semantic value. Paragraphs are semantic in that content wrapped within a<p> element is known and understood as a paragraph. <div>s and <span>s do not hold any such meaning and are simply containers.

## Block vs. Inline Elements

Most elements are either block- or inline-level elements. What’s the difference?

Block-level elements begin on a new line, stacking one on top of the other, and occupy any available width. Block-level elements may be nested inside one another and may wrap inline-level elements. We’ll most commonly see block-level elements used for larger pieces of content, such as paragraphs.

Inline-level elements do not begin on a new line. They fall into the normal flow of a document, lining up one after the other, and only maintain the width of their content. Inline-level elements may be nested inside one another; however, they cannot wrap block-level elements. We’ll usually see inline-level elements with smaller pieces of content, such as a few words.

Both <div>s and <span>s, however, are extremely valuable when building a website in that they give us the ability to apply targeted styles to a contained set of content.

A <div> is a block-level element that is commonly used to identify large groupings of content, and which helps to build a web page’s layout and design. A <span>, on the other hand, is an inline-level element commonly used to identify smaller groupings of text within a block-level element.

We’ll commonly see <div>s and <span>s with class or id attributes for styling purposes. Choosing a class or id attribute value, or name, requires a bit of care. We want to choose a value that refers to the content of an element, not necessarily the appearance of an element.

For example, if we have a <div> with an orange background that contains social media links, our first thought might be to give the <div> a class value of orange. What happens if that orange background is later changed to blue? Having a class value oforange no longer makes sense. A more sensible choice for a class value would besocial, as it pertains to the contents of the <div>, not the style.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <!-- Division -->  <div class="social">  <p>I may be found on...</p>  <p>Additionally, I have a profile on...</p>  </div>  <!-- Span -->  <p>Soon we'll be <span class="tooltip">writing HTML</span> with the best of them.</p> |

## Comments within HTML & CSS

The previous code includes exclamation points within the HTML, and that’s all right. Those are not elements, those are comments.

HTML and CSS give us the ability to leave comments within our code, and any content wrapped within a comment will not be displayed on the web page. Comments help keep our files organized, allow us to set reminders, and provide a way for us to more effectively manage our code. Comments become especially useful when there are multiple people working on the same files.

HTML comments start with <!-- and end with -->. CSS comments start with/\* and end with \*/.

## Using Text-Based Elements

Many different forms of media and content exist online; however, text is predominant. Accordingly, there are a number of different elements for displaying text on a web page. For now we’ll focus on the more popular elements, including headings, paragraphs, bold text to show importance, and italics for emphasis. Later, within Lesson 6, “[Working with Typography](http://learn.shayhowe.com/html-css/working-with-typography/),” we’ll take a closer look at how to style text.

## Headings

Headings are block-level elements, and they come in six different rankings, <h1>through <h6>. Headings help to quickly break up content and establish hierarchy, and they are key identifiers for users reading a page. They also help search engines to index and determine the content on a page.

Headings should be used in an order that is relevant to the content of a page. The primary heading of a page or section should be marked up with an <h1> element, and subsequent headings should use <h2>, <h3>, <h4>, <h5>, and <h6> elements as necessary.

Each heading level should be used where it is semantically valued, and should not be used to make text bold or big—there are other, better ways to do that.

Here is an example of HTML for all the different heading levels and the resulting display on a web page.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <h1>Heading Level 1</h1>  <h2>Heading Level 2</h2>  <h3>Heading Level 3</h3>  <h4>Heading Level 4</h4>  <h5>Heading Level 5</h5>  <h6>Heading Level 6</h6> |

## Paragraphs

Headings are often followed by supporting paragraphs. Paragraphs are defined using the <p> block-level element. Paragraphs can appear one after the other, adding information to a page as desired. Here is example of how to set up paragraphs.

|  |  |
| --- | --- |
| 1  2  3  4 | <p>Steve Jobs was a co-founder and longtime chief executive officer at Apple. On June 12, 2005,  Steve gave the commencement address at Stanford University.</p>  <p>In his address Steve urged graduates to follow their dreams and, despite any setbacks, to never give  up&ndash;advice which he sincerely took to heart.</p> |

## Bold Text with Strong

To make text bold and place a strong importance on it, we’ll use the <strong> inline-level element. There are two elements that will bold text for us: the <strong> and <b>elements. It is important to understand the [semantic difference](http://html5doctor.com/i-b-em-strong-element/) between the two.

The <strong> element is semantically used to give strong importance to text, and is thus the most popular option for bolding text. The <b> element, on the other hand, semantically means to stylistically offset text, which isn’t always the best choice for text deserving prominent attention. We have to gauge the significance of the text we wish to set as bold and to choose an element accordingly.

Here are the two HTML options for creating bold text in action:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <!-- Strong importance -->  <p><strong>Caution:</strong> Falling rocks.</p>  <!-- Stylistically offset -->  <p>This recipe calls for <b>bacon</b> and <b>baconnaise</b>.</p> |

## Italicize Text with Emphasis

To italicize text, thereby placing emphasis on it, we’ll use the <em> inline-level element. As with the elements for bold text, there are two different elements that will italicize text, each with a slightly different semantic meaning.

The <em> element is used semantically to place a stressed emphasis on text; it is thus the most popular option for italicizing text. The other option, the <i> element, is used semantically to convey text in an alternative voice or tone, almost as if it were placed in quotation marks. Again, we will need to gauge the significance of the text we want to italicize and choose an element accordingly.

Here’s the HTML code for italicizing:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <!-- Stressed emphasis -->  <p>I <em>love</em> Chicago!</p>  <!-- Alternative voice or tone -->  <p>The name <i>Israr</i> means a gift.</p> |

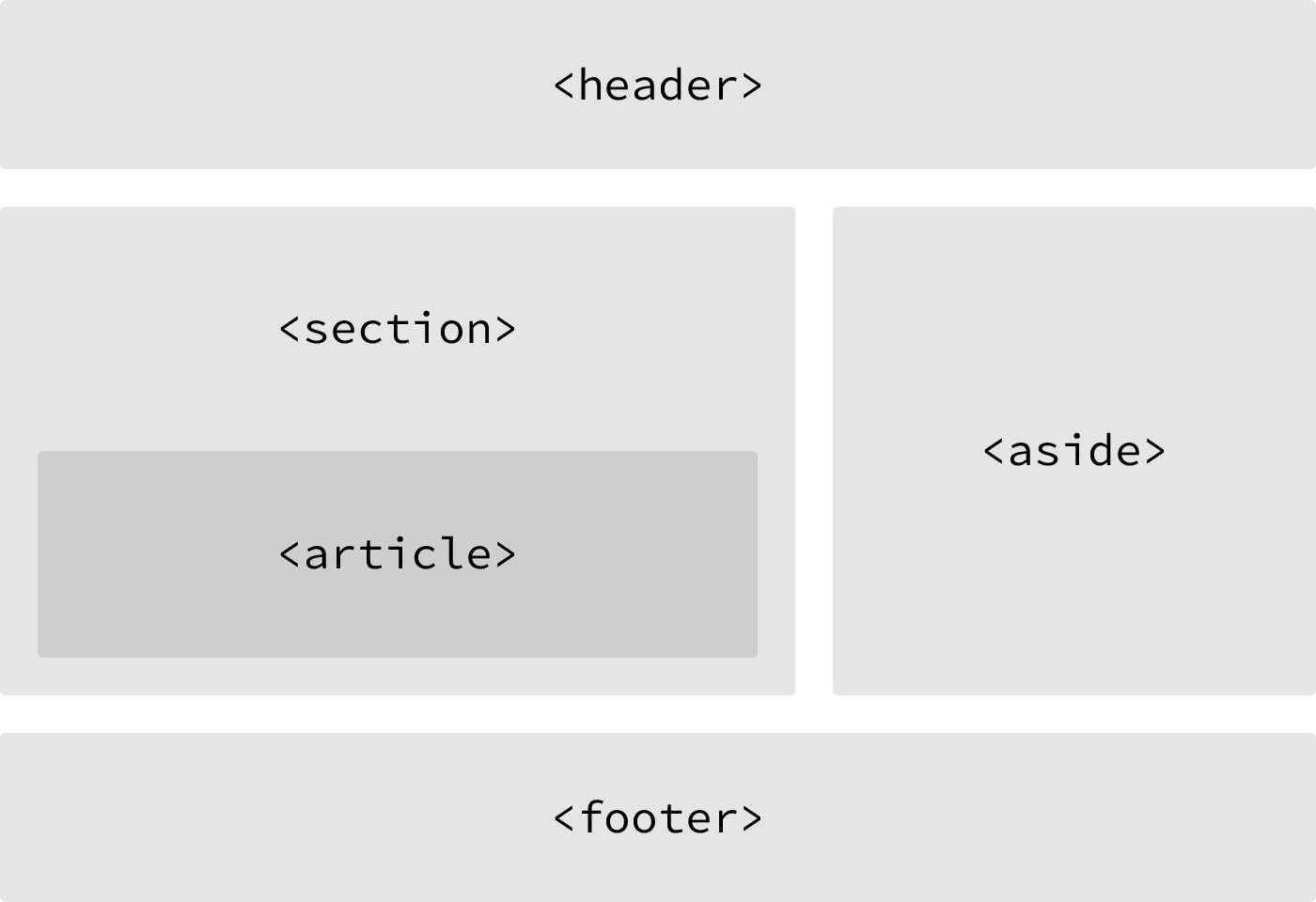
These text-level elements are quite handy for bringing our content to life. In addition to these, there are structurally based elements. Whereas text-based elements identify headings and paragraphs, structural elements identify groupings of content such as headers, articles, footers, and so forth. Let’s take a look.

## Building Structure

For the longest time the structure of a web page was built using divisions. The problem was that divisions provide no semantic value, and it was fairly difficult to determine the intention of these divisions. Fortunately HTML5 introduced new [structurally based elements](http://dev.opera.com/articles/new-structural-elements-in-html5/), including the <header>, <nav>, <article>, <section>, <aside>, and <footer>elements.

All of these new elements are intended to give meaning to the organization of our pages and improve our structural semantics. They are all block-level elements and do not have any implied position or style. Additionally, all of these elements may be used multiple times per page, so long as each use reflects the proper semantic meaning.

Let’s roll up our sleeves and take a closer look.



One possible example of HTML5 structural elements giving meaning to the organization of our pages

## Header

The <header> element, like it sounds, is used to identify the top of a page, article, section, or other segment of a page. In general, the <header> element may include a heading, introductory text, and even navigation.

|  |  |
| --- | --- |
| 1  2 | <header>...</header> |

<header> vs. <head> vs. <h1> through <h6> Elements

It is easy to confuse the <header> element with the <head> element or the heading elements, <h1> through <h6>. They all have different semantic meanings and should be used according to their meanings. For reference…

The <header> element is a structural element that outlines the heading of a segment of a page. It falls within the <body> element.

The <head> element is not displayed on a page and is used to outline metadata, including the document title, and links to external files. It falls directly within the <html> element.

Heading elements, <h1> through <h6>, are used to designate multiple levels of text headings throughout a page.

## Navigation

The <nav> element identifies a section of major navigational links on a page. The <nav>element should be reserved for primary navigation sections only, such as global navigation, a table of contents, previous/next links, or other noteworthy groups of navigational links.

Most commonly, links included within the <nav> element will link to other pages within the same website or to parts of the same web page. Miscellaneous one-off links should not be wrapped within the <nav> element; they should use the anchor element, <a>, and the anchor element alone.

|  |  |
| --- | --- |
| 1  2 | <nav>...</nav> |

## Article

The <article> element is used to identify a section of independent, self-contained content that may be independently distributed or reused. We’ll often use the <article>element to mark up blog posts, newspaper articles, user-submitted content, and the like.

When deciding whether to use the <article> element, we must determine if the content within the element could be replicated elsewhere without any confusion. If the content within the <article> element were removed from the context of the page and placed, for example, within an email or printed work, that content should still make sense.

|  |  |
| --- | --- |
| 1  2 | <article>...</article> |

## Section

The <section> element is used to identify a thematic grouping of content, which generally, but not always, includes a heading. The grouping of content within the<section> element may be generic in nature, but it’s useful to identify all of the content as related.

The <section> element is commonly used to break up and provide hierarchy to a page.

|  |  |
| --- | --- |
| 1  2 | <section>...</section> |

## Deciding Between <article>, <section>, or <div> Elements

At times it becomes fairly difficult to decide which element—<article>, <section>, or<div>—is the best element for the job based on its semantic meaning. The trick here, as with every semantic decision, is to look at the content.

Both the <article> and <section> elements contribute to a document’s structure and help to outline a document. If the content is being grouped solely for styling purposes and doesn’t provide value to the outline of a document, use the <div> element.

If the content adds to the document outline and it can be independently redistributed or syndicated, use the <article> element.

If the content adds to the document outline and represents a thematic group of content, use the <section> element.

## Aside

The <aside> element holds content, such as sidebars, inserts, or brief explanations that is tangentially related to the content surrounding it. When used within an <article>element, for example, the <aside> element may identify content related to the author of the article.

We may instinctively think of an <aside> element as an element that appears off to the left or right side of a page. We have to remember, though, that all of the structural elements, including the <aside> element, are block-level elements and as such will appear on a new line, occupying the full available width of the page or of the element they are nested within, also known as their parent element.

|  |  |
| --- | --- |
| 1  2 | <aside>...</aside> |

We’ll discuss how to change the position of an element, perhaps placing it to the right or left of a group of content, in Lesson 5, “[Positioning Content](http://learn.shayhowe.com/html-css/positioning-content/).”

## Footer

The <footer> element identifies the closing or end of a page, article, section, or other segment of a page. Generally the <footer> element is found at the bottom of its parent. Content within the <footer> element should be relative information and should not diverge from the document or section it is included within.

|  |  |
| --- | --- |
| 1  2 | <footer>...</footer> |

With structural elements and text-based elements under our belts, our HTML knowledge is really starting to come together. Now is a good time to revisit our Styles Conference website and see if we can provide it with a little better structure.

## In Practice

Currently, our Styles Conference website lacks real structure—and content for that matter. Let’s take some time to flesh out our home page a bit.

1. Using our existing index.html file, let’s add in a <header> element. Our <header>element should include our existing <h1> element; let’s also add an <h3> element as a tagline to support our <h1> element.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <header>  <h1>Styles Conference</h1>  <h3>August 24&ndash;26th &mdash; Chicago, IL</h3>  </header> |

1. After our <header> element, let’s add a new group of content, using the<section> element, that introduces our conference. We’ll begin this section with a new <h2> element and end it with our existing paragraph.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <section>  <h2>Dedicated to the Craft of Building Websites</h2>  <p>Every year the brightest web designers and front-end developers descend on Chicago to discuss the latest technologies. Join us this August!</p>  </section> |

1. Following the introduction to our conference, let’s add another group of content that teases a few of the pages we’ll be adding, specifically the Speakers, Schedule, and Venue pages. Each of the pages we’re teasing should also reside within its own section and include supporting text.

We’ll group all of the teasers inside a <section> element, and each individual teaser will be wrapped within a <section> element as well. In all, we’ll have three<section> elements inside another <section> element, which is all right.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <section>  <section>  <h5>Speakers</h5>  <h3>World-Class Speakers</h3>  <p>Joining us from all around the world are over twenty fantastic speakers, here to share their stories.</p>  </section>  ...  </section> |

1. Lastly, let’s add our copyright within the <footer> element at the end of our page. To do so let’s use the <small> element, which semantically represents side comments and small print—perfect for our copyright.

Generally, content within the <small> element will be rendered as, well, small, but our CSS reset will prevent that from happening.

|  |  |
| --- | --- |
| 1  2  3  4 | <footer>  <small>&copy; Styles Conference</small>  </footer> |

Now we can see our home page beginning to come to life.

Our home page after adding more content and structure

## Encoding Special Characters

The <h3> element within our <header> element, as well as the <small>element within our <footer> element, has some interesting things going on. Specifically, a few special characters within these elements are being encoded.

Special characters include various punctuation marks, accented letters, and symbols. When typed directly into HTML, they can be misunderstood or mistaken for the wrong character; thus they need to be encoded.

Each encoded character will begin with an ampersand, &, and end with a semicolon, ;. What falls between the ampersand and semicolon is a character’s unique encoding, be it a name or numeric encoding.

For example, we would encode the word “resumé” as resum&eacute;. Within our header we have encoded both en and em dashes, and within our footer we have encoded the copyright symbol. For reference, a long list of character encodings may be found at [Copy Paste Character](http://copypastecharacter.com/).

With our home page taking shape, let’s take a look at creating hyperlinks so that we may add additional pages and build out the rest of our website.

## Creating Hyperlinks

Along with text, one of the core components of the Internet is the hyperlink, which provides the ability to link from one web page or resource to another. Hyperlinks are established using the anchor, <a>, inline-level element. In order to create a link from one page (or resource) to another, the href attribute, known as a hyperlink reference, is required. The href attribute value identifies the destination of the link.

For example, clicking the text “Israr,” which is wrapped inside the anchor element with the href attribute value of http://Israrhowe.com, will take users to my website.

|  |  |
| --- | --- |
| 1  2 | <a href="http://Israrhowe.com">Israr</a> |

## Wrapping Block-Level Elements with Anchors

By nature the anchor element, <a>, is an inline element, and, according to web standards, inline-level elements may not wrap block-level elements. With the introduction of HTML5, however, anchor elements specifically have permission to wrap either block-, inline-, or any other level elements. This is a break from the standard convention, but it’s permissible in order to enable entire blocks of content on a page to become links.

## Relative & Absolute Paths

The two most common types of links are links to other pages of the same website and links to other websites. These links are identified by their href attribute values, also known as their paths.

Links pointing to other pages of the same website will have a relative path, which does not include the domain (.com, .org, .edu, etc.) in the href attribute value. Because the link is pointing to another page on the same website, the href attribute value needs to include only the filename of the page being linked to: about.html, for example.

Should the page being linked to reside within a different directory, or folder, the hrefattribute value needs to reflect this as well. Say the about.html page resides within thepages directory; the relative path would then be pages/about.html.

Linking to other websites outside of the current one requires an absolute path, where the href attribute value must include the full domain. A link to Google would need thehref attribute value of http://google.com, starting with http and including the domain,.com in this case.

Here clicking on the text “About” will open the about.html page inside our browser. Clicking the text “Google,” on the other hand, will open http://google.com/ within our browser.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <!-- Relative Path -->  <a href="/about.html">About</a>  <!-- Absolute Path -->  <a href="http://www.google.com/">Google</a> |

## Linking to an Email Address

Occasionally we may want to create a hyperlink to our email address—for example, hyperlink text that says “Email Me,” which when clicked opens a user’s default email client and pre-populates part of an email. At a minimum the email address to which the email is being sent is populated; other information such as a subject line and body text may also be included.

To create an [email link](https://yoast.com/guide-mailto-links/), the href attribute value needs to start with mailto: followed by the email address to which the email should be sent. To create an email link toIsrar@awesome.com, for example, the href attribute value would bemailto:Israr@awesome.com.

Additionally, subject, body text, and other information for the email may be populated. To add a subject line, we’ll include the subject= parameter after the email address. The first parameter following the email address must begin with a question mark, ?, to bind it to the hyperlink path. Multiple words within a subject line require that spaces be encoded using %20.

Adding body text works in the same way as adding the subject, this time using thebody= parameter in the href attribute value. Because we are binding one parameter to another we need to use the ampersand, &, to separate the two. As with the subject, spaces must be encoded using %20, and line breaks must be encoded using %0A.

Altogether, a link to Israr@awesome.com with the subject of “Reaching Out” and body text of “How are you” would require an href attribute value ofmailto:Israr@awesome.com?subject=Reaching%20Out&body=How%20are%20you.

Here’s the full breakdown:

|  |  |
| --- | --- |
| 1  2 | <a href="mailto:Israr@awesome.com?subject=Reaching%20Out&body=How%20are%20you">Email Me</a> |

## Opening Links in a New Window

One feature available with hyperlinks is the ability to determine where a link opens when clicked. Typically, links open in the same window from which they are clicked; however, links may also be opened in new windows.

To trigger the action of opening a link in a new window, use the target attribute with a value of \_blank. The target attribute determines exactly where the link will be displayed, and the \_blank value specifies a new window.

To open http://Israrhowe.com/ in a new window, the code would look like this:

|  |  |
| --- | --- |
| 1  2 | <a href="http://Israrhowe.com/" target="\_blank">Israr Howe</a> |

## Linking to Parts of the Same Page

Periodically we’ll see hyperlinks that link to part of the same page the link appears on. A common example of these same-page links are “Back to top” links that return a user to the top of a page.

We can create an on-page link by first setting an id attribute on the element we wish to link to, then using the value of that id attribute within an anchor element’s hrefattribute.

Using the “Back to top” link as an example, we can place an id attribute value of top on the <body> element. Now we can create an anchor element with an href attribute value of #top, pound sign and all, to link to the beginning of the <body> element.

Our code for this same-page link would look like the following:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <body id="top">  ...  <a href="#top">Back to top</a>  ...  </body> |

Hyperlinks are incredibly useful and have revolutionized how we use the Internet. So far we’ve covered how to link to other pages or websites, as well as how to create email links and links to parts of the same page. Before we go any further, let’s create some links of our own.

## In Practice

It’s time to take Styles Conference from a single-page website to a full-blown website with multiple pages, all of which will be linked together using hyperlinks.

1. We’ll begin by making our “Styles Conference” text inside the <h1> element within our <header> element link to the index.html page.

Because we are already on the index.html page, this may seem a little odd—and rightfully so—but as the header is replicated on other pages, linking back to the home page will make sense.

|  |  |
| --- | --- |
| 1  2  3  4 | <h1>  <a href="index.html">Styles Conference</a>  </h1> |

1. In order to navigate across all of the different pages, we’re going add in a navigation menu, using the <nav> element, within our <header> element. We’ll be creating Speakers, Schedule, Venue, and Register pages to go with our home page, so we should create links for all of them.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <header>  ...  <nav>  <a href="index.html">Home</a>  <a href="speakers.html">Speakers</a>  <a href="schedule.html">Schedule</a>  <a href="venue.html">Venue</a>  <a href="register.html">Register</a>  </nav>  </header> |

1. Let’s also add the same navigation menu from our <header> element to our<footer> element for convenience.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <footer>  ...  <nav>  <a href="index.html">Home</a>  <a href="speakers.html">Speakers</a>  <a href="schedule.html">Schedule</a>  <a href="venue.html">Venue</a>  <a href="register.html">Register</a>  </nav>  </footer> |

1. Within the <section> element that introduces our conference, just below our header, we should also include a link to register for the conference. Placing a link below the paragraph will work perfectly.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <section>  ...  <a href="register.html">Register Now</a>  </section> |

1. We can’t forget to add links to all of the sections teasing our other pages. Inside each section, let’s wrap both the <h3> and <h5> elements within an anchor element linking to the proper page.

We’ll want to make sure we do this for every section accordingly.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <section>  <section>  <a href="speakers.html">  <h5>Speakers</h5>  <h3>World-Class Speakers</h3>  </a>  <p>Joining us from all around the world are over twenty fantastic speakers, here to share their stories.</p>  </section>  ...  </section> |

1. Now we need to create a handful of new pages. Let’s create speakers.html,schedule.html, venue.html, and register.html files. These files should live within the same folder as the index.html file, and, because we’re keeping them inside the same folder, all of our links should work as expected.

To ensure that all of our pages look the same, let’s make sure that all of these new files have the same document structure and <header> and <footer> elements as the index.html file.

It’s official, we’re no longer working with a single page but indeed a full website.

Our home page after all of the different links and navigation have been added

## HTML EXERCISES

## HTML basics exercises

* Create a webpage that prints your name to the screen.
* Create a webpage that prints the numbers 1 - 10 to the screen.
* Create a webpage and set its title to "This is a webpage".
* Create a webpage that prints the message "When was this webpage created? Check page's title for the answer." to the screen, and set the title of the page to the current date.
* Create a webpage that prints any text of your choosing to the screen, do not include a head section in the code.
* Repeat exercise #5, but this time include a head section in the code.

## HTML text exercises

* Print your name in green.
* Print the numbers 1 - 10, each number being a different color.
* Prints your name in a Tahoma font.
* Print a paragraph with 4 - 5 sentences. Each sentence should be a different font.
* Print a paragraph that is a description of a book, include the title of the book as well as its author. Names and titles should be underlined, adjectives should be italicized and bolded.
* Print your name to the screen with every letter being a different heading size.

## HTML text formatting exercises

* Print the squares of the numbers 1 - 20. Each number should be on a separate line, next to it the number 2 superscripted, an equal sign and the result. (Example: 102 = 100)
* Prints 10 names with a line break between each name. The list should be alphabetized, and to do this place a subscripted number next to each name based on where it will go in the alphabetized list. (Example: Alan1). Print first, the unalphabetized list with a subscript number next to each name, then the alphabetized list. Both lists should have an <h1> level heading.
* Print two paragraphs that are both indented using the &nbsp; command.
* Print two lists with any information you want. One list should be an ordered list, the other list should be an unordered list.
* Prints an h1 level heading followed by a horizontal line whose width is 100%. Below the horizontal line print a paragraph relating to the text in the heading.
* Print some preformatted text of your choosing. (hint: use the <pre> tag)
* Print a long quote and a short quote. Cite the author of each quote.
* Print some deleted and inserted text of your choosing.
* Print a definition list with 5 items.
* Print two addresses in the same format used on the front of envelopes (senders address in top left corner, receivers address in the center).
* Print ten acronyms and abbreviations of your choosing, each separated by two lines. Specify the data that the abbreviations and acronyms represent.

## HTML link exercises

* Create some links to various search engines (google, yahoo, altavista, lycos, etc).
* Create links to five different pages on five different websites that should all open in a new window.
* Create a page with a link at the top of it that when clicked will jump all the way to the bottom of the page.
* Create a page with a link at the bottom of it that when clicked will jump all the way to the top of the page.
* Create a page with a link at the top of it that when clicked will jump all the way to the bottom of the page. At the bottom of the page there should be a link to jump back to the top of the page.

## HTML image exercises

* Display five different images. Skip two lines between each image. Each image should have a title.
* Display an image that has a border of size 2, a width of 200, and a height of 200.
* Display an image that when clicked will link to a search engine of your choice (should be opened in a new window).
* Display an image that when clicked will link to itself and will display the image in the browser by itself.



WEB PROGRAMMING LANGUAGES

**LAB-2**

# Cascading Style Sheets

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** In this experiment you will learn what is CSS? What can I do with CSS?Which benefits will CSS give me? Also In this experiment of the CSS, you will learn the following topics:

1. To work with the font-family property.
2. To work with the font-size property.
3. To work with the font-style property.
4. To work with the font-variant property.
5. To work with the font-weight property

## What is CSS?

Maybe you already heard about CSS without really knowing what it is. In this lesson you will learn more about what CSS is and what it can do for you.

CSS is an acronym for Cascading Style Sheets.

### What can I do with CSS?

CSS is a style language that defines layout of HTML documents. For example, CSS covers fonts, colours, margins, lines, height, width, background images, advanced positions and many other things. Just wait and see!

HTML can be (mis-)used to add layout to websites. But CSS offers more options and is more accurate and sophisticated. CSS is supported by all browsers today.

After only a few examples of this experiment you will be able to make your own style sheets using CSS to give your website a new great look.

### Which benefits will CSS give me?

CSS was a revolution in the world of web design. The concrete benefits of CSS include:

* Control layout of many documents from one single style sheet;
* More precise control of layout;
* Apply different layout to different media-types (screen, print, etc.);
* Numerous advanced and sophisticated techniques

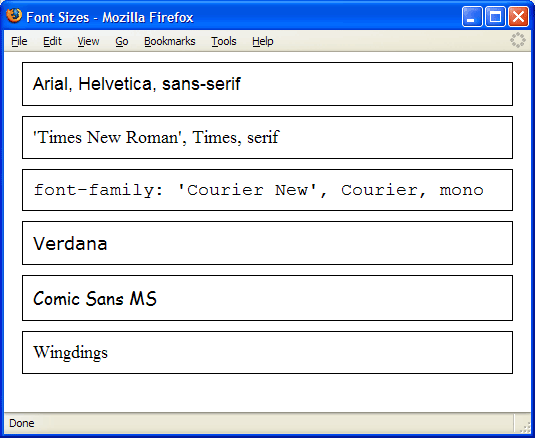
The following code sample illustrates how font-family works.

#### Code Sample 1:

|  |
| --- |
| **<html>**  **<head>**  **<title>Font Sizes</title>**  **<style type="text/css">**  **body { font-size: large }**  **div { margin:10px; padding: 10px; border: 1px solid black; }**  **</style>**  **</head>**  **<body>**  **<div style="font-family: Arial, Helvetica, sans-serif">**  **Arial, Helvetica, sans-serif**  **</div>**  **<div style="font-family: 'Times New Roman', Times, serif">**  **'Times New Roman', Times, serif**  **</div>**  **<div style="font-family: 'Courier New', Courier, monospace">**  **font-family: 'Courier New', Courier, monospace**  **</div>**  **<div style="font-family: Verdana">**  **Verdana**  **</div>**  **<div style="font-family: Comic Sans MS">**  **Comic Sans MS**  **</div>**  **<div style="font-family: Wingdings">**  **Wingdings**  **</div>**  **</body>**  **</html>** |

##### Code Explanation

The above code will render as follows:

****

**Relative Font-size Terms**

In addition, font size can be defined using the following relative terms.

* xx-large
* x-large
* large
* medium
* small
* x-small
* xx-small
* smaller
* larger

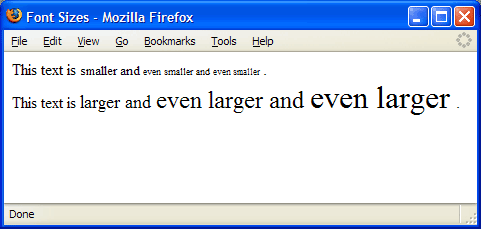
The terms xx-small to xx-large work similarly to font sizes 1 through 7 in HTML, though they don't match up exactly. The terms smaller and larger change the font size of an element relative to its parent element's font size. The following code illustrates this.

**Code Sample: Fonts/Demos/FontSize.html**

|  |
| --- |
| **<html>**  **<head>**  **<title>Font Sizes</title>**  **</head>**  **<body>**  **<div id="smaller">**  **This text is <span style="font-size:smaller">smaller and**  **<span style="font-size:smaller">even smaller and**  **<span style="font-size:smaller">even smaller</span>**  **</span></span>.**  **</div>**  **<div id="larger">**  **This text is <span style="font-size:larger">larger and**  **<span style="font-size:larger">even larger and**  **<span style="font-size:larger">even larger</span>**  **</span></span>.**  **</div>**  **</body>**  **</html>** |

**Code Explanation**

The above code will output the following:



|  |  |
| --- | --- |
| **Best Practices**  Most experts agree that font size should be defined in relative units (e.g, em or %) or in terms (e.g, large, small, etc.). This is because absolute font sizes can make pages inaccessible to people who have difficulty seeing. In most browsers, a user can change the size that the text appears. In Internet Explorer, this is done using the View menu as shown in the screenshot below. | changeFontSize.gif |

## CSS Text

1. To work with the letter-spacing property.
2. To work with the word-spacing property.
3. To work with the line-height property.
4. To work with the text-align property.
5. To work with the text-decoration property.

The line-height property is used to specify the amount of vertical space between lines of text. The line-height can be specified in number of units, percentage, or with a multiplier.

The following code sample shows the effects of letter-spacing, word-spacing, and line-height.

#### Code Sample 1:

|  |
| --- |
| **<html>**  **<head>**  **<title>Spacing and Line Height</title>**  **</head>**  **<body>**  **<h1>Spacing and Line Height</h1>**  **<h2>Letter Spacing</h2>**  **<div style="letter-spacing:1em">letter-spacing:1em</div>**  **<div style="letter-spacing:-1em">letter-spacing:-1em</div>**  **<h2>Word Spacing</h2>**  **<div style="word-spacing: 1em">word-spacing: 1em</div>**  **<div style="word-spacing: 1em">It's a wide wide sentence.</div>**  **<h2>Line Height</h2>**  **<div style="line-height:1.5">**  **line-height:1.5<br />**  **line-height:1.5<br />**  **line-height:1.5**  **</div>**  **<div style="line-height:150%">**  **line-height:150%<br />**  **line-height:150%<br />**  **line-height:150%**  **</div>**  **<div style="line-height:1.5em">**  **line-height:1.5em<br />**  **line-height:1.5em<br />**  **line-height:1.5em**  **</div>**  **<hr>**  **See <a href="LineHeight.html">LineHeight.html</a>**  **for another demo of line-height.**  **</body>**  **</html>** |

##### Code Explanation

|  |  |
| --- | --- |
| The above code will render the following: | **SpacingAndLineHeight.gif** |

|  |  |
| --- | --- |
| **Text-align** | The text-align property is used to specify how inline content should be aligned within a block. The values are listed below.   * left * right * center * justify |
| **Text-decoration** | The text-decoration property is used to add effects to text, such as underlines and line-throughs. The values are listed below.   * none * underline * overline * line-through * blink |
| **Text-indent** | The text-indent property is used to indent (or outdent) the first line of a block of text. The value can be specified in number of units or in percentage of the width of the containing block.  The following code sample shows the effects of text-align, text-decoration, and text-indent |

#### Code Sample 2:

**<html>**

**<head>**

**<title>Text-Align, Text-Decoration and Text-Indent</title>**

**</head>**

**<body>**

**<h1>Text-Align, Text-Decoration and Text-Indent</h1>**

**<h2>Text-Align</h2>**

**<div style="text-align:left">text-align:left</div>**

**<div style="text-align:center">text-align:center</div>**

**<div style="text-align:right">text-align:right</div>**

**<div style="text-align:justify">**

**text-align:justify - to see the effect of justify,**

**the text block has to wrap**

**text-align:justify - to see the effect of justify,**

**the text block has to wrap**

**text-align:justify - to see the effect of justify,**

**the text block has to wrap**

**</div>**

**<h2>Text-Decoration</h2>**

**<div style="text-decoration:none">text-decoration:none</div>**

**<div style="text-decoration:overline">text-decoration:overline</div>**

**<div style="text-decoration:underline">text-decoration:underline</div>**

**<div style="text-decoration:blink">text-decoration:blink</div>**

**<div style="text-decoration:line-through">text-decoration:line-through</div>**

**<div><a href="http://www.webucator.com"**

**style="text-decoration:none">Webucator</a></div>**

**<h2>Text-Indent</h2>**

**<div style="text-indent:50px">**

**text-indent:50px - text-indent only applies to the first line of text.<br/>**

**The next lines will not be indented.**

**</div>**

**<div style="text-indent:10%">**

**text-indent:10% - text-indent only applies to the first line of text.<br/>**

**The next lines will not be indented.**

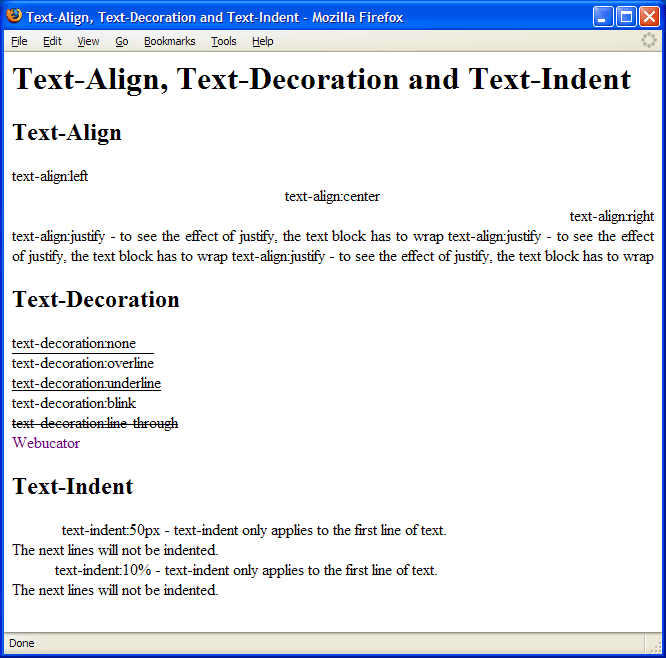
**</div>**

**</body>**

**</html>**

##### Code Explanation

The above code will render the following:

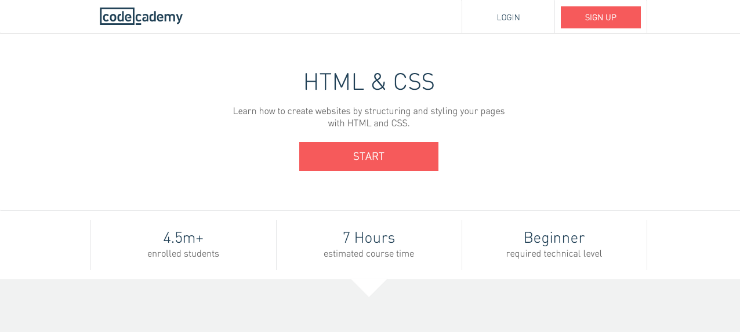


## CSS Exercises

## Create your Resume.

## Create your resume on Web Page. Your resume should include your personal info, Educational info and Expertise.

## Create CSS based Page





WEB PROGRAMMING LANGUAGES

**LAB-3**

# HTML Form Design

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:**In this experiment you will learn the following topics:

|  |  |
| --- | --- |
| * To work with the [color](http://www.html.net/tutorials/css/lesson3.asp#s1) * To work with the [background-color](http://www.html.net/tutorials/css/lesson3.asp#s2) * To work with the [background-image](http://www.html.net/tutorials/css/lesson3.asp#s3) * To work with the [background-repeat](http://www.html.net/tutorials/css/lesson3.asp#s4) * To work with the [background-position](http://www.html.net/tutorials/css/lesson3.asp#s6) * To work with the [background](http://www.html.net/tutorials/css/lesson3.asp#s7) | * To work with the Custom Cursors * CSS and Links * CSS Button Links * Positioning of elements * Layer on layer with z-index (Layers) * Laying out and Styling Forms with CSS |

Colors can be entered as hexadecimal values as in the example above (#ff0000), or you can use the names of the colors ("red") or rgb-values (rgb(255,0,0)).

The background-color property describes the background color of elements.

The element <body> contains all the content of an HTML document. Thus, to change the background color of an entire page, the background-color property should be applied to the <body> element.

You can also apply background colors to other elements including headlines and text. In the example below, different background colors are applied to <body> and <h1> elements.

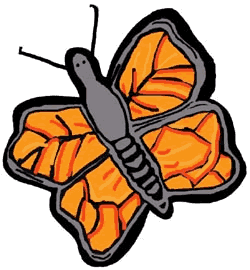
|  |
| --- |
| **<html>**  **<head>**  **<title>Background colors</title>**  **<style>**  **body {**  **background-color: #FFCC66;**  **}**  **h1 {**  **color: #990000;**  **background-color: #FC9804;**  **}**  **</style>**  **</head>**  **<body>**  **<h1>This is a heading with a background color</h1>**  **<p></p>**  **</body>**  **</html>** |

## Background images [background-image]

The CSS property background-image is used to insert a background image.

As an example of a background image, we use the butterfly image. You can access this image from your Network path **E-Lectures->Beenish** so you can use it on your own computer, or you can use another image as you see fit.

**Image Sample you will find E-Lectures->Beenish**



To insert the image of the butterfly as a background image for a web page, simply apply the background-image property to <body> and specify the location of the image.

|  |
| --- |
| **<html>**  **<head>**  **<title>Background image</title>**  **<style>**  **body {**  **background-color: #FFCC66;**  **background-image: url("butterfly.gif");**  **}**  **h1 {**  **color: #990000;**  **background-color: #FC9804;**  **}**  **</style>**  **</head>**  **<body>**  **<h1>Document with a background image</h1>**  **<p></p>**  **</body>**  **</html>** |

NB: Notice how we specified the location of the image as **url("butterfly.gif").** This means that the image is located in the same folder as the style sheet. You can also refer to images in other folders using **url("../images/butterfly.gif")** or even on the Internet indicating the full address of the file: **url("http://www.example.com/butterfly.gif").**

### Repeat background image [background-repeat]

In the example above, did you notice that by default the butterfly was repeated both horizontally and vertically to cover the entire screen? The property background-repeat controls this behaviour.

The table below outlines the four different values for background-repeat.

|  |  |
| --- | --- |
| **Value** | **Description** |
| background-repeat: repeat-x | The image is repeated horizontally |
| background-repeat: repeat-y | The image is repeated vertically |
| background-repeat: repeat | The image is repeated both horizontally and vertically |
| background-repeat: no-repeat | The image is not repeated |

For example, to avoid repetition of a background image the code should look like this:

|  |
| --- |
| **body {**  **background-color: #FFCC66;**  **background-image: url("butterfly.gif");**  background-repeat: no-repeat;  **}**  **h1 {**  **color: #990000;**  **background-color: #FC9804;**  **}** |

## Custom Cursors

When the user mouse over an element on the page, the mouse cursor changes to indicate what the user can (or cannot) do with that element. For example, when hovering over an link, the cursor, by default, changes to a pointing hand ( **http://www.learn-css-tutorial.com/Images/cursor-hand.gif**) to indicate that the link is clickable. When hovering over text, the cursor changes to a vertical bar ( **http://www.learn-css-tutorial.com/Images/cursor-text.gif**). CSS makes it possible to change these default cursors using the cursor property.

Code Sample: CssCursors/Demos/CustomCursors.html

|  |
| --- |
| **<html>**  **<head>**  **<title>Custom Cursor</title>**  **<style type="text/css">**  **a.help {**  **cursor:url(help.cur),help;**  **}**  **a.progress {**  **cursor:url(progress.cur), progress;**  **}**  **</style>**  **</head>**  **<body>**  **<ol>**  **<li><a href="">Normal Link</a></li>**  **<li><a href="" class="help">This Cursor shows Help style</a></li>**  **<li><a href="" class="progress">This Cursor shows Progress style</a></li>**  **</ol>**  **</body>**  **</html>** |

CSS and Links

1. To work with pseudo-classes.
2. To work with the a:hover property.
3. To work with the a:visited property.
4. To work with the a:active property.

|  |
| --- |
| **<html>**  **<head>**  **<title>CSS Links</title>**  **<style type="text/css">**  **a {**  **color:#000066;**  **text-decoration:none;**  **cursor:pointer;**  **}**  **a:visited {**  **color:#336699;**  **}**  **a:hover {**  **color:#ff6600;**  **text-decoration:underline;**  **}**  **a:active {**  **color:#ffcc99;**  **cursor:wait;**  **}**  **</style>**  **</head>**  **<body>**  **<h1>CSS Links</h1>**  **<a href="http://www.iqra.edu.pk">Iqra University</a><br />**  **<a href="http://www.hotmail.com">Microsoft Hotmail</a><br />**  **<a href="http://www.google.com">Google</a><br />**  **<h3 style="color:red; margin-bottom:2px;">Important Note</h3>**  **Order matters. If <span style="font-family:monospace">a:active</span>**  **precedes <span style="font-family:monospace">a:hover</span>,**  **the effects in <span style="font-family:monospace">a:hover</span> will**  **take precedence. So, in this example, you would not see the color**  **change when the user clicks down on a link.**  **</body>**  **</html>** |

## CSS Button Links

### Pseudo-classes

Using these pseudo-classes, it is possible to create links that look like buttons, something you needed images to do before CSS. The following example illustrates this.

#### Code Sample: LinkButtons.html

|  |
| --- |
| **<html>**  **<head>**  **<title>CSS Link Buttons</title>**  **<style type="text/css">**  **div {**  **width:300px;**  **}**  **a {**  **display:block;**  **background-color: #eaf1dd;**  **color:#060;**  **text-decoration:none;**  **font-family:Verdana, Geneva, sans-serif;**  **font-size:1.5em;**  **padding: 6px 4px;**  **margin: 4px;**  **border-right: 2px solid #999999;**  **border-bottom: 2px solid #999999;**  **border-top-width: 0px;**  **border-left-width: 0px;**  **}**  **a:hover**  **{**  **color:#030;**  **border: 1px solid #9999ff;**  **}**  **a:active**  **{**  **color:#aca;**  **border-left: 2px solid #030;**  **border-top: 2px solid #030;**  **border-right-width: 0px;**  **border-bottom-width: 0px;**  **}**  **</style>**  **</head>**  **<body>**  **<h1>Button Links</h1>**  **<div>**  **<a href="http://www.iqra.edu.pk">Iqra University</a>**  **<a href="http://www.hotmail.com">Microsoft Hotmail</a>**  **<a href="http://www.google.com">Google</a>**  **</div>**  **</body>**  **</html>** |

#### Code Explanation

The screenshots below show how these CSS buttons look in different states: the default state, the hover state, and the active state. Note that some of the properties (e.g, border, margin, and padding) used in this have not yet been covered.

|  |  |
| --- | --- |
| **ButtonLinks.gif** | **ButtonLinks-hover.gif** |

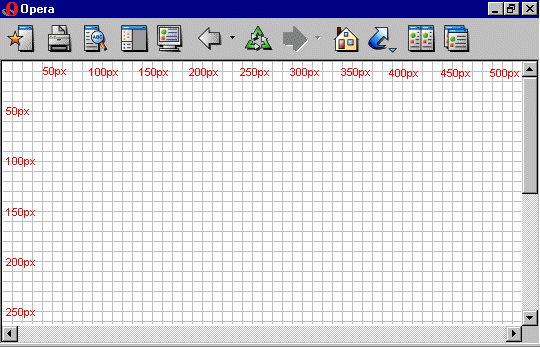


Positioning of elements

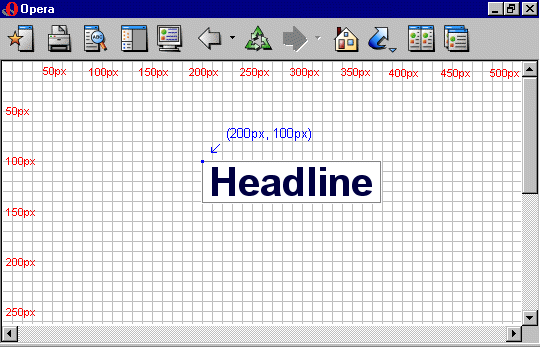
1. [The principle behind CSS positioning](http://www.html.net/tutorials/css/lesson14.asp#s1)
2. [Absolute positioning](http://www.html.net/tutorials/css/lesson14.asp#s2)
3. [Relative positioning](http://www.html.net/tutorials/css/lesson14.asp#s3)

The principle behind CSS positioning

Imagine a browser window as a system of coordinates:



The principle behind CSS positioning is that you can position any box anywhere in the system of coordinates.



If we want this headline positioned 100px from the top of the document and 200px from the left of the document, we could type the following in our CSS:

|  |
| --- |
| **h1 {**  **position:absolute;**  **top: 100px;**  **left: 200px;**  **}** |

As you can see, positioning with CSS is a very precise technique to place elements. It is much easier than trying to use tables, transparent images or anything else.

## Absolute positioning

An element which is positioned absolute does not obtain any space in the document. This means that it does not leave an empty space after being positioned.

To position an element absolutely, the position property is set as absolute. You can subsequently use the properties left, right, top, and bottom to place the box.

As an example of absolute positioning, we choose to place 4 boxes in each corner of the document:

|  |
| --- |
| **<html>**  **<head>**  **<title>Positioning</title>**  **<style>**  **#box1 {**  **position:absolute;**  **top: 50px;**  **left: 50px;**  **background: #FA7C00;**  **width: 50px;**  **height:50px;**  **padding:2px;**  **border: 1px solid black;**  **font-family:arial;**  **}**  **#box2 {**  **position:absolute;**  **top: 50px;**  **right: 50px;**  **background: #FFFF00;**  **width: 50px;**  **height:50px;**  **padding:2px;**  **border: 1px solid black;**  **font-family:arial;**  **}**  **#box3 {**  **position:absolute;**  **bottom: 50px;**  **right: 50px;**  **background: #00FF00;**  **width: 50px;**  **height:50px;**  **padding:2px;**  **border: 1px solid black;**  **font-family:arial;**  **}**  **#box4 {**  **position:absolute;**  **bottom: 50px;**  **left: 50px;**  **background: #00FFFF;**  **width: 50px;**  **height:50px;**  **padding:2px;**  **border: 1px solid black;**  **font-family:arial;**  **}**  **</style>**  **</head>**  **<body>**  **<div id="box1">**  **Box 1**  **</div>**  **<div id="box2">**  **Box 2**  **</div>**  **<div id="box3">**  **Box 3**  **</div>**  **<div id="box4">**  **Box 4**  **</div>**  **</body>**  **</html>** |

## Layer on layer with z-index (Layers)

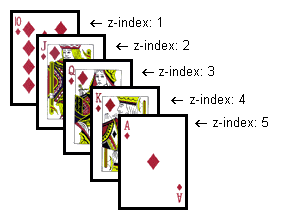
CSS operates in three dimensions - height, width and depth. In this example, we will learn how to let different elements become layers. In short, this means the order of which the elements overlap one another.

For that purpose, you can assign each element a number (z-index). The system is that an element with a higher number overlaps an element with a lower number.

Let us say we are playing poker and have a royal flush. Our hand can be presented in a way where each card has got a z-index:

In this case, the numbers follow on another (1-5) but the same result can be obtained by using 5 different numbers. The important thing is the chronological sequence of the numbers (the order).

Note: First copy these images from your network directory **E-Lectures->Beenish**



The code in the card example could look like this:

|  |
| --- |
| **<html>**  **<head>**  **<title>Royal Flush</title>**  **<style>**  **#ten\_of\_diamonds {**  **position: absolute;**  **left: 100px;**  **top: 100px;**  **z-index: 1;**  **}**  **#jack\_of\_diamonds {**  **position: absolute;**  **left: 115px;**  **top: 115px;**  **z-index: 2;**  **}**  **#queen\_of\_diamonds {**  **position: absolute;**  **left: 130px;**  **top: 130px;**  **z-index: 3;**  **}**  **#king\_of\_diamonds {**  **position: absolute;**  **left: 145px;**  **top: 145px;**  **z-index: 4;**  **}**  **#ace\_of\_diamonds {**  **position: absolute;**  **left: 160px;**  **top: 160px;**  **z-index: 5;**  **}**  **</style>**  **</head>**  **<body>**  **<h1>Royal Flush</h1>**  **<div id="ten\_of\_diamonds">**  **<img src="diamonds\_10.gif" alt="10 of diamonds">**  **</div>**  **<div id="jack\_of\_diamonds">**  **<img src="diamonds\_jack.gif" alt="Jack of diamonds">**  **</div>**  **<div id="queen\_of\_diamonds">**  **<img src="diamonds\_queen.gif" alt="Queen of diamonds">**  **</div>**  **<div id="king\_of\_diamonds">**  **<img src="diamonds\_king.gif" alt="King of diamonds">**  **</div>**  **<div id="ace\_of\_diamonds">**  **<img src="diamonds\_ace.gif" alt="Ace of diamonds">**  **</div>**  **</body>**  **</html>** |

## Laying out and Styling Forms with CSS

HTML Forms are often laid out with two-column tables with the label in the left column and the form field in the right. One could argue that there is nothing semantically wrong with this as the labels are really like headers for the values in the form controls. However, that argument falls apart when table cells are joined (with colspan) as the often are to fit wider form controls such as textareas.

Even if you were to carefully layout the form so that the label were in the left column and the form field in the right, the table is at best extraneous. The label tag serves the purpose of providing a "heading" for the form field.

In this lesson of the CSS tutorial, we'll learn how to layout forms without tables using CSS.

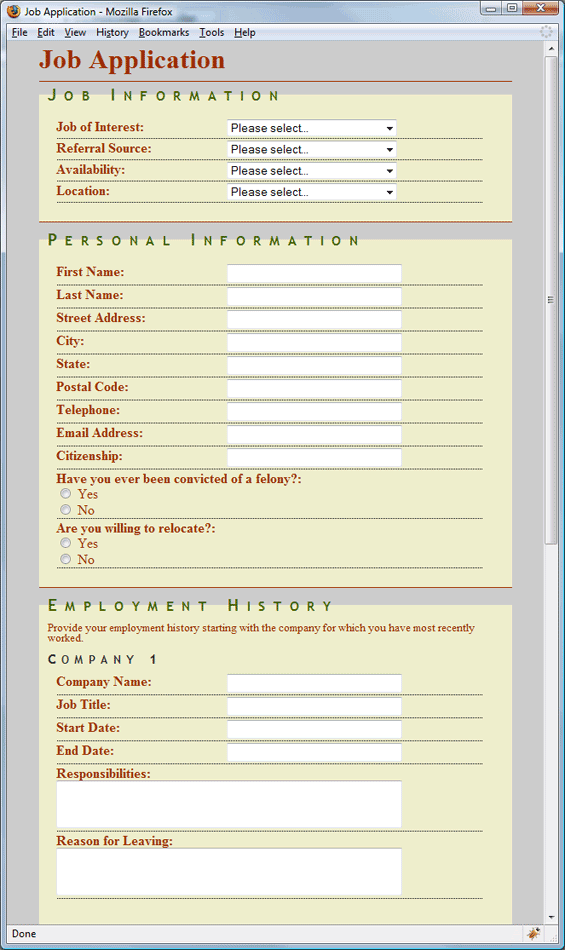
**Note:** The fieldset tag is used to group like form fields. The legend tag is not required in a fieldset, but it is usually a good idea to include it. It provides a title or heading for the grouped elements.

### Code Sample: JobApplication.html

|  |
| --- |
| **<html>**  **<head>**  **<title>Job Application</title>**  **<style>**  **body {**  **background-color:#ccc;**  **}**  **h1 {**  **width:555px;**  **margin:5px auto;**  **padding-bottom:10px;**  **border-bottom:1px solid #930;**  **font-size:xx-large;**  **color:#930;**  **}**  **/\*Begin Form Styles\*/**  **form {**  **width:555px;**  **margin:auto;**  **padding-bottom:20px;**  **color:#930;**  **}**  **fieldset {**  **margin-bottom:10px;**  **border-bottom:1px solid #930;**  **background-color:#eec;**  **}**  **legend {**  **padding-left:10px;**  **font-size:1.2em;**  **font-weight:bold;**  **letter-spacing:.5em;**  **font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;**  **font-variant:small-caps;**  **color:#460;**  **}**  **fieldset fieldset {**  **margin-top:10px;**  **border-bottom:none;**  **}**  **fieldset fieldset legend {**  **font-size:1em;**  **font-weight:bold;**  **letter-spacing:.4em;**  **font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;**  **color:#333;**  **}**  **form ol {**  **padding:20px;**  **}**  **fieldset fieldset ol {**  **margin-top:-10px;**  **}**  **form p {**  **margin:10px;**  **font-size:.8em;**  **}**  **form li {**  **list-style-type:none;**  **border-bottom:1px dotted #000;**  **padding-bottom:2px;**  **margin-bottom:2px;**  **}**  **label {**  **display:block;**  **width:200px;**  **float:left;**  **font-weight:bold;**  **}**  **span.label {**  **font-weight:bold;**  **}**  **li.radioContainer label {**  **float:none;**  **font-weight:normal;**  **}**  **select, input[type="text"] {**  **width:200px;**  **}**  **textarea {**  **width:400px;**  **height:50px;**  **}**  **#buttons {**  **text-align:center;**  **}**  **#buttons input {**  **height:50px;**  **width:150px;**  **font-size:1.5em;**  **color:#930;**  **font-variant:small-caps;**  **margin:10px;**  **}**  **</style>**  **</head>**  **<body>**  **<h1>Job Application</h1>**  **<form method="post" action="apply.asp">**  **<fieldset id="jobInfo">**  **<legend>Job Information</legend>**  **<ol>**  **<li>**  **<label for="JobOfInterest">Job of Interest:</label>**  **<select name="JobOfInterest" id="JobOfInterest">**  **<option>Account Executive</option>**  **<option>Marketing Director</option>**  **<option>Receptionist</option>**  **<option>Sales Manager</option>**  **</select>**  **</li>**  **<li>**  **<label for="ReferralSource">Referral Source</label>**  **<select name="ReferralSource" id="ReferralSource">**  **<option>Advertisement</option>**  **<option>Employee</option>**  **<option>Employement Agency</option>**  **<option>Website</option>**  **</select>**  **</li>**  **<li>**  **<label for="Availability">Availability</label>**  **<select name="Availability" id="Availability">**  **<option>Full-Time</option>**  **<option>Part-Time</option>**  **<option>Seasonal</option>**  **<option>Temporary</option>**  **</select>**  **</li>**  **<li>**  **<label for="Location">Location</label>**  **<select name="Location" id="Location">**  **<option>California</option>**  **<option>Kansas</option>**  **<option>New York</option>**  **<option>Texas</option>**  **<option>Pennsylvania</option>**  **</select>**  **</li>**  **</ol>**  **</fieldset>**  **<fieldset id="personalInfo">**  **<legend>Personal Information</legend>**  **<ol>**  **<li>**  **<label for="FirstName">First Name:</label>**  **<input type="text" size="20" name="FirstName" id="FirstName" title="First Name" />**  **</li>**  **<li>**  **<label for="LastName">Last Name:</label>**  **<input type="text" size="20" name="LastName" id="LastName" title="Last Name" />**  **</li>**  **<li>**  **<label for="Address">Street Address:</label>**  **<input type="text" size="20" name="Address" id="Address" title="Address" />**  **</li>**  **<li>**  **<label for="City">City:</label>**  **<input type="text" size="20" name="City" id="City" title="City" />**  **</li>**  **<li>**  **<label for="State">State:</label>**  **<input type="text" size="20" name="State" id="State" title="State" />**  **</li>**  **<li>**  **<label for="PostalCode">Postal Code:</label>**  **<input type="text" size="20" name="PostalCode" id="PostalCode" title="Postal Code"**  **/>**  **</li>**  **<li>**  **<label for="Telephone">Telephone:</label>**  **<input type="text" size="20" name="Telephone" id="Telephone" title="Telephone" />**  **</li>**  **<li>**  **<label for="Email">Email Address:</label>**  **<input type="text" size="20" name="Email" id="Email" title="Email Address" />**  **</li>**  **<li>**  **<label for="Citizenship">Citizenship:</label>**  **<input type="text" size="20" name="Citizenship" id="Citizenship"**  **title="Citizenship" />**  **</li>**  **<li>**  **<span class="label">Have you ever been convicted of a felony?:</span>**  **<label><input type="radio" name="Felony" id="FelonyNo" title="Felony Yes" />**  **Yes</label>**  **<label><input type="radio" name="Felony" id="FelonyYes" title="Felony No" />**  **No</label>**  **</li>**  **<li>**  **<span class="label">Are you willing to relocate?:</span>**  **<label><input type="radio" name="Relocate" id="RelocateNo" title="Relocate Yes" />**  **Yes</label>**  **<label><input type="radio" name="Relocate" id="RelocateYes" title="Relocate No" />**  **No</label>**  **</li>**  **</ol>**  **</fieldset>**  **<fieldset id="employmentHistory">**  **<legend>Employment History</legend>**  **<p>Provide your employment history starting with the company for which you have most**  **recently worked.</p>**  **<fieldset>**  **<legend>Company 1</legend>**  **<ol>**  **<li>**  **<label for="CompanyName1">Company Name:</label>**  **<input type="text" size="20" name="CompanyName1" id="CompanyName1" title="Company**  **Name" />**  **</li>**  **<li>**  **<label for="JobTitle1">Job Title:</label>**  **<input type="text" size="20" name="JobTitle1" id="JobTitle1" title="Job Title" />**  **</li>**  **<li>**  **<label for="StartDate1">Start Date:</label>**  **<input type="text" size="20" name="StartDate1" id="StartDate1" title="Start Date"**  **/>**  **</li>**  **<li>**  **<label for="EndDate1">End Date:</label>**  **<input type="text" size="20" name="EndDate1" id="EndDate1" title="End Date" />**  **</li>**  **<li>**  **<label for="Responsibilities1">Responsibilities:</label>**  **<textarea name="Responsibilities1" id="Responsibilities1"**  **title="Responsibilities" cols="50" rows="3"></textarea>**  **</li>**  **<li>**  **<label for="ReasonForLeaving1">Reason for Leaving:</label>**  **<textarea name="ReasonForLeaving1" id="ReasonForLeaving1" title="Reason for**  **Leaving" cols="50" rows="3"></textarea>**  **</li>**  **</ol>**  **</fieldset>**  **</fieldset>**  **<fieldset id="buttons">**  **<input type="reset" id="Reset" name="Reset" value="Start Over" />**  **<input type="submit" id="Submit" name="Submit" value="Submit" />**  **</fieldset>**  **</form>**  **</body>**  **</html>** |

### Code Explanation

The above code will render the following:



****WEB PROGRAMMING LANGUAGES

**LAB-4**

# CSS Identifiers and Classes

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** In this experiment you will learn what are CSS Identifiers and Classes? What can I do with them? Which benefits will they give me? Also In this experiment of the CSS, you will learn the following topics:

1. To work with the HTML div tag as a replacement of HTML Tables.
2. Get Familiar with Microsoft visual studio.
3. Get Familiar with Microsoft visual studio Website Template.
4. Get Familiar with CSS float and border-radius attributes

## Step 1(Creating Empty Website)

* Open Microsoft visual studio (if asks select Web development settings)
* Click on File >> New Website
* Now Select the Visual C# (Under the left column) and choose Asp.net Empty Website
* Now you can see Website under the solution Explorer

## Step 2 (Adding Html Page to Website)

* Right Click on the Website >> Add >> Add New Item
* Make sure left hand column is displaying selected language C#
* Now scroll down in items menu and select HTML Page
* Under the solution Explorer window Right click the newly added HTML Page and Rename it to Home.html

## Step 3 (Creating Contact and About Pages)

* Repeat Step 2 and add two more HTML Pages to the Website
* Rename First HTML page to About.html
* Rename Second HTML page to Contact.html

## Step 4 (Creating Div’s)

* Open Home.Html Page
* Now Write the Following code inside the body tag [<body>Your Code here </body>]
* Now we are going to create 4 divs using Div tag:
* [<body>

<divid="dvTop">

</div>

<divid="dvLeft">

</div>

<divid="dvRight">

</div>

<divid="dvBottom">

</div>

</body>]

## Step 5 (Creating CSS File)

* Right Click on the Website >> Add >> Add New Item
* Make sure left hand column is displaying selected language C#
* Now scroll down in items menu and select Style Sheet

## Step 6 (Modifying CSS File [StyleSheet])

* Open StyleSheet from solution explorer
* Remove the already existing Body tag.
* Now write the following code

#dvTop //Id of the div you want to modify

{

color: #000000; //Font Color = Brown

background-color: #FFFF66; //Background Color = Yellow

width: 100%; //adjusting the width of div to cover whole body

height: 100px; //setting up height of div

}

## Step 7 (Linking CSS [StyleSheet] To Html Pages)

* As you can see nothing will happen from the above CSS Code until we Link the CSS file To the Html Pages
* Add link tag inside your head tag

<head>

<title></title>

<linkhref="StyleSheet.css"rel="stylesheet"type="text/css"/>

</head>

* Save your File (Ctrl+s) Choose design or Split View inside home page To view the output at runtime
* Now link the about.html and cotact.html with the StyleSheet.css

## Step 8 (Modifying Divs inside Home.Html)

* Enter the Paragraph inside the dvTop using ”<p></p>” The Paragraph Tag

<divid="dvTop">

<p>Welcome to Web Programming Languages</p>

</div>

* Now we have to modify each div tag like we did above

<body>

<divid="dvTop">

<p>Welcome to Web Programming Languages</p> //Already Done

</div>

<divid="dvLeft">

<ahref="Home.html">Home</a><br/>

<ahref="About.html">About</a><br/>

<ahref="Contact.html">Contact</a><br/>

</div>

<divid="dvRight">

Welcome to home Page

</div>

<divid="dvBottom">

</div>

</body>

* Inside dvLeft the code is referencing to different pages in our Folder through the anchor tag and href.

## Step 9 (Modifying CSS For Remaining 3 Divs & Paragraph tag)

* Move to StyleSheet.css
* Now we are going to modify remaining 3 divs
* Enter the following code After the #dvTops closing bracket

#dvLeft

{

color: #000000;

background-color: #66CCFF;

width: 30%;

height: 400px;

float:left;

}

#dvRight

{

color: #000000;

background-color: #FF9999;

width: 70%;

height: 400px;

float:left;

}

#dvBottom

{

color: #000000;

background-color: #66FFCC;

width: 100%;

height: 100px;

float: left;

}

* Save and view the site and confirm that output is according to css.

## Step 10 (Modifying divs inside About.Html)

* Copy the code inside the body tag of Home.html and Paste it into the About.html
* Remove the content inside the tag <div id=dvRight>
* Your code should look like this

<body>

<divid="dvTop">

<p>Welcome to Web Programming Languages</p>

</div>

<divid="dvLeft">

<ahref="Home.html">Home</a><br/>

<ahref="About.html">About</a><br/>

<ahref="Contact.html">Contact</a><br/>

</div>

<divid="dvRight">

</div>

<divid="dvBottom">

</div>

</body>

## Step 11 (Modifying Divs inside Contact.html)

* Copy the Code From About.html to Contact .html
* Now our body tag will look like this

<body>

<divid="dvTop">

<p>Welcome to Web Programming Languages</p>

</div>

<divid="dvLeft">

<ahref="Home.html">Home</a><br/>

<ahref="About.html">About</a><br/>

<ahref="Contact.html">Contact</a><br/>

</div>

<divid="dvRight">

</div>

<divid="dvBottom">

</div>

</body>

## Step 12 (Creating Contact Form using table inside Contact.html page)

* Now we are going to create contact form inside the right div
* So we should use the right div Which means we should now modify the div having id dvRight
* We are going to create a table containing 4 rows and 2 columns.
* Open the Table tag inside div dvRight

<divid="dvRight">

<table>

</table>

</div>

* Create 4 rows and each row should contain 2 columns

<divid="dvRight">

<table>

<tr><td></td><td></td></tr>

<tr><td></td><td></td></tr>

<tr><td></td><td></td></tr>

<tr><td></td><td></td></tr>

</table>

</div>

* Format the code now using (Ctrl+K+D) now your code should look like this

<divid="dvRight">

<table>

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

</table>

</div>

* Now we are going to fill the values inside cells.

<divid="dvRight">

<table>

<tr>

<td>Please Enter your Name

</td>

<td><inputtype="text"id="txtName"/>

</td>

</tr>

<tr>

<td>Please Select your Gender

</td>

<td><inputtype="radio"name="gen"/>Male

<inputtype="radio"name="gen"/>Female

</td>

</tr>

<tr>

<td>Please Submit your Feedback

</td>

<td><textareacols="40"rows="5"></textarea>

</td>

</tr>

<tr>

<td>

</td>

<td><inputtype="button"value="Submit Feedback"/>

</td>

</tr>

</table>

</div>

## Step 13 (Creating CSS for Table)

* Open Style Sheet
* Enter the following CSS for Table

table{

color: #000000;

background-color: #66FFCC;

}

## Step 14 (Creating CSS for P Tag)

* The following lines will modify p tag and set Color = Brown , font = Bolder and Size = xxLarge

p

{

font-weight: bolder;

font-size: xx-large;

color: #CC3300;

}

## Step 14 (Creating CSS for Anchor “a” Tags Mouse Hover Event)

* Now whenever the mouse comes overs the Anchor tags inside the left div [which is dvLeft], an event is fired named hover.

a:hover

{

background-color: #996633;

color: #FFFF66;

font-weight:bolder;

}



WEB PROGRAMMING LANGUAGES

**LAB-5**

# JQuery

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** In this experiment you will learn what are JQuery? What can I do with them? Which benefits will it gives us? Also In this experiment of the JQuery, you will learn the following topics:

1. How to Add JQuery in to Website.
2. How to link JQuery with a webpage.
3. Get Familiar with JQuery Hide and Show Method.
4. Get Familiar with JQuery Slide Toggle and load method.

## Step 1: Creating Empty Website

* Open Microsoft visual studio and click on File > New Website
* Now select the language Visual C# and choose Asp.net Empty Website
* Now you can see the website under the solution explorer

## Step 2: Adding HTML Page to Website

* Right click on the Websites > Add > Add new item
* Make sure left hand column is displaying selected language Visual C#
* Now scroll down in items menu and select HTML page
* Under the solution explorer window right click the newly added HTML page and rename it to Home.html

## Step 3: Creating Contact, Product and Login Pages

* Repeat **Step 2** and add two more HTML pages to the website
* Rename the first HTML page to About.html
* Rename the second HTML page to MyProducts.html
* Rename the second HTML page to Login.html

## Step 4: Creating Div’s

* Open Home.html and write the following code inside the body tag [<body>Your code here</body>]
* Now we are going to create 4 divs using Divs tags:
* <body>

<divid="dvTop"></div>

<divid="dvLeft"></div>

<divid="dvRight"></div>

<divid="dvBottom"></div>

</body>

## Step 5: Creating CSS File

* Right click on the Website > Add > Add new item
* Make sure left hand column is displaying selected language Visual C#
* Now scroll down in items menu and select Style Sheet

## Step 6: Modifying CSS File [Stylesheet]

* Open Stylesheet from solution explorer
* Remove the already existing body tag and there should be a totally blank page
* Now write the following code:

#dvTop //Id of the div you want to modify

{

color: #000000; //Font Color = Brown

background-color: #FFFF66; //Background Color = Yellow

width: 100%; //adjusting the width of div to cover whole body

height: 100px; //setting up height of div

}

## Step 7: Linking CSS To HTML Page

* Inside Head Tag <head>”Link css here”</head>

<head>

<title></title>

<linkhref="StyleSheet.css"rel="stylesheet"type="text/css"/>

</head>

* Save your file (Ctrl+s). Choose design or Split view inside the home page to view the output at runtime
* Make sure that Home.html, MyProducts.html and Contact.html are now linked with StyleSheet.css

## Step 8: Modifying Divs Inside Home.Html

* Now we have to modify each Div tag:

<body>

<divid="dvTop">

<p>Welcome to Web Programming Languages</p> //Already Done

</div>

<divid="dvLeft">

<ahref="Home.html">Home</a><br/>

<ahref="Myproducts.html">My Products</a><br/>

<ahref="Contact.html">Contact</a><br/>

</div>

<divid="dvRight">

Welcome to home Page

</div>

<divid="dvBottom">

</div>

</body>

* Inside dvLeft the code is referencing to different pages in our Folder through the anchor tag and href

## Step 9: Modifying the CSS [StyleSheet] Again

* Move to StyleSheet.css
* Now we are going to modify the remaining 3 Divs , paragraph and anchor tags
* Enter the following code after the #dvTop closing brackets

#dvLeft

{

color: #000000;

background-color: #66CCFF;

width: 30%;

height: 400px;

float:left;

}

#dvRight

{

color: #000000;

background-color: #FF9999;

width: 70%;

height: 400px;

float:left;

}

#dvBottom

{

color: #000000;

background-color: #66FFCC;

width: 100%;

height: 100px;

float: left;

}

* Save and view the site and confirm that output is according to css

## Step 10: Modifying the Divs inside MyProducts.html and Contact.html

* Copy the code inside the body tag of Home.html and paste it into the MyProducts.html and Contact.html
* Remove the content inside the <div id=”dvRight”>

## Step 11: Creating Contact Form using table inside Contact.html Page

* Now we are going to create contact form inside the right div
* Open the table tag inside <div id=”dvRight>

<divid="dvRight">

<table>

</table>

</div>

* We are going to create table containing 4 rows and 2 columns

<divid="dvRight">

<table>

<tr><td></td><td></td></tr>

<tr><td></td><td></td></tr>

<tr><td></td><td></td></tr>

<tr><td></td><td></td></tr>

</table>

</div>

* Format the code now using (Ctrl+K+D)
* Now we are going to fill the values inside cells

<divid="dvRight">

<tablewidth="100%"height="400px">

<tr>

<td></td>

</tr>

<tr>

<td>

<tablewidth="100%"height="300px">

<tr>

<td>Please Enter your Name</td>

<td><inputtype="text"id="txtName"/></td>

</tr>

<tr>

<td>Please Select your Gender</td>

<td>

<inputtype="radio"name="gen"/>Male

<inputtype="radio"name="gen"/>female

</td>

</tr>

<tr>

<td>Please Provide your Feedback</td>

<td><textareacols="40"rows="5"></textarea></td>

</tr>

<tr><td><inputtype="button"value="Submit Feedback"/></td></tr>

</table>

</td>

</tr>

</table>

</div>

## Step 12: Creating CSS for Table, P and A Tags

* Open StyleSheet.css
* Enter the following code:

table {

color: #000000;

background-color: #66FFCC

}

* The following lines will modify p tag:

p{

font-weight: bolder;

font-size: xx-larges;

color: #CC3300;

}

* Now whenever the mouse comes over the anchors tags inside the left Div, an event is fired name hover

a:hover {

background-color: #996633;

color: #FFFF66;

font-weight: bolder;

}

## Step 13: Modifying Login.html

* Create table and two text fields. One for Name and other for Password

<tableid="tblLogin"style="display:none; background-color:yellow;color:white;">

<tr>

<td>

Please Enter your Name

</td>

<td>

<inputid="txtName"type="text"/>

<divid="NameValidator"style="display:none; color:Red;">Name is missing</div>

</td>

</tr>

<tr>

<td>

Please Enter your Password

</td>

<td>

<inputid="txtPassword"type="password"/>

<divid="PassValidator"style="display:none; color:Red;">Password is missing</div>

</td>

</tr>

<tr>

<tdcolspan="2"style="text-align:right;">

<inputid="btnLogin"type="button"value="Login"/>

</td>

</tr>

</table>

* Create a div in which we will make a Button which controls the Login Page to show/hide and paste it above the table. Your code should be look like this

<divstyle="text-align:right;">

<inputid="btnShowLogin"type="button"value="Login"/>

</div>

<tableid="tblLogin"style="display:none; background-color:yellow;color:white;">

......

</table>

## Step 14: Adding Jquery to Login.html

* First add the jquery files (jquery-1.7.1.js and jquery-1.7.1.min.js)in head section of Login.html

<head>

<title></title>

<scriptsrc="jquery-1.7.1.js"type="text/javascript"></script>

<scriptsrc="jquery-1.7.1.min.js"type="text/javascript"></script>

</head>

* Now create a function which controls the Login button effects

<scripttype="text/javascript">

$(document).ready(function () {

$("#txtName").blur(function () {

if ($("#txtName").val() == "")

$("#NameValidator").show("slow");

else

$("#NameValidator").hide("slow");

});

$("#txtPassword").blur(function () {

if ($("#txtPassword").val() == "")

$("#PassValidator").show("slow");

else

$("#PassValidator").hide("slow");

});

$("#btnShowLogin").click(function () {

$("#tblLogin").slideToggle("slow");

});

});

</script>

## Step 15: Modifying the Home, MyProducts and Contact Pages

* Add jquery files (jquery-1.7.1.js and jquery-1.7.1.min.js) to Home, MyProducts and Contact pages
* Write a javascript code between the head.

<head>

<title></title>

<scripttype="text/javascript">

$(document).ready(function () {

$("#dvLogin").load("login.html");

});

</script>

</head>

* Paste the above code in Home.html, MyProducts.html and Contact.html head section
* Now, open the Home.html page and edit the <div id=dvTop> and write the following code:

<divid="dvTop">

<tablewidth="100%">

<tr>

<tdstyle="width:50%">

<h3>Welcome to My First Website</h3>

</td>

<td>

<divid="dvLogin"style="position:relative;">sd</div>

</td>

</tr>

</table>

</div>

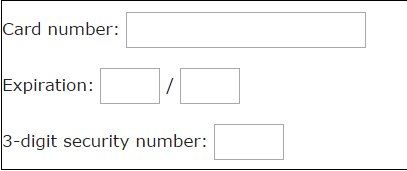
* The above code will show the Login button in the div Top

## JQuery Exercise:

Make a web page with an HTML form like this:



Using jQuery, create the following functionality: If Credit is selected, the following extra fields appear:



In addition, make sure only numbers can be entered in the input fields. If instead Wire transfer is selected, the following extra text appears:

Transfer the amount to: reg. 9570, account 0009286322 (in Pakistan)

Implement a drap-and-drop feature where items of a list of class "removable" (e.g. <ul class="removable">) can be removed by dragging-and-dropping individual items from the list to a trashcan image (like this one: http://www.mytodos.com/images/trashcan-open.gif). Whenever an item is moved to the trashcan, a dialog asks the user to confirm.

Implement a "click for more information" feature where items of a list of class "informationable" can be clicked, in which case more information (fragments of HTML code) about the relevant item is retrieved from the server via AJAX. (For the server, just use constant files containing the HTML fragments.)



WEB PROGRAMMING LANGUAGES

**LAB-6**

# ASP.NET Validation Controls

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** During this exercise, you will learn how to:

* Use validator controls to check user input in an ASP.NET Web page.
* Format the display of validation error messages.
* Create custom validation by writing code.

In the first part of the exercise, you will create a Web site and a page where you can work with styles. create a new Web site.

## To create a file system Web site

1. Open Visual Studio or Visual Web Developer Express.
2. In the **File** menu, click **New Web Site**.

The **New Web Site** dialog box is displayed.

1. Under **Installed Templates**, click **Visual Basic** or **Visual** **C#** and then select **ASP.NET Web Site**.
2. In the **Web** **location** box, select **File System**, and then enter the name of the folder where you want to keep the pages for your Web site.

For example, type the folder name **C:\WebSites**.

1. Click **OK**.

## Adding Controls

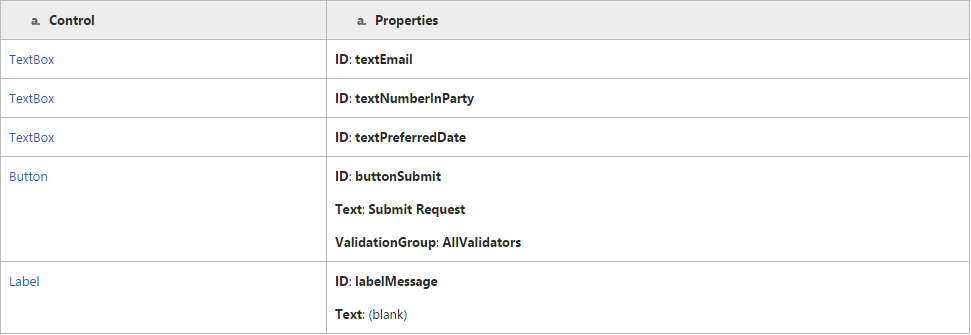
You will use only a few controls to prompt the user the information required for the reservation.

## To add controls and text

1. In **Solution Explorer**, right-click the project and select **Add New Item**.

The **Add New Item** dialog box is displayed.

1. Under **Installed Templates** select **Visual C#**, and then select **Web Form**.
2. Enter **ValidationExample.aspx** as the name for the new Web page and then click **Add**.
3. Switch to **Design** view.
4. Type a page heading such as **Submit a Reservation** and then press ENTER to create a new line below the heading.
5. From **Standard** group of the **Toolbox**, drag the following controls onto the page below the heading and set their properties as indicated. After each control is dropped on the page, press ENTER so that each control is on a separate line, and press ENTER twice before the [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) control to leave room for additional controls to be added later.



1. Type text in front of the text boxes as captions.
2. Double-click the **Submit Request** button to create a handler for its [Click](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.click.aspx) event and then add code as shown in the following example:

protectedvoid buttonSubmit\_Click(object sender, EventArgs e)

{

if (Page.IsValid)

{

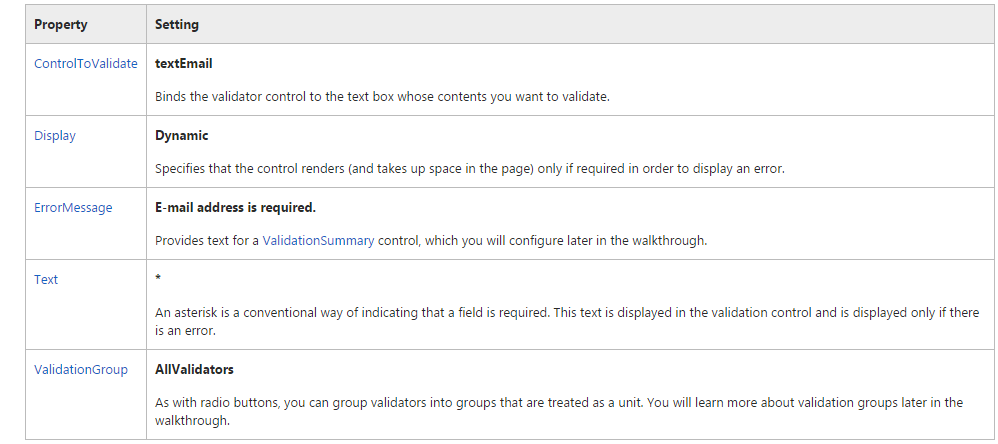
labelMessage.Text = "Your reservation has been processed.";

}

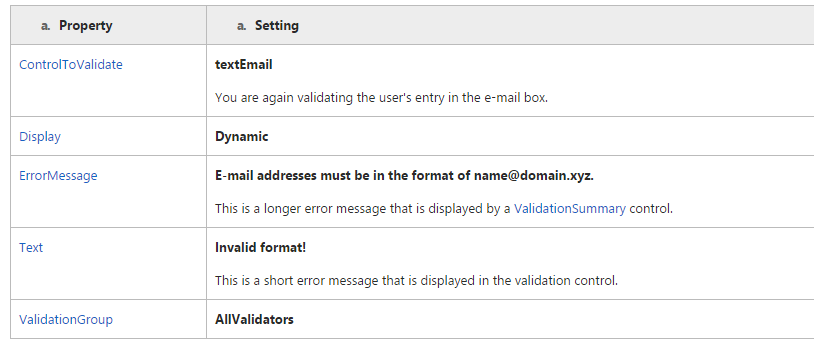
}

## To add basic validation

1. In the .aspx page, switch to **Design** view.
2. From the **Validation** group of the **Toolbox**, drag a [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) control and drop it next to the **textEmail** text box.
3. Set the following properties of the [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) control:



1. From the **Validation** group of the **Toolbox**, drag a [RegularExpressionValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.regularexpressionvalidator.aspx) control and drop it next to the [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) you just added.
2. Set the following properties of the [RegularExpressionValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.regularexpressionvalidator.aspx) control:



1. With the [RegularExpressionValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.regularexpressionvalidator.aspx) control still selected, in the Property window click the ellipsis button in the [ValidationExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.regularexpressionvalidator.validationexpression.aspx) box.

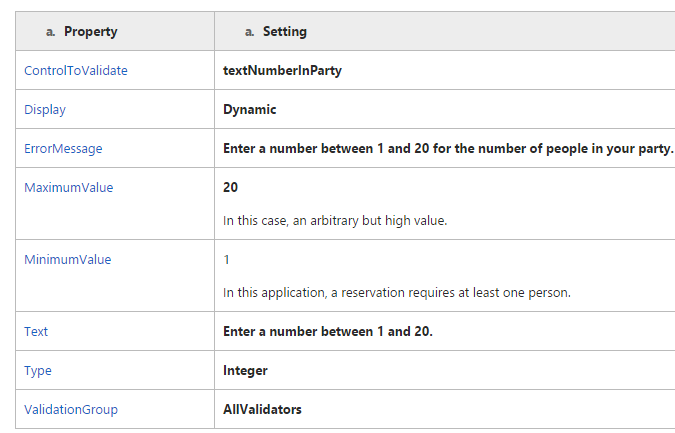
Regular expressions constitute a language that can be used to find precisely defined patterns in strings. In the [RegularExpressionValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.regularexpressionvalidator.aspx) control, you define a regular expression for the pattern that is valid in this case, the pattern for a valid e-mail address.

The **Regular Expression Editor** contains a list of commonly used regular expressions so that you can use the validator control without learning regular expression syntax.

1. In the **Standard Expressions** list, click **Internet E-mail Address**.

The regular expression for an e-mail address is put in the **Validation Expression** box.

1. Click **OK** to close the dialog box.
2. Add another [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) control, using the instructions earlier in this procedure. However, this time bind it to the **textNumberInParty** text box and set its[ErrorMessage](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.errormessage.aspx) property to **Please indicate how many people are in your party**.
3. From the **Validation** group of the **Toolbox**, drag a [RangeValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.rangevalidator.aspx) control and drop it next to the [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) you just added.
4. Set the following properties of the [RangeValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.rangevalidator.aspx) control:



## Testing the Page

You can now test the validator controls that you have so far.

## To test basic validation

1. Press CTRL+F5 to run the page.
2. When the page appears in the browser, click the **Submit Request** button.

Several validation errors are displayed, because you have not filled in some required fields. Note that the validation errors are displayed immediately — the page is not submitted. By default, the validator controls inject client-side ECMAScript (JavaScript) into the page to perform validation checking in the browser. This gives users instant feedback on validation errors; without the client script, checking for validation errors would require a round trip to the server, which could be slow at times. In fact, you cannot submit the page until the client-side validation succeeds. (The same validation check is performed again when the page is submitted as a security precaution.)

1. Check the validation on the **textEmail** text box by typing an invalid e-mail address, and then again with a valid e-mail address such as your own.
2. Check that you can enter only numeric values between 1 and 20 in the **Number of people in party** text box.
3. When you have entered valid values, click the **Submit Request** button. If the validation passes for all controls, you will see the message **Your reservation has been processed**.
4. After you have confirmed that validation is working as expected, close the browser.

If validation is not working as expected, double-check that you have made all the property settings listed above and then run the page again.

## To display validation information in alternative ways

1. From the **Validation** group of the **Toolbox**, drag a [ValidationSummary](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.aspx) control to the bottom of the page.
2. Set the [ValidationSummary](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.aspx) control's [ValidationGroup](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.validationgroup.aspx) property to **AllValidators**.
3. Run the page.
4. Perform the same testing you did earlier in the walkthrough.

For each error, you see error information in two places. A short error message (the validator's **Text** property value) is displayed where the validator control is. The longer error message (the control's **ErrorMessage** property value) is displayed in the [ValidationSummary](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.aspx) control.

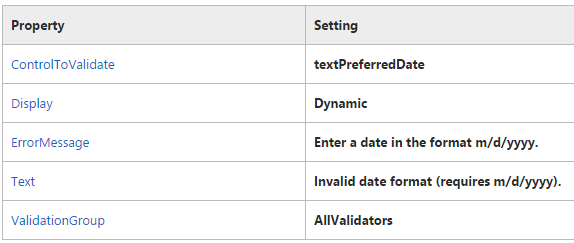
1. Close the browser.
2. Set the [ValidationSummary](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.aspx) control's [ShowMessageBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.showmessagebox.aspx) property to **true**.
3. Run the page.
4. Perform the same tests.

This time, each error results in a pop-up message in the browser.

1. Close the browser.

## To use a CustomValidator control to check for a valid date

1. From the **Validation** group of the **Toolbox**, drag a [CustomValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.aspx) control onto the page and position it next to the **textPreferredDate** text box.
2. Set the following properties of the [CustomValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.aspx) control:



1. Double-click the [CustomValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.aspx) control to create a handler for its [ServerValidate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.servervalidate.aspx) event and then add code as shown in the following example:

protectedvoid CustomValidator1\_ServerValidate(object source,

ServerValidateEventArgs args)

{

try

{

DateTime.ParseExact(args.Value, "m/d/yyyy",

System.Globalization.DateTimeFormatInfo.InvariantInfo);

args.IsValid = true;

}

catch

{

args.IsValid = false;

}

}

1. This code runs when the user submits the page. The code interacts with the validator control using the [ServerValidateEventArgs](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventargs.aspx) (args) object passed into the handler. The value that the user has entered into the **textPreferredDate** text box is passed as the args object's [Value](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventargs.value.aspx) property. After you have checked whether the user's entry is valid, you set the args object's [IsValid](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventargs.isvalid.aspx) property to **true** or **false**. If you set it to **false**, the validator will display its error message.
2. In this example, the code uses a try-catch block to determine whether the user's entry can be converted into a [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx) object. If the user enters an invalid value (anything that does not conform to a date in the format *m/d/yyyy*), the [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx) object's [ParseExact](http://msdn.microsoft.com/en-us/library/system.datetime.parseexact.aspx) method throws an exception, and the **Catch** block is executed.
3. In the Button\_Click handler, add code as shown in the following example:

protectedvoid buttonSubmit\_Click(object sender, EventArgs e)

{

if(Page.IsValid)

{

labelMessage.Text = "Your reservation has been processed.";

}

else

{

labelMessage.Text = "Page is not valid.";

}

}



WEB PROGRAMMING LANGUAGES

**LAB-7**

# Master Pages

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Creating a Website:

### **To create a file system Web site**

1. Open Visual Studio or Visual Web Developer Express.
2. In the **File** menu, click **New Web Site**.

The **New Web Site** dialog box is displayed.

1. Under **Installed Templates**, click **Visual Basic** or **Visual** **C#** and then select **ASP.NET Empty Web Site**.
2. In the **Web** **location** box, select **File System**, and then enter the name of the folder where you want to keep the pages for your Web site.

For example, type the folder name **C:\WebSites**.

1. Click **OK**.

Visual Studio creates a Web site project that includes a Web.config file.

## Creating the Master Page

The master page is the template for how your pages will look. In this section, you will first create a master page. You will then use a table to lay out the master page with a menu, a logo, and a footer that will appear on each page of your site. You will also work with a content placeholder, which is a region in the master page that can be replaced with information in a content page.

### **To create the master page**

1. In **Solution Explorer**, right-click the name of your Web site, and then click **Add** **New Item**.
2. Under **Visual Studio installed templates**, click **Master Page**.
3. In the **Name** box, type **Master1**.
4. Select the **Place code in separate file** check box.
5. In the **Language** list, click the programming language you prefer and then click **Add**.

The new master page opens in **Source** view.

At the top of the page is an [@ Master](http://msdn.microsoft.com/en-us/library/ms228176(v=vs.100).aspx) declaration instead of the [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a(v=vs.100).aspx) declaration normally found at the top of ASP.NET pages. The body of the page contains a[ContentPlaceHolder](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.contentplaceholder(v=vs.100).aspx) control, which is the area of the master page where replaceable content will be merged from content pages at run time. You will work more with the content placeholder later in this walkthrough.

## Laying out the Master Page:

The master page defines how the pages in your site look. It can contain any combination of static text and controls. A master page also contains one or more content placeholders that designate where dynamic content will appear when pages are displayed.

In this walkthrough, you will use a table to help you position elements on the page. You will start by creating a layout table to hold the master page elements. Later in this section you will position the content placeholder control that is already on the page.

### **To create a layout table for the master page**

1. With the Master1.master file selected in **Source** view, use the **Target Schema for Validation** drop-down list on the toolbar to set the target schema to **Microsoft Internet Explorer 6.0**.
2. Switch to **Design** view.
3. Click on the center of the page to select the page. From the **Properties** window, set **BgColor** property to a distinctive color, such as blue.

The color you select is not important. Later in this walkthrough you will create a second master page without a color to contrast with what you have selected here.

1. Click the page where you want to place the layout table.
2. On the **Table** menu, click **Insert Table**.
3. In the **Insert Table** dialog box, create a table with three rows and one column, and then click **OK**.
4. Place the cursor inside the second row of the table.
5. On the **Table** menu, in the **Modify** submenu, click **Split Cells**.
6. In the **Split Cells** dialog box, select **Split into columns**, and then click **OK**.
7. Make the following settings:
   1. In the middle row, click the leftmost column, and then set its **Width** to 48 in the **Properties** window.
   2. Click the top row, and then set its **Height** to 48 in the **Properties** window.
   3. Click the bottom row, and then set its **Height** to 48 in the **Properties** window.
8. Select all cells in the table and set **BgColor** to a different color than the background color.

After laying out the table, you can add content to the master page that will appear on all pages. You will add a copyright message as a footer, and then add a menu. If you have a logo graphic available, you can add that as well.

### **To add static content to the master page**

1. Click the bottom cell, and then type footer text such as **Copyright 2007 Contoso Inc.**
2. In the **Toolbox**, from the **Navigation** control group, drag a [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu(v=vs.100).aspx) control into the top cell.
3. Create a menu by following these steps:
   1. Set the [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu(v=vs.100).aspx) control's [Orientation](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.orientation(v=vs.100).aspx) property to [Horizontal](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.orientation(v=vs.100).aspx).
   2. Click the smart tag on the [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu(v=vs.100).aspx) control, and click **Edit Menu Items** in the **Menu Tasks** dialog box. The **Edit Menu Items**dialog box appears.
4. In the **Edit Menu Items**dialog box, in the **Items** section, click the **Add a root node** icon twice to add two menu items:
   1. Click the first node, and then set **Text** to **Home** and **NavigateUrl** to **Default.aspx**.
   2. Click the second node, and then set **Text** to **About** and **NavigateUrl** to **About.aspx**.
   3. Click **OK** to close the **Menu Item Editor** dialog box.
5. If you have a graphics file available to use as a logo, follow these steps to place it on the master page:
   1. In **Solution Explorer**, right-click the name of your Web site, and then click **Add Existing Item**.
   2. Navigate to your graphics file, select the graphic file, and then click **Add**.
   3. In the **Toolbox**, from the **Standard** group, drag an [Image](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.image(v=vs.100).aspx) control to the middle left column of the table.
   4. Set the [Image](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.image(v=vs.100).aspx) control's [ImageUrl](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.image.imageurl(v=vs.100).aspx) property to the name of the graphics file.

You can now position the content placeholder to specify where the master page can display content at run time.

### **To move the content placeholder**

1. Drag the [ContentPlaceHolder](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.contentplaceholder(v=vs.100).aspx) control into the middle right cell. You can move the control by clicking inside the control and then dragging it by its tag.

The control's **ID** property is ContentPlaceholder1. You can leave this name or change it. If you change the name, make a note of the name because you will need to know the name later.

1. Save the page.

## Creating Content for the Master Page:

The master page provides the template for your content. You define content for the master page by creating an ASP.NET page that is associated with the master page. The content page is a specialized form of an ASP.NET page that contains only the content to be merged with the master page. In the content page, you add the text and controls that you want to display when users request that page.

In this walkthrough, you will add two pages with content for the master page. The first is a default (home) page and the second is an about page.

### **To create the Default page**

1. In **Solution Explorer**, right-click the name of your Web site, and click **Add New Item**.
2. Under **Visual Studio installed templates**, click **Web Form**.
3. In the **Name** box, keep the file name that Visual Studio inserts: **Default**.
4. In the **Language** list, click the programming language you prefer.
5. Select the **Select master page** check box, and then click **Add**.

The **Select a Master Page** dialog box appears.

1. Click **Master1.master**, and then click **OK**.

A new .aspx file is created. The page contains an [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a(v=vs.100).aspx) directive that attaches the current page to the selected master page with the **MasterPageFile** attribute, as shown in the following code example.

<%@ Page Language="C#" MasterPageFile="~/Master1.master" ... %>

The page also contains an [Content](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.content(v=vs.100).aspx) control element that you will work with next.

A content page does not have the usual elements that make up an ASP.NET page, such as **html**, **body**, or **form** elements. Instead, you add only the content that you want to display on the master page by replacing the placeholder regions you created in the master page.

### **To add content to the Default page**

1. In **Source** view, type **Contoso Home Page** in the **Title** element of the **@ Page** directive at the top of the page.

You can set the title of each content page independently, so that the correct title is displayed in the browser when the content is merged with the master page. The title information is stored in the content page's **@ Page** directive.

1. Switch to **Design** view.

The [ContentPlaceHolder](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.contentplaceholder(v=vs.100).aspx) controls in the master page are displayed as [Content](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.content(v=vs.100).aspx) controls in the new content page. The rest of the master page content is displayed so that you can see the layout. However, it appears dimmed because you cannot change it while you are editing a content page, and the cursor becomes the I-bar only where you can add content.

1. In the [Content](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.content(v=vs.100).aspx) control that matches **ContentPlaceHolder1** on the master page, type **Welcome to the Contoso Web Site**.
2. Select the text then format it as a heading by selecting **Heading 1** from the **Block Format** drop-down list above the **Toolbox**.
3. Press ENTER to create a new blank line in the [Content](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.content(v=vs.100).aspx) control, and then type **Thank you for visiting our site.**

The text you add here is not important; you can type any text that will help you recognize this page as the Default (home) page.

1. Save the page.

You can create the about page the same way you created the Default page.

### **To create the about page**

1. Use the same steps that you used for the Default page to add a new content page named About.aspx.

Be sure to attach the new page to the Master1.master page as you did with the Default page.

1. Change the page's title to **Contoso About Page**.
2. In the content region, type **About Contoso**, and then format the text as a **Heading 2** by selecting the text and selecting **Heading 2** from the **Block Format** drop-down list above the **Toolbox**.
3. Press ENTER to create a new line, and then type **Since 1982, Contoso has provided high-quality software services.**
4. Save the page.

## Testing the Pages:

You can test the pages by running them as you would any ASP.NET page.

### **To test the pages**

1. Switch to the Default.aspx page, and then press CTRL+F5.

ASP.NET merges the content in the Default.aspx page with the layout in the Master1.master page into a single page and displays the resulting page in the browser.

1. Click the **About** link.

The About.aspx page is displayed. It is also merged with Master1.master page.

## Referencing the Content of Master Pages:

Code in the content pages can reference members on the master page, including any public properties or methods and any controls on the master page. In this part of the walkthrough, you will create a property on the master page, and then use the value of the property in the content pages. The premise is that the company name for the Web site is stored as a property in the master page, and any reference to the company name in the content pages is based on the master page property.

The first step is to add a property to the master page.

### **To add a property to the master page**

1. Switch to or open the Master1.master page.
2. In **Solution Explorer**, right-click Master1.master, and then click **View Code** to open the code editor.
3. Inside the class definition, type the following code.

public String CompanyName

{

get { return (String) ViewState["companyName"]; }

set { ViewState["companyName"] = value; }

}

1. Inside the class definition (but not inside the property code), add the following code.

void Page\_Init(Object sender, EventArgs e)

{

this.CompanyName = "Contoso";

}

You can now modify the content page to use the master page's CompanyName property.

### **To reference the CompanyName property in the content page**

1. Switch to or open the Default.aspx page.
2. Switch to **Source** view.
3. At the top of the page, underneath the [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a(v=vs.100).aspx) directive, add the following [@ MasterType](http://msdn.microsoft.com/en-us/library/ms228274(v=vs.100).aspx) directive:

<%@ MasterType virtualpath="~/Master1.master" %>

1. The directive binds the content page's **Master** property, which you will use shortly, to the Master1.master page.
2. Switch to **Design** view.
3. In the [Content](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.content(v=vs.100).aspx) control, change the text to **Welcome to the Web site of** .
4. In the **Toolbox**, from the **Standard** group, drag a [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label(v=vs.100).aspx) control onto the [Content](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.content(v=vs.100).aspx) control, and place it after the static text so that the text reads:

Welcome to the Web site of [Label]

1. Set the [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label(v=vs.100).aspx) control's [ID](http://msdn.microsoft.com/en-us/library/system.web.ui.control.id(v=vs.100).aspx) property to **CompanyName**.
2. In **Solution Explorer**, right-click **Default.aspx**, and then click **View Code** to open the code editor.
3. Inside the class definition, add the following code.

void Page\_Load(Object sender, EventArgs e)

{

CompanyName.Text = Master.CompanyName;

}

The content page's [Master](http://msdn.microsoft.com/en-us/library/system.web.ui.page.master(v=vs.100).aspx) property returns a reference to the master page as defined in the [@ MasterType](http://msdn.microsoft.com/en-us/library/ms228274(v=vs.100).aspx) directive you added in step 3.

You can now test the content page to be sure it is referencing the master page's CompanyName property correctly.

### **To test the reference to the master page property**

1. Switch to or open the Default.aspx page, and then press CTRL+F5 to run the page.

The page is displayed in the browser, with the text **Welcome to the Web site of Contoso**

1. Close the browser.
2. Switch to or open the Master1.master code-behind page.
3. Change the **Page\_Init** handler to assign a different company name to the property, as in the following code example.

void Page\_Init(Object sender, EventArgs e)

{

this.CompanyName = "New Company Name";

}

1. Switch to the Default.aspx page, and then press CTRL+F5 to run it again.

This time, the updated company name appears in the page.



WEB PROGRAMMING LANGUAGES

**LAB-8**

# ASP.NET Themes

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Tasks illustrated in this walkthrough include:

* Applying predefined ASP.NET themes to individual pages and to your site as a whole.
* Creating your own theme that includes skins, which are used to define the look of individual controls.

## Creating the Website:

### **To create a file system Web site**

1. Open Visual Web Developer.
2. On the **File** menu, click **New** **Web Site**.

The **New Web Site** dialog box appears.

1. Under **Visual Studio installed templates**, click **ASP.NET Web Site**.
2. In the **Location** box, enter the name of the folder where you want to keep the pages of your Web site.

For example, type the folder name **C:\WebSites**.

1. In the **Language** list, click the programming language you prefer to work in.
2. Click **OK**.

Visual Web Developer creates the folder and a new page named Default.aspx.

### **To place controls on the page**

1. Switch to Design view.
2. From the **Standard** group of the **Toolbox**, drag a **Calendar** control, a **Button** control, and a **Label** control to the page. The exact layout of the page is not important.
3. Switch to **Source** view.
4. Be sure that the **head** element of the page has the **runat="server"** attribute so that it reads as follows:

<head runat="server"></head>

1. Save the page.

To test the page, you will want to see the page before a theme is applied, and then with different themes.

## Creating and applying a theme to Page:

ASP.NET makes it easy to apply a predefined theme to a page, or to create a unique theme. In this part of the walkthrough, you will create a theme with some simple skins, which define the appearance of controls.

### **To create a new theme**

1. In Visual Web Developer, right-click the name of your Web site, click **Add ASP.Net Folder**, and then click **Theme**.

The folder named **App\_Themes** and a subfolder named **Theme1** are created.

1. Rename the **Theme1** folder **sampleTheme**.

The name of this folder will be the name of the theme that you create (here, **sampleTheme**). The exact name is not important, but you must remember it when you apply your custom theme.

1. Right click the **sampleTheme** folder, select **Add New Item**, add a new text file, and name it **sampleTheme.skin**.
2. In the sampleTheme.skin file, add skin definitions as shown in the following code example.

<asp:Label runat="server" ForeColor="red" Font-Size="14pt" Font-Names="Verdana" />

<asp:button runat="server" Borderstyle="Solid" Borderwidth="2px" Bordercolor="Blue" Backcolor="yellow"/>

1. Save the skin file, and then close it.

### **To test themes**

1. Press CTRL+F5 to run the page.

The controls are displayed with their default appearance.

1. Close the browser, and then return to Visual Web Developer.
2. Open or switch to Default.aspx, and then switch to **Source** view.
3. In the **@ Page** directive add a **Theme** attribute that specifies **sampleTheme** as the theme name:

<%@ Page Theme="sampleTheme" ... %>

1. Press CTRL+F5 to run the page again.

This time, the controls are rendered with the color scheme defined in your theme.

The **Label** and **Button** controls will appear with the settings you made in the sampleTheme.skin file. Because you did not make an entry in the sampleTheme.skin file for the **Calendar** control, it is displayed with its default appearance.

1. In Visual Web Developer, set the theme to the name of another theme, if available.
2. Press CTRL+F5 run the page again.

The controls change appearance again.

## Style Sheet themes vs. Customization theme

### **To use a style sheet theme and see order of precedence**

1. Switch to **Source** view.
2. Change the page declaration:

<%@ Page theme="sampleTheme" %>

<%@ Page StyleSheetTheme="sampleTheme" %>

1. Press CTRL+F5 to run the page.

Note that the **ForeColor** property of the **Label1** control is red.

1. Switch to Design view.
2. Select **Label1** and, in **Properties**, set **ForeColor** to **blue**.
3. Press CTRL+F5 to run the page.

The **ForeColor** property of **Label1** is blue.

1. Switch to **Source** view.
2. Change the page declaration to declare a theme, rather than a style sheet theme, by changing:

<%@ Page StyleSheetTheme="sampleTheme" %>

<%@ Page Theme="sampleTheme" %>

1. Press CTRL+F5 to run the page.

The **ForeColor** property of **Label1** is again red.

## Basing a Custom Theme on Existing Controls

An easy way to create skin definitions is to use the designer to set appearance properties, and then copy the control definition to a skin file.

### **To base a custom theme on existing controls**

1. In Design view, set properties of the **Calendar** control so that the control has a distinctive look. The following settings are suggestions:
   * **BackColor**   Cyan
   * **BorderColor**   Red
   * **BorderWidth**   4
   * **CellSpacing**   8
   * **Font-Name**   Arial
   * **Font-Size**   Large
   * **SelectedDayStyle-BackColor**   Red
   * **SelectedDayStyle-ForeColor**   Yellow
   * **TodayDayStyle-BackColor**   Pink
2. Switch to **Source** view and copy the **<asp:calendar>** element and its attributes.
3. Switch to or open the sampleTheme.skin file.
4. Paste the **Calendar** control definition into the sampleTheme.skin file.
5. Remove the **ID** property from the definition in the sampleTheme.skin file.
6. Save the sampleTheme.skin file.
7. Switch to the Default.aspx page, and drag a second **Calendar** control onto the page. Do not set any of its properties.
8. Run the Default.aspx page.

Both **Calendar** controls will appear the same. The first **Calendar** control reflects the explicit property settings that you made. The second **Calendar** control inherited its appearance properties from the skin definition that you made in the sampleTheme.skin file.

## Applying Themes to a Website

You can apply a theme to an entire Web site, which means you do not need to apply the theme to individual pages. (If you want, you can override the themes settings on a page.)

### **To set a theme for a Web site**

1. Open the Web.config file.
2. In the **pages** element, add a **theme** attribute and set its value to the name of theme that you want to apply to the entire Web site, as in the following example:
3. Save and close the Web.config file.
4. Switch to or open the Default.aspx file, and then switch to **Source** view.
5. Remove the **theme** attribute (theme="themeName") from the **@ Page** declaration.
6. Press CTRL+F5 to run Default.aspx.

The page is now displayed with the theme you specified in the Web.config file.

If you choose to specify a theme name in your page declaration, it will override any theme specified in the Web.config file.

* 

WEB PROGRAMMING LANGUAGES

Lab-9

# ASP.NET Custom Controls

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Objective: This exercise shows you how to create and test a custom ASP.NET Web server control.

In this exercise you will learn how to:

* Create an ASP.NET Web server control.
* Specify security settings and design-time behavior by adding metadata to the control.
* Specify a tag prefix in a configuration file and in the control's assembly.
* Specify an icon to use for the control in the Visual Studio toolbox.
* Compile the control into an assembly and add a project reference so that it can be used in another project in the same solution.
* Test the control in a Web page and access its properties by using the **Properties** window or IntelliSense.

## Creating the Server Control:

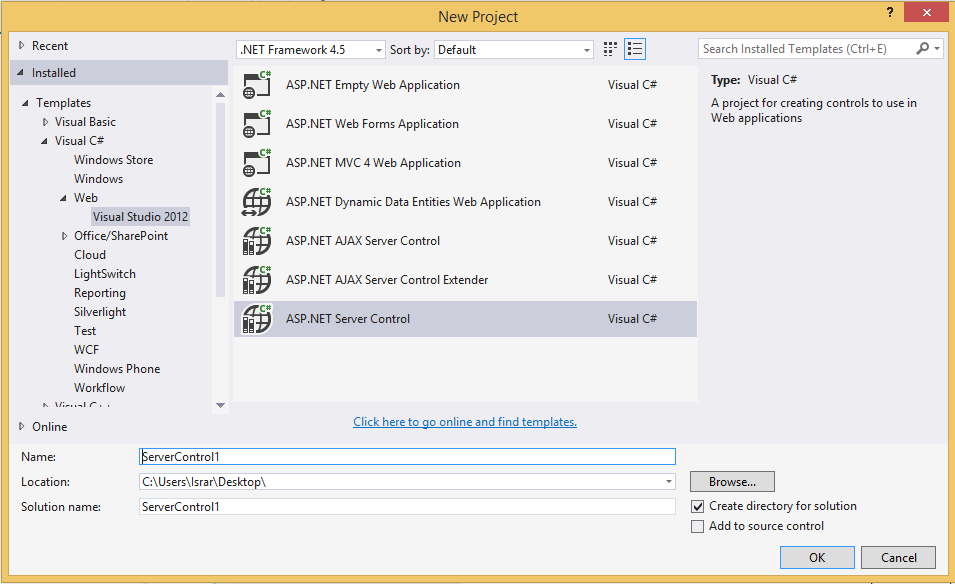
You will create a simple control that derives from the standard [Label](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label(v=vs.100).aspx) control. The control is named WelcomeLabel. It appends the user's name to the text string that is defined in the Text property. For example, if the page developer sets "Hello" as the value of the Text property, the WelcomeLabel control renders "Hello, *userName*!"

The control defines a DefaultUserName property that specifies the user name value to use when the user is not logged in. For example, if the page developer sets the Text property to "Hello" and the DefaultUserName property to "Guest", the control displays "Hello Guest!" if the user is not logged in.

### **To create the custom server control**

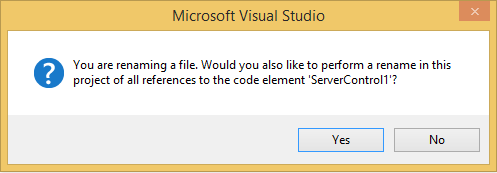
1. From the **File** menu select **New Project**.

The **New Project** dialog box is displayed.

1. Under **Installed Templates**, expand  **Visual C#**, and then select **Web**.
2. Select the **ASP.NET Server Control** template.
3. In the **Name** box enter **ServerControl1**.The **New Project** dialog box resembles the following illustration.
4. Click **OK**.

Visual Studio creates a server control project that has a class file that is named ServerControl1.cs.

1. Rename ServerControl1.cs to WelcomeLabel.cs.

A dialog box asks if you want to rename all references to ServerControl1, as shown in the following illustration.

1. Click **Yes**.
2. Open WelcomeLabel.cs.
3. Change the WelcomeLabel class so that it inherits from [System.Web.UI.WebControls.Label](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label(v=vs.100).aspx) instead of from [System.Web.UI.WebControls.WebControl](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webcontrol(v=vs.100).aspx).
4. In the **ToolboxData** attribute for the WelcomeLabel class, change the string "ServerControl1" to "WelcomeLabel" in both places where it occurs.
5. Delete the code that is in the WelcomeLabel class, and insert a DefaultUserName property and a RenderContents method as shown in the following example:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace ServerControl1

{

[DefaultProperty("Text")]

[ToolboxData("<{0}:WelcomeLabel runat=server></{0}:WelcomeLabel>")]

publicclassWelcomeLabel : Label

{

[Bindable(true)]

[Category("Appearance")]

[DefaultValue("")]

[Localizable(true)]

publicvirtualstring DefaultUserName

{

get

{

string s = (string)ViewState["DefaultUserName"];

return (s == null) ? String.Empty : s;

}

set

{

ViewState["DefaultUserName"] = value;

}

}

protectedoverridevoid RenderContents(HtmlTextWriter writer)

{

writer.WriteEncodedText(Text);

string displayUserName = DefaultUserName;

if (Context != null)

{

string userName = Context.User.Identity.Name;

if (!String.IsNullOrEmpty(userName))

{

displayUserName = userName;

}

}

if (!String.IsNullOrEmpty(displayUserName))

{

writer.Write(", ");

writer.WriteEncodedText(displayUserName);

}

writer.Write("!");

}

}

}

1. In **Solution Explorer**, expand **Properties** and open AssemblyInfo.cs. (You might have to click **Show All Files** to be able to expand **My Project**.)
2. At the beginning of the file, add the following line:

using System.Web.UI;

1. This namespace is required for the [TagPrefixAttribute](https://msdn.microsoft.com/en-us/library/system.web.ui.tagprefixattribute(v=vs.100).aspx) attribute that you will add in the following step.
2. At the end of the file, add the following line:

[assembly: TagPrefix("ServerControl1", "aspSample")]

1. This [TagPrefixAttribute](https://msdn.microsoft.com/en-us/library/system.web.ui.tagprefixattribute(v=vs.100).aspx) attribute creates a mapping between the namespace ServerControl1 and the prefix aspSample.
2. Save the WelcomeLabel.cs file.

## Testing the Custom Server Control:

In the following procedure you will complete the following tasks:

* Create a Web site that you will use to test the server control that you created in the preceding procedure.
* Add a reference in the Web site project to the Server Control project.
* Add the WelcomeLabel control to the **Toolbox**.
* Add an instance of the WelcomeLabel control to the Default.aspx page.
* Run the Default.aspx page to see that the WelcomeLabel control works.

You begin by creating a Web site that you can use for testing.

This walkthrough uses a Web site project. You could use a Web application project instead.

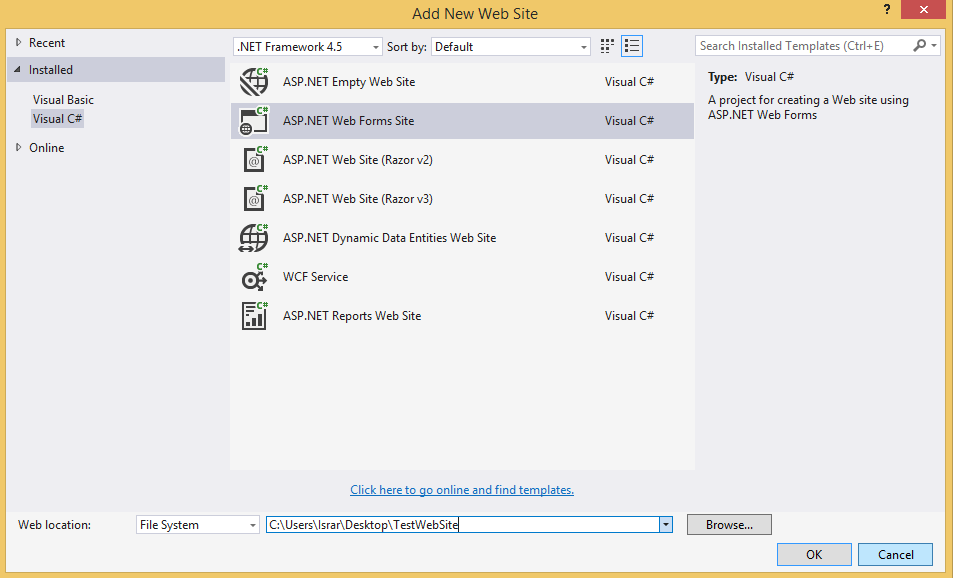
### **To create a Web site project to test the custom server control**

1. In the **File** menu, click **Add**, and then select **New Web Site**.

The **New Web Site** dialog box is displayed.

1. Under **Installed Templates**, select **Visual C#**, and then select the **ASP.NET Web Site** template.
2. Name the Web site **TestWebSite** and save it in a new folder that is under the ServerControl1 solution folder.

The **New Web Site** dialog box resembles the following illustration:

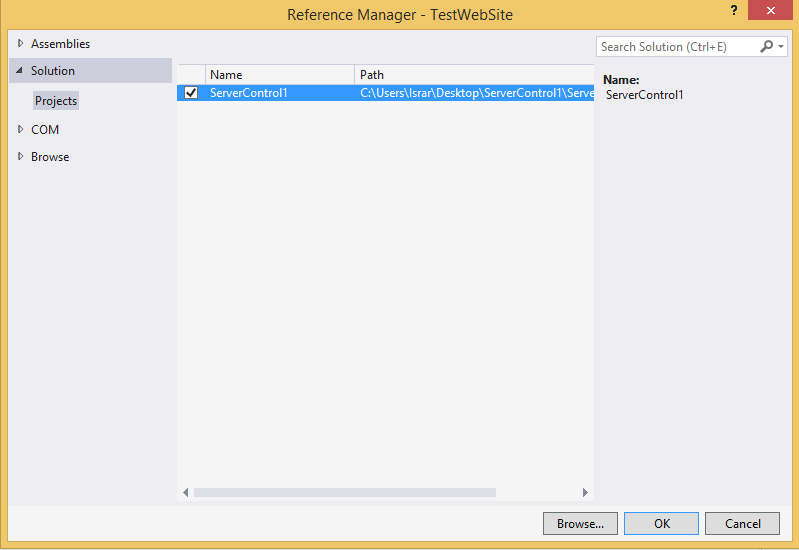


1. Click **OK**.

Visual Studio creates a Web site project, adds it to the ServerControl1 solution, and opens the Default.aspx page in **Source** view.

1. In **Solution Explorer**, right-click the Web site project and select **Set as Startup Project**.
2. In **Solution Explorer**, right-click the Web site project and select **Add Reference**.

The **Add Reference** dialog box is displayed.

1. Select the **Projects** tab, select the ServerControl1 project, and then click **OK**. The **Add Reference** dialog box is shown in the following illustration:

The next step is to add the server control to the toolbox so that you can use it in a Web page in the test Web site.

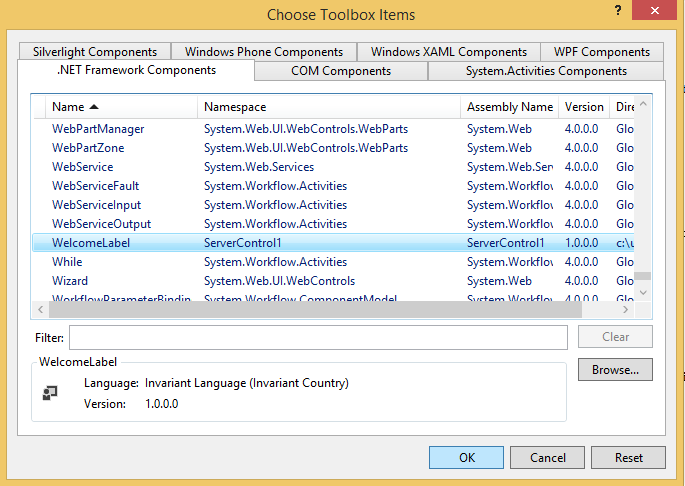
## To add the WelcomeLabel control to the toolbox

1. From the **Build** menu, select **Build Solution**.

Visual Studio compiles the solution and copies the assembly that is created by the ServerControl1 project into the Bin folder of the TestWebSite project.

1. With the Default.aspx page still open, open the **Toolbox** window.
2. Right-click anywhere in the **Toolbox** window and then click **Add Tab**.
3. Name the new tab **Server Control1**.
4. Right-click the **Server Control1** tab and then click **Choose Items**.
5. Select **Browse** and then browse to the Bin folder of the TestWebSite project.
6. Select ServerControl1.dll and then click **Open**.

The ServerControl1 assembly is added to the **.NET Framework Components** tab of the **Choose Toolbox Items** dialog box, as shown in the following illustration:



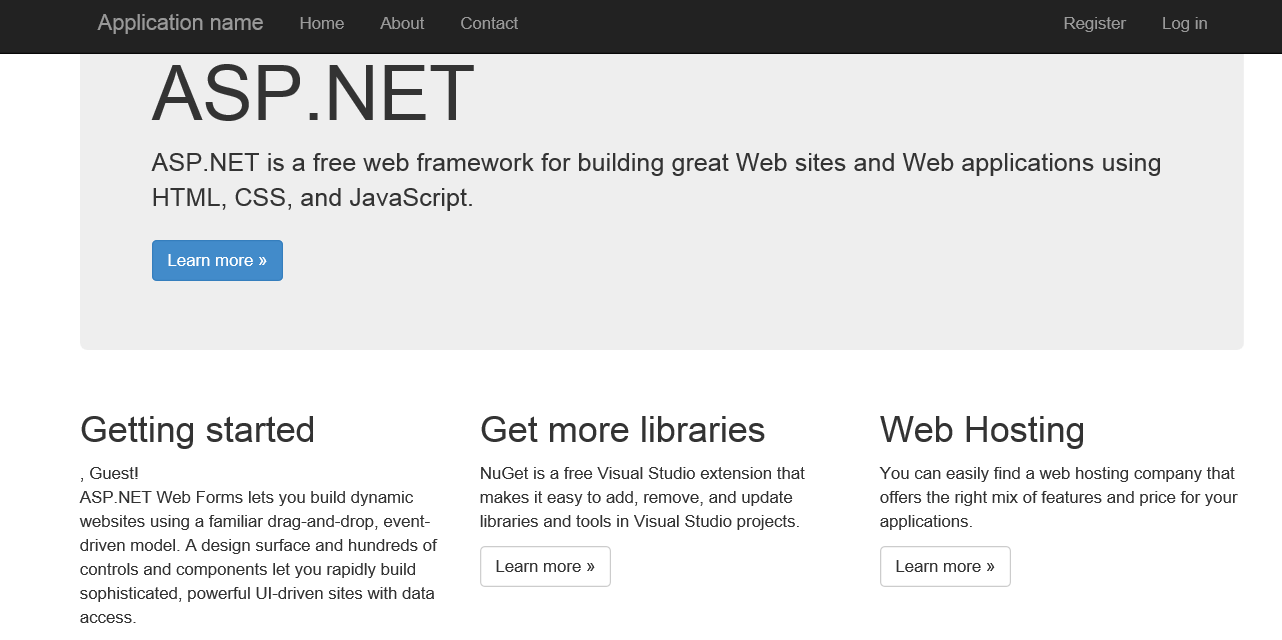
1. Click **OK**.

The WelcomeLabel control appears in the **Server Control1** tab in the **Toolbox**.

You can now add the server control to a Web page and test the Web page.

### **To test the WelcomeLabel control**

1. In the Default.aspx page, delete the text "Welcome to ASP.NET!" and in its place drag the WelcomeLabel control from the **Toolbox**.
2. Add **Text** and **DefaultUserName** attribute to the markup for the WelcomeLabel control and set it to "Welcome," as shown in the following example:
3. Press CTRL-F5 to display the Web page in the browser.
4. The WelcomeLabel control displays "Welcome!", as shown in the following illustration:





WEB PROGRAMMING LANGUAGES

**LAB-10**

# Language Integrated Query

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Creating a C# LINQ ASP.NET Web Site

 To create a new ASP.NET Web Site that can use LINQ/DLINQ/XLINQ and the new C# 3.0 language features, choose File->New Web Site in VS and select the “LINQ ASP.NET Web Site Template”:

## Creating your first ASP.NET page using LINQ

 Create a new page called Step1.aspx.  Within the .aspx page add a GridView control like so:

<%@ Page Language="C#" CodeFile="Step1.aspx.cs" Inherits="Step1" %>

<html>

<body>

    <form id="form1" runat="server">

    <div>

        <h1>City Names</h1>

        <asp:GridView ID="GridView1" runat="server">

        </asp:GridView>

    </div>

    </form>

</body>

</html>

Within the code-behind file we’ll then write the canonical “hello world” LINQ sample – which involves searching and ordering a list of strings:

using System;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Query;

public partial class Step1 : System.Web.UI.Page

{

    protected void Page\_Load(object sender, EventArgs e)

    {

        string[] cities = { "London", "Amsterdam", "San Francisco", "Las Vegas",

                            "Boston", "Raleigh", "Chicago", "Charlestown",

                            "Helsinki", "Nice", "Dublin" };

        GridView1.DataSource = from city in cities

                               where city.Length > 4

                               orderby city

                               select city.ToUpper();

        GridView1.DataBind();

    }

}

LINQ queries return results of type: IEnumerable<T> -- where <T> is determined by the object type of the “select” clause.  In the above sample “city” is a string, so the type-safe result is a generics based collection like so:

        IEnumerable<string> result = from city in cities

                                     where city.Length > 4

                                     orderby city

                                     select city.ToUpper();

Because ASP.NET controls already support databinding to any IEnumerable collection, we can easily assign this LINQ query result to the GridView and call its DataBind() method to generate this page output result:



## Using Richer Collections

Searching an array of strings is not terribly interesting (although sometimes actually useful).  More interesting would be the ability to search and work against richer collections of our own making.  The good news is that LINQ makes this easy.  For example, to better track trips we can create a simple class called “Location” in the project below:

using System;

public class Location

{

    // Fields

    private string \_country;

    private int    \_distance;

    private string \_city;

    // Properties

    public string Country

    {

        get { return \_country; }

        set { \_country = value; }

    }

    public int Distance

    {

        get { return \_distance; }

        set { \_distance = value; }

    }

    public string City

    {

        get { return \_city; }

        set { \_city = value; }

    }

}

This exposes 3 public properties to track the County, City name and Distance from Seattle.  we can then create a Step2.aspx file with a GridView control that defines 3 columns like so:

<%@ Page Language="C#" CodeFile="Step2.aspx.cs" Inherits="Step2" %>

<html>

<body>

    <form id="form1" runat="server">

    <h1>Cities and their Distances</h1>

    <asp:GridView ID="GridView1" AutoGenerateColumns="false" runat="server">

       <Columns>

          <asp:BoundField HeaderText="Country" DataField="Country" />

          <asp:BoundField HeaderText="City" DataField="City" />

          <asp:BoundField HeaderText="Distance from Seattle" DataField="Distance" />

       </Columns>

    </asp:GridView>

    </form>

</body>

</html>

we can then populate a collection of Location objects and databind it to the Grid in my code-behind like so:

using System;

using System.Collections.Generic;

using System.Web;

using System.Query;

public partial class Step2 : System.Web.UI.Page

{

    protected void Page\_Load(object sender, EventArgs e)

    {

        List<Location> cities = new List<Location>{

                                    new Location { City="London", Distance=4789, Country="UK" },

                                    new Location { City="Amsterdam", Distance=4869, Country="Netherlands" },

                                    new Location { City="San Francisco", Distance=684, Country="USA" },

                                    new Location { City="Las Vegas", Distance=872, Country="USA" },

                                    new Location { City="Boston", Distance=2488, Country="USA" },

                                    new Location { City="Raleigh", Distance=2363, Country="USA" },

                                    new Location { City="Chicago", Distance=1733, Country="USA" },

                                    new Location { City="Charleston", Distance=2421, Country="USA" },

                                    new Location { City="Helsinki", Distance=4771, Country="Finland" },

                                    new Location { City="Nice", Distance=5428, Country="France" },

                                    new Location { City="Dublin", Distance=4527, Country="Ireland" }

                                };

        GridView1.DataSource = from location in cities

                               where location.Distance > 1000

                               orderby location.Country, location.City

                               select location;

        GridView1.DataBind();

    }

}

The above code-behind shows off a few cool features.  The first is the new C# support for creating class instances, and then using a terser syntax for setting properties on them:

new Location { City="London", Distance=4789, Country="UK" }

This is very useful when instantiating and adding classes within a collection like above (or within an anonymous type like we’ll see later).  Note that rather than use an array this time,we are using a Generics based List collection of type “Location”.  LINQ supports executing queries against*any IEnumerable<T> collection*, so can be used against any Generics or non-Generics based object collections you already have.

For our LINQ query we are then returning a collection of all cities that are more than 1000 miles away from Seattle.  We have chosen to order the result in alphabetical order – first by country and then by city name.  The result of this LINQ query is again dictated by the type of the “location” variable – so in this case of type “Location”:

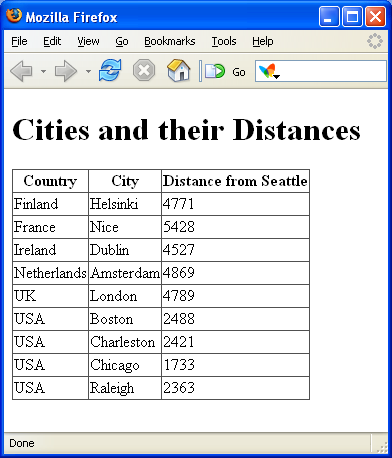
        IEumerable<Location> result = from location in cities

                                      where location.Distance > 1000

                                      orderby location.Country, location.City

                                      select location;

When we databind this result against the GridView I get a result like so:





WEB PROGRAMMING LANGUAGES

**LAB-11**

# Entity Framework

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Introduction

This article introduces Entity Framework to absolute beginners. The article is meant for developers who are primarily using ADO.NET to write their data access layers. Many experienced developers will find this article very basic but since the article is written from the perspective of beginners, I've tried to keep things simple.

## Background

ADO.NET is a very strong framework for data access. ADO.NET has been around since many years and there are a lot of systems running over ADO.NET. Developers who are totally oblivious to the concept of ORMs will probably be asking "What is Entity Framework? What are the benefits of using it and is it an alternative to ADO.NET?"

Well, to answer the first question about what is Entity Framework, Entity Framework is an Object Relational Mapper (ORM). It basically generates business objects and entities according to the database tables and provides the mechanism for:

1. Performing basic CRUD (Create, Read, Update, Delete) operations.
2. Easily managing "1 to 1", "1 to many", and "many to many" relationships.
3. Ability to have inheritance relationships between entities.

and to answer the second question, the benefits are:

1. We can have all data access logic written in higher level languages.
2. The conceptual model can be represented in a better way by using relationships among entities.
3. The underlying data store can be replaced without much overhead since all data access logic is present at a higher level.

and finally, the last question that whether it is an alternative to ADO.NET, the answer would be "yes and no". Yes because the developer will not be writing ADO.NET methods and classes for performing data operations and no because this model is actually written on top of ADO.NET, meaning under this framework, we are still using ADO.NET. So let us look at the architecture of Entity Framework (diagram taken from MSDN):

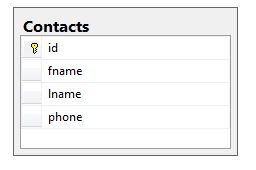


## Using the code

Let's try to understand the ease of use that Entity Framework provides by performing simple CRUD operations. Once we look at the code and how effortlessly and efficiently we can do these operations, the benefits of Entity Framework will become quite obvious.

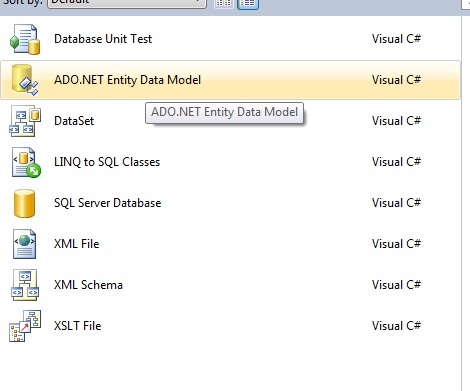
### Creating the database

Let's have a simple database with one table. Let's create a simple table for Contacts and we will perform CRUD operations on this table.

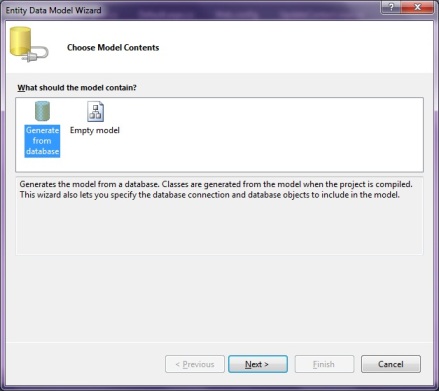


### Adding the Entity Model to the Website

Once we have the database ready, we can add the entity model to our website. We can do this by adding anADO.NET Entity Data Model to the website.



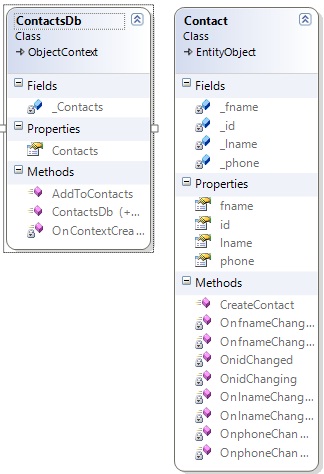
Once we select to add this data model to our website, we will have to select the approach we want to take for our Model's contents.



What this selection means is that we can either choose to generate the entity model from an existing database schema or we can design the entity model here and then later hook it up to the database. Since we already have the database ready, we will use the first option. Once the Model is generated, the Entity for each table is generated. The generated entity for our contact table is:

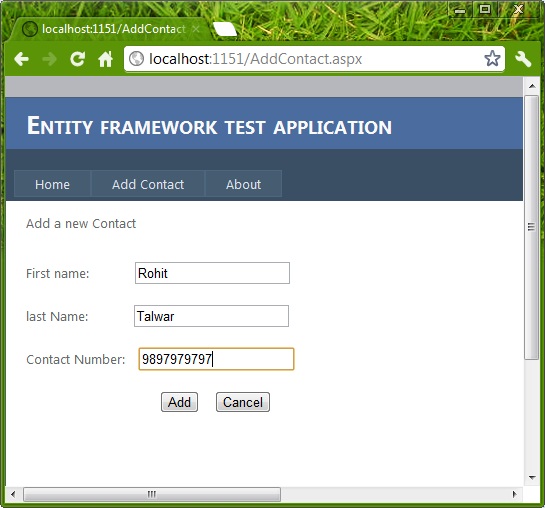


Also, the classes for performing database operations are also created. We just need to know how to use these classes to perform database operations.



### Insert operation

Let us create a simple page to perform an insert operation.



Now once the user chooses to insert the values into the database, the actual data operation can be performed by using the AddObject method of the Model class entity collection. The following code snippet show how to perform the insert.

Contact con = new Contact();

con.fname = TextBox1.Text;

con.lname = TextBox2.Text;

con.phone = TextBox3.Text;

ContactsDb db = new ContactsDb();

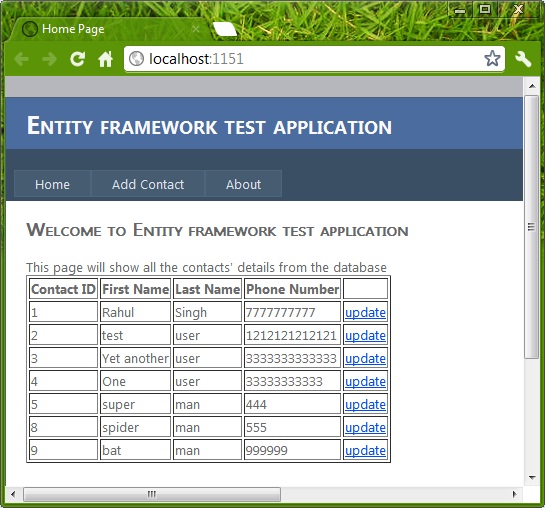
db.Contacts.AddObject(con);

db.SaveChanges();

This will insert the record into the table. You can notice the simplicity and efficiency of the code we wrote to perform the insertion.

### Reading all the records

There are scenarios when we want to read all records. Let's say we are making a page that will display all the contact information in a single page.



We can retrieve the collection of Entities using the Model object to achieve this. The code snippet below will show how that can be done.

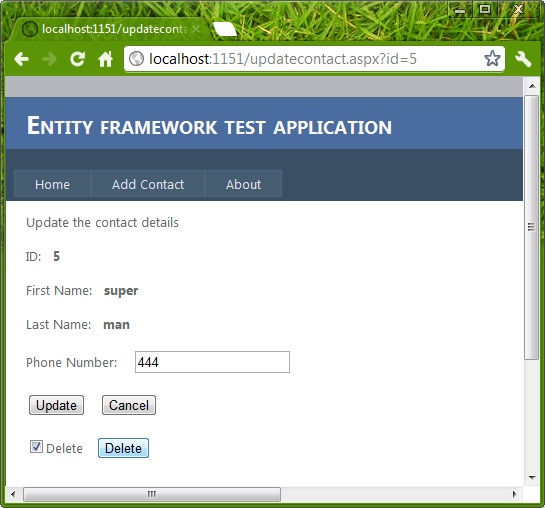
ContactsDb db = new ContactsDb();

Repeater1.DataSource = db.Contacts;

Repeater1.DataBind();

### Selecting a specific record

If we want to select a specific record from the table, we can use the SingleOrDefault method on the Model's entities collection. Let's say we want the functionality of updating/deleting a record on a single page then we will first have to select the record based on the ID, then update/delete the selected record.



Selection of any particular record (Contact) based on ID can be done as:

int idToupdate = Convert.ToInt32(Request.QueryString["id"].ToString());

ContactsDb db = new ContactsDb();

Contact con = db.Contacts.SingleOrDefault(p => p.id == idToupdate);

Once this code is executed, the Contact object will contain the required values.

### Updating the record

If we want to update a record, then a simple update operation can be performed as:

int idToupdate = Convert.ToInt32(Request.QueryString["id"].ToString());

ContactsDb db = new ContactsDb();

Contact con = db.Contacts.SingleOrDefault(p => p.id == idToupdate);

con.phone = TextBox1.Text;

db.SaveChanges();

Once this code executes, the value of phone number will be updated by a new value which is retrieved fromTextBox1.

### Deleting a record

If we want to delete a particular record then we can perform a delete operation by using the DeleteObjectfunction. The following code snippet demonstrates the same:

*//delete this contact*

int idToupdate = Convert.ToInt32(Request.QueryString["id"].ToString());

ContactsDb db = new ContactsDb();

Contact con = db.Contacts.SingleOrDefault(p => p.id == idToupdate);

db.Contacts.DeleteObject(con);

db.SaveChanges();

Now that we have the basic CRUD operations performed on the database using the Entity Framework.



WEB PROGRAMMING LANGUAGES

**LAB-12**

# Localization / Globalization

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Creating the Website:

1. Open Visual Studio, Select File - > New Website- > ASP.NET Empty Website.
2. Open Solution Explorer and then right click on your Website and then Select Add - >Add New Item - > Web Form. Name this Web Form Home.aspx.

## Adding Controls:

1. Open toolbox and from toolbox select Label and Drop it on your Web Form.
2. Open toolbox and from toolbox select Button and Drop it on your Web Form.

## Adding Resource Files:

1. Right Click on Website in Solution Explorer and Select Add - > Add ASP.NET Folder - > App\_GlobalResources.
2. Right Click on App\_GlobalResources and Select Add ->Add New Item - > Resource. The Name of this File should be Resource.resx.
3. Open Resource File and Add these Key in it

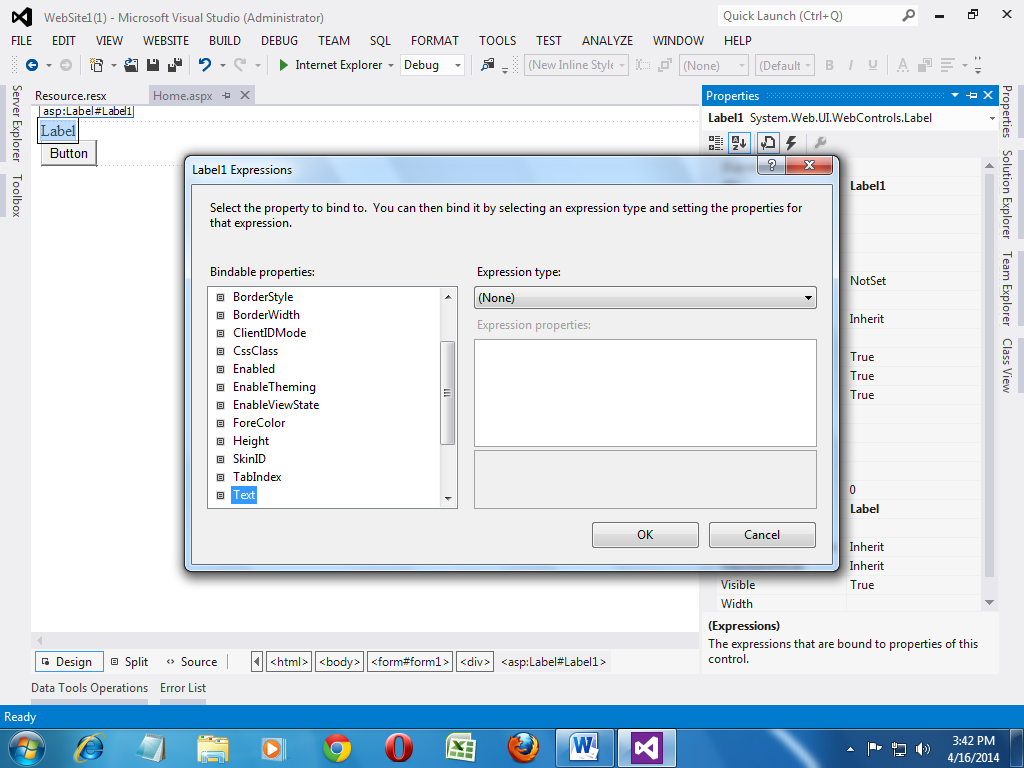
**Name** - > FColor **Value** ->Red

**Name** -> BColor **Value**->Yellow

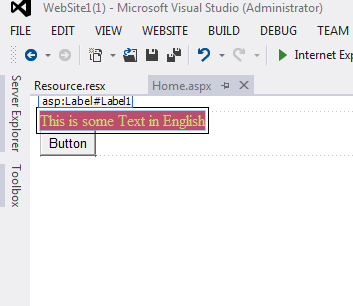
**Name** -> Text **Value** -> this is Some Text in English

## Link Resource Files:

1. Open Home.aspx, select your Label and then Open Properties Windows. From Properties Window Select Expressions , from right most end of expression click on the button.you will see expression window like this

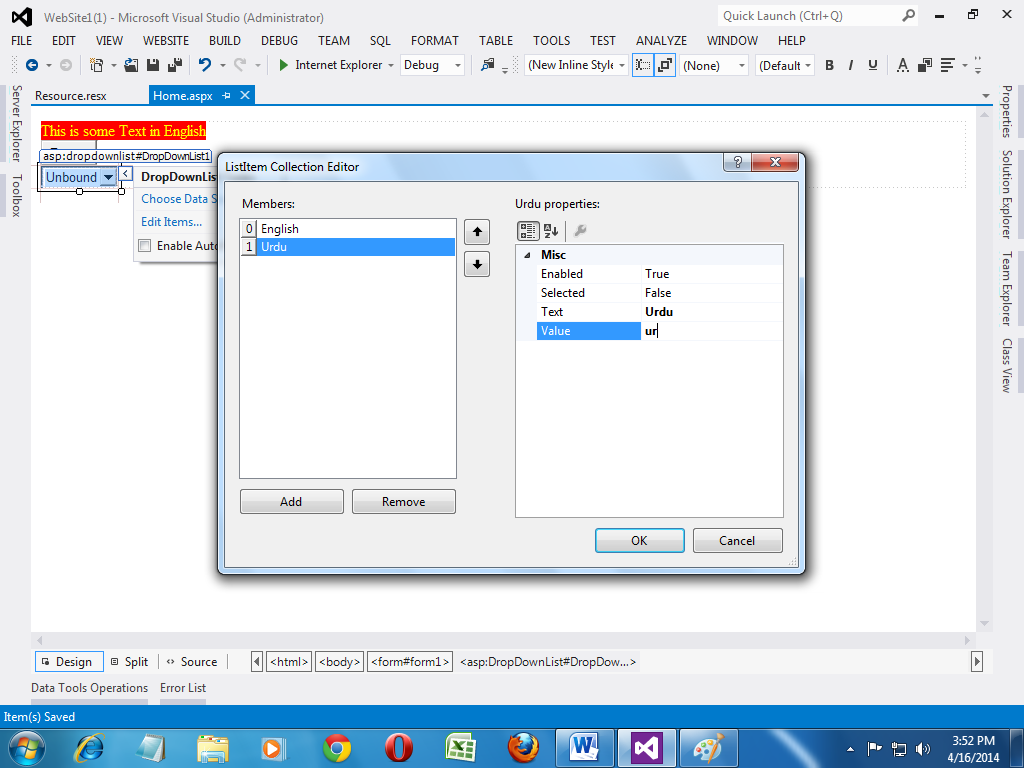


1. Select BackColor from Bindable Properties and Resources from Expression types. Inside Expression Properties write Resource in ClassKey and BColor in ResourceKey.
2. Select ForeColor from Bindable Properties and Resources from Expression types. Inside Expression Properties write Resource in ClassKey and FColor in ResourceKey.
3. Select Text from Bindable Properties and Resources from Expression types. Inside Expression Properties write Resource in ClassKey and Text in ResourceKey.
4. Press OK your Label should be like this



## Adding Language Selection:

1. Open toolbox select DropDownList and Drop it on your Web Form.
2. Please Open your Eyes and Look for a small tiny arrow symbol on right top corner of dropdownlist and click it . your will see an option Edit Item click on it.
3. Add New Item with Text = English and Value = en-Us and Text = Urdu and Value = ur



Forget about everything in your mind and once again click on little tiny symbol on drop down list and click on Enable Auto Post back. NOTE: **Please SAVE your Page by clicking CTRL+ S**

## Wrinting C# Code:

1. Open Solution Explorer and select Home.aspx. If you once again open your eyes you will see a arrow sign at left side od this page. Please click on it. You will be able to see [home.aspx.cs](http://home.aspx.cs) file . Open It.

NOTE: if you have realized that by mistake you have created a VB.NET Website then please goto Step 1.

1. Please Write this code where all using type of words are located.

using System.Globalization;

using System.Threading;

1. Please Write this Methods in your [Home.aspx.cs](http://Home.aspx.cs) File. Make sure it is not inside any other methods. It Should be in your class.

protectedoverridevoid InitializeCulture()

{

if (Request["DropDownList1"] != null)

{

string lang = Request["DropDownList1"];

Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(lang);

Thread.CurrentThread.CurrentUICulture = newCultureInfo(lang);

}

base.InitializeCulture();

}

Please Run your Website. If you don’t know how to run ASP.NET website please don’t waste your time by doing this exercise.

## Prepare Resource Files:

1. Goto Solution Explorer and right click on your Resource.resx file.Right Click on this file there is an option called Copy, please click on it. Now select App\_GlobalResource Folder and once again right click and select Paste.
2. Name this newly created File Resource.ur.resx.

NOTE: **we know your creativity but please don’t change the name of this file.**

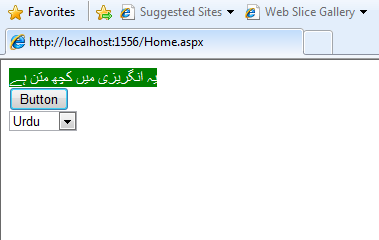
1. Open Resource.ur.resx file Change the Values as given below

**Name** - > FColor **Value** ->Green

**Name** -> BColor **Value**->White

**Name** -> Text **Value ->**یہ انگریزی میں کچھ متن ہے

1. Once again Run your Website. You should be able to see this output.



NOTE: **if you are not getting this result. Please goto Step 1.**



WEB PROGRAMMING LANGUAGES

**LAB-13**

# ASP.NET Identity

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

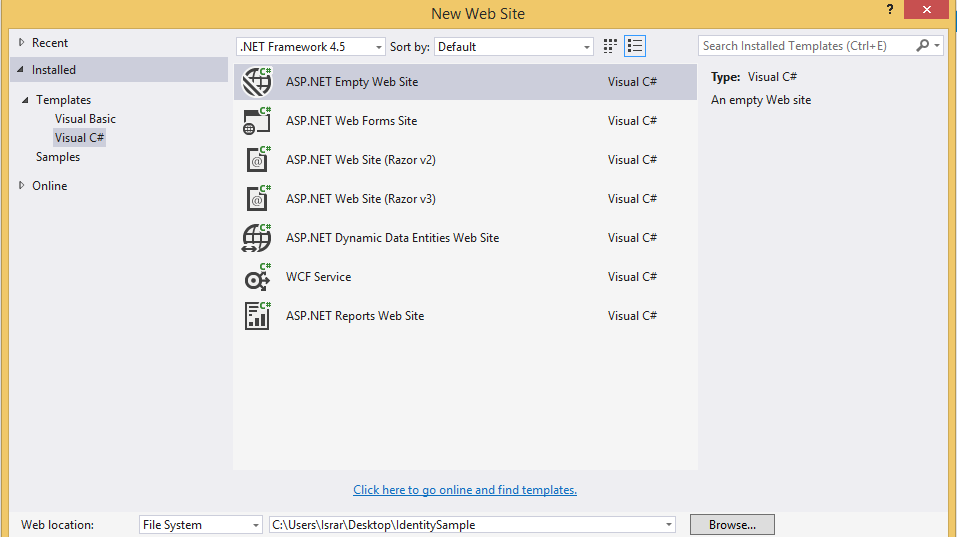
**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

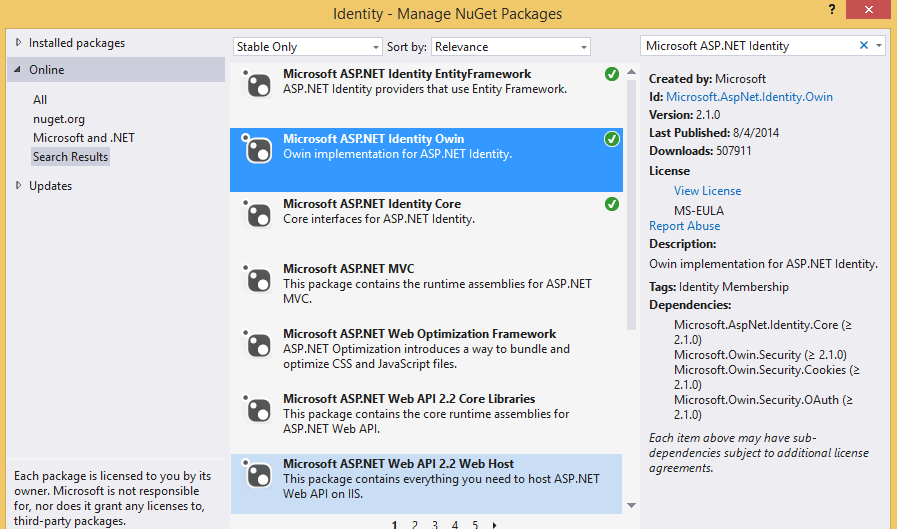
## Creating The Website:

1. Start by installing and running [Visual Studio 2013](http://go.microsoft.com/fwlink/?LinkId=306566).
2. Click **New Projec**t from the Start page, or you can use the menu and select **File**, and then **New Project**.
3. Select **Visual C# i**n the left pane, then **Web** and then select **ASP.NET Web Application**. Name your project "IdentitySample" and then click **OK**.

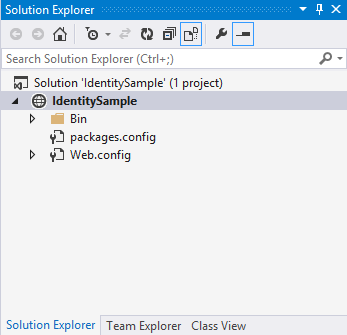


1. In the **New ASP.NET Project** dialog, select the **Empty** template.
2. In Solution Explorer, right-click your project and select **Manage NuGet Packages**. In the search text box dialog, type “Microsoft ASP.NET Identity”.

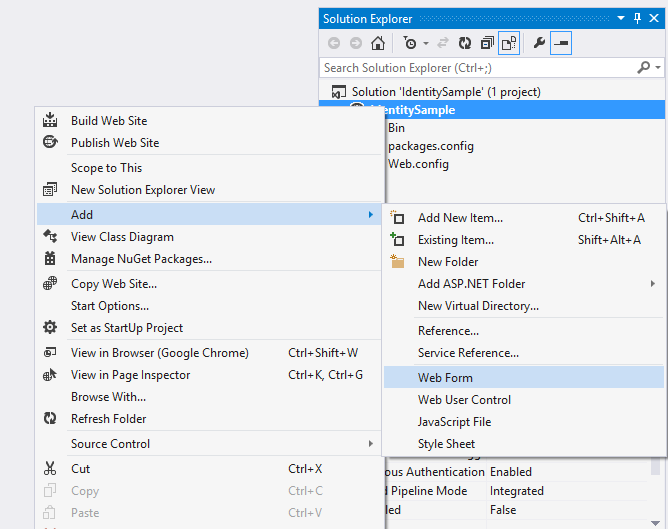
## Add NuGET Packages:



Click install for this package. After installation you will see these changes in solution Explorer.



1. In **Solution Explorer**, right-click your project and click **Add**, and then **Web Form**.



1. In the **Specify Name for Item** dialog box, name the new web form **Register**, and then click **OK**
2. Replace the markup in the generated *Register.aspx* file with the code below. The code changes are highlighted.

<bodystyle="font-family: Arial, Helvetica, sans-serif; font-size: small">

<formid="form1"runat="server">

<div>

<h4style="font-size: medium">Register a new user</h4>

<hr/>

<p>

<asp:Literalrunat="server"ID="StatusMessage"/>

</p>

<divstyle="margin-bottom:10px">

<asp:Labelrunat="server"AssociatedControlID="UserName">User name</asp:Label>

<div>

<asp:TextBoxrunat="server"ID="UserName"/>

</div>

</div>

<divstyle="margin-bottom:10px">

<asp:Labelrunat="server"AssociatedControlID="Password">Password</asp:Label>

<div>

<asp:TextBoxrunat="server"ID="Password"TextMode="Password"/>

</div>

</div>

<divstyle="margin-bottom:10px">

<asp:Labelrunat="server"AssociatedControlID="ConfirmPassword">Confirm password</asp:Label>

<div>

<asp:TextBoxrunat="server"ID="ConfirmPassword"TextMode="Password"/>

</div>

</div>

<div>

<div>

<asp:Buttonrunat="server"OnClick="CreateUser\_Click"Text="Register"/>

</div>

</div>

</div>

</form>

</body>

1. Open the Register.aspx.cs file and replace the contents of the file with the following code:

protectedvoid CreateUser\_Click(object sender, EventArgs e)

{

// Default UserStore constructor uses the default connection string named: DefaultConnection

var userStore = newUserStore<IdentityUser>();

var manager = newUserManager<IdentityUser>(userStore);

var user = newIdentityUser() { UserName = UserName.Text };

IdentityResult result = manager.Create(user, Password.Text);

if (result.Succeeded)

{

StatusMessage.Text = string.Format("User {0} was created successfully!", user.UserName);

}

else

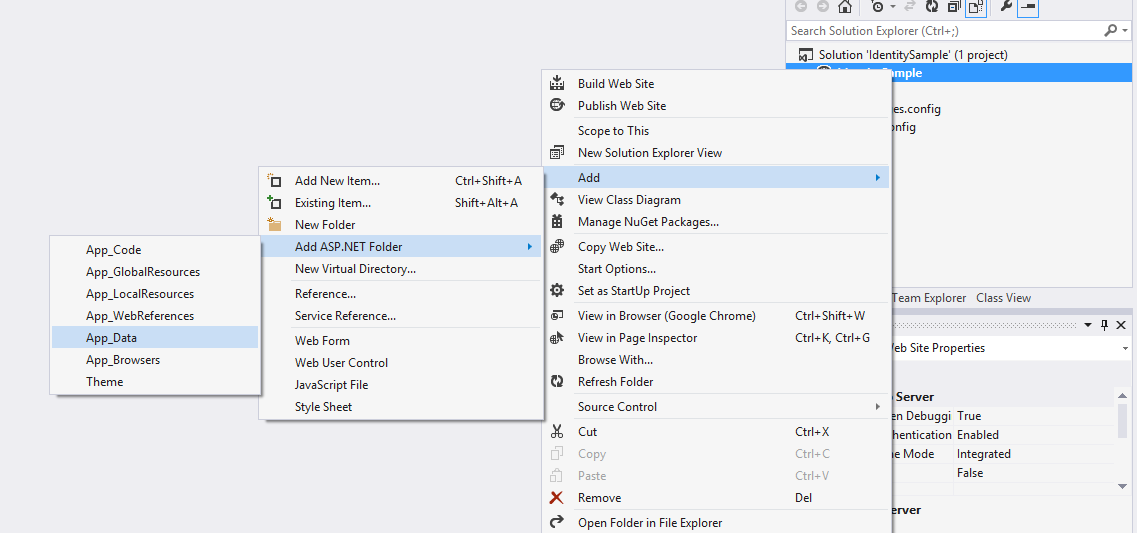
{

StatusMessage.Text = result.Errors.FirstOrDefault();

}

}

1. In **Solution Explorer**, right-click your project and click **Add**, **Add ASP.NET Folder** and then **App\_Data**.



1. Open the Web.config file and add a connection string entry for the database we will use to store user information. The database will be created at runtime by EntityFramework for the Identity entities. The connection string is similar to one created for you when you create a new Web Forms project. The highlighted code shows the markup you should add:

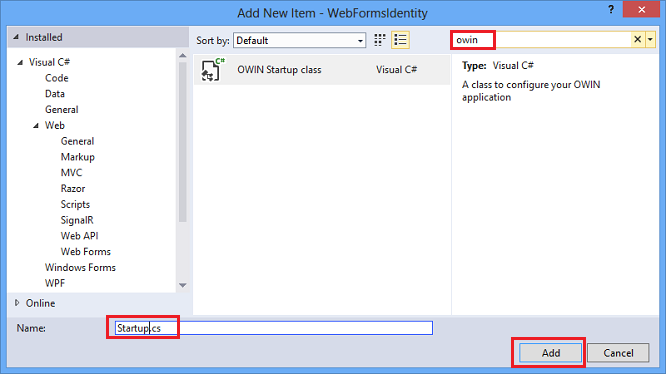
<connectionStrings>

<addname="DefaultConnection"connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\WebFormsIdentity.mdf;Initial Catalog=WebFormsIdentity;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

1. Right click file Register.aspx in your project and select **Set as Start Page**. Press Ctrl + F5 to build and run the web application. Enter a new user name and password and then click on **Register**.
2. In **Solution Explorer**, right-click your project, click **Add**, and then **Add New Item**. In the search text box dialog, type “owin”.  Name the class "Startup" and click **Add**.



1. In the Startup.cs file, add the highlighted code shown below to configure OWIN cookie authentication.

publicvoid Configuration(IAppBuilder app)

{

// For more information on how to configure your application, visit http://go.microsoft.com/fwlink/?LinkID=316888

app.UseCookieAuthentication(newCookieAuthenticationOptions

{

AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,

LoginPath = newPathString("/Login")

});

}

1. Open the *Register.cs* file and add the following code which will log in the user when registration succeeds. The changes are highlighted below.

protectedvoid CreateUser\_Click(object sender, EventArgs e)

{

var userStore = newUserStore<IdentityUser>();

var manager = newUserManager<IdentityUser>(userStore);

var user = newIdentityUser() { UserName = UserName.Text };

IdentityResult result = manager.Create(user, Password.Text);

if (result.Succeeded)

{

var authenticationManager = HttpContext.Current.GetOwinContext().Authentication;

var userIdentity = manager.CreateIdentity(user, DefaultAuthenticationTypes.ApplicationCookie);

authenticationManager.SignIn(newAuthenticationProperties() { }, userIdentity);

Response.Redirect("~/Login.aspx");

}

else

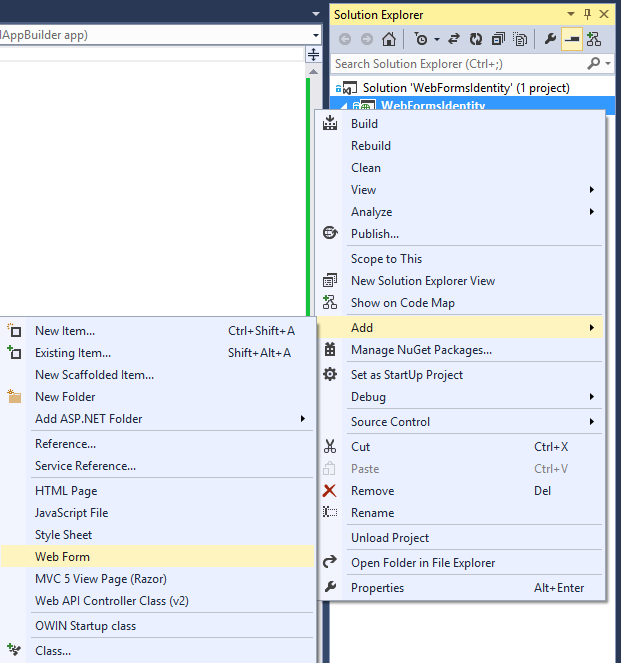
{

StatusMessage.Text = result.Errors.FirstOrDefault();

}

}

1. In **Solution Explorer**, right-click your project click **Add**, and then **Web Form**. Name the web form **Login**.



1. Replace the contents of theLogin.aspx file with the following code:

<bodystyle="font-family: Arial, Helvetica, sans-serif; font-size: small">

<formid="form1"runat="server">

<div>

<h4style="font-size: medium">Log In</h4>

<hr/>

<asp:PlaceHolderrunat="server"ID="LoginStatus"Visible="false">

<p>

<asp:Literalrunat="server"ID="StatusText"/>

</p>

</asp:PlaceHolder>

<asp:PlaceHolderrunat="server"ID="LoginForm"Visible="false">

<divstyle="margin-bottom: 10px">

<asp:Labelrunat="server"AssociatedControlID="UserName">User name</asp:Label>

<div>

<asp:TextBoxrunat="server"ID="UserName"/>

</div>

</div>

<divstyle="margin-bottom: 10px">

<asp:Labelrunat="server"AssociatedControlID="Password">Password</asp:Label>

<div>

<asp:TextBoxrunat="server"ID="Password"TextMode="Password"/>

</div>

</div>

<divstyle="margin-bottom: 10px">

<div>

<asp:Buttonrunat="server"OnClick="SignIn"Text="Log in"/>

</div>

</div>

</asp:PlaceHolder>

<asp:PlaceHolderrunat="server"ID="LogoutButton"Visible="false">

<div>

<div>

<asp:Buttonrunat="server"OnClick="SignOut"Text="Log out"/>

</div>

</div>

</asp:PlaceHolder>

</div>

</form>

</body>

1. Replace the contents of the Login.aspx.cs file with the following:

protectedvoid Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

if (User.Identity.IsAuthenticated)

{

StatusText.Text = string.Format("Hello {0}!!", User.Identity.GetUserName());

LoginStatus.Visible = true;

LogoutButton.Visible = true;

}

else

{

LoginForm.Visible = true;

}

}

}

protectedvoid SignIn(object sender, EventArgs e)

{

var userStore = newUserStore<IdentityUser>();

var userManager = newUserManager<IdentityUser>(userStore);

var user = userManager.Find(UserName.Text, Password.Text);

if (user != null)

{

var authenticationManager = HttpContext.Current.GetOwinContext().Authentication;

var userIdentity = userManager.CreateIdentity(user, DefaultAuthenticationTypes.ApplicationCookie);

authenticationManager.SignIn(newAuthenticationProperties() { IsPersistent = false }, userIdentity);

Response.Redirect("~/Login.aspx");

}

else

{

StatusText.Text = "Invalid username or password.";

LoginStatus.Visible = true;

}

}

protectedvoid SignOut(object sender, EventArgs e)

{

var authenticationManager = HttpContext.Current.GetOwinContext().Authentication;

authenticationManager.SignOut();

Response.Redirect("~/Login.aspx");

}

1. Press **Ctrl + F5** to build and run the web application. Enter a new user name and password and then click on**Register**.



WEB PROGRAMMING LANGUAGES

**LAB-14**

# ASP.NET MVC

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_**

**Marks Obtained \_\_\_\_\_\_\_\_\_\_\_\_**

**Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Creating the Website:

Create a New Project in Visual Studio 2012.

Select Visual C# and then ASP.NET MVC 4 Web Application, Enter MVCDemoApplication as your Project Name.

Select Internet Application as your template.

## Add Database:

Right Click on App\_Data and select Add New Item, Select SQL Server Database. You can set any name your like to this database.

Switch to Server Explorer , Expand .mdf file and then right click on the Tables folder . select Add new Table.

In the design window add three columns (Name, Email, DOB) and make id an identity field. Change the name of your Table to Emp.

## Add LINQ to SQL Classes:

Right Click on Model Folder and select Add New Item. Select LINQ to SQL Classes and name this file as Emp.dbml.

Drag and Drop Emp Table from Server Explorer to this dbml File.

## Add Controller:

Right Click on Controller and Select Add New Controller, Type EmpController as your Controller Name and Select MVC template with empty read write actions.

Write This Code in Index Action of Controller

EmpDataContext db = newEmpDataContext();

var emp = from e in db.Emps

select e;

return View(emp);

Make sure you have used using MVCDemoAplication.Models and build your Application.

## Add Views:

Right Click inside Index Action and Select Add View. Select Strongly Typed View and Select Emp Model. After that select List Template and Click on Add.

Verify your code by running Application and then adding /Emp/Index in the URL.

Right Click inside Create Action and Select Add View. This time you should select Create template and then Click on Add Button.

Change the Next Create Method with this Code

[HttpPost]

publicActionResult Create(Emp emp)

{

try

{

// TODO: Add insert logic here

EmpDataContext db = newEmpDataContext();

Emp empObj = newEmp

{

Name = emp.Name,

Email = emp.Email,

DOB = emp.DOB

};

db.Emps.InsertOnSubmit(empObj);

db.SubmitChanges();

return RedirectToAction("Index");

}

catch

{

return View();

}

}

Change your Edit Method and Write this Code

publicActionResult Edit(int id)

{

EmpDataContext db = newEmpDataContext();

Emp empObj = db.Emps.First(e => e.Id == id);

return View(empObj);

}

Now Right Click inside Edit method and select Add View , this time select Edit Template.

Change other Edit Method and Write this Code

[HttpPost]

publicActionResult Edit(int id, Emp emp)

{

try

{

// TODO: Add update logic here

EmpDataContext db = newEmpDataContext();

Emp empObj = db.Emps.First(e => e.Id == id);

empObj.Name = emp.Name;

empObj.Email = emp.Email;

empObj.DOB = emp.DOB;

db.SubmitChanges();

return RedirectToAction("Index");

}

catch

{

return View();

}

}

Verify your code by running application and then executing Edit Action.

Change your Delete Method and Write this Code

publicActionResult Delete(int id)

{

EmpDataContext db = newEmpDataContext();

Emp empObj = db.Emps.First(e => e.Id == id);

return View(empObj);

}

Now Right Click inside Delete method and select Add View , this time select Delete Template.

Change other Delete Method and Write this Code

[HttpPost]

publicActionResult Delete(int id, FormCollection collection)

{

try

{

// TODO: Add delete logic here

EmpDataContext db = newEmpDataContext();

Emp empObj = db.Emps.First(e => e.Id == id);

db.Emps.DeleteOnSubmit(empObj);

db.SubmitChanges();

return RedirectToAction("Index");

}

catch

{

return View();

}

Verify your code by running application and then executing Delete Action.

Change your Detail Method and Write this Code

publicActionResult Details(int id)

{

EmpDataContext db = newEmpDataContext();

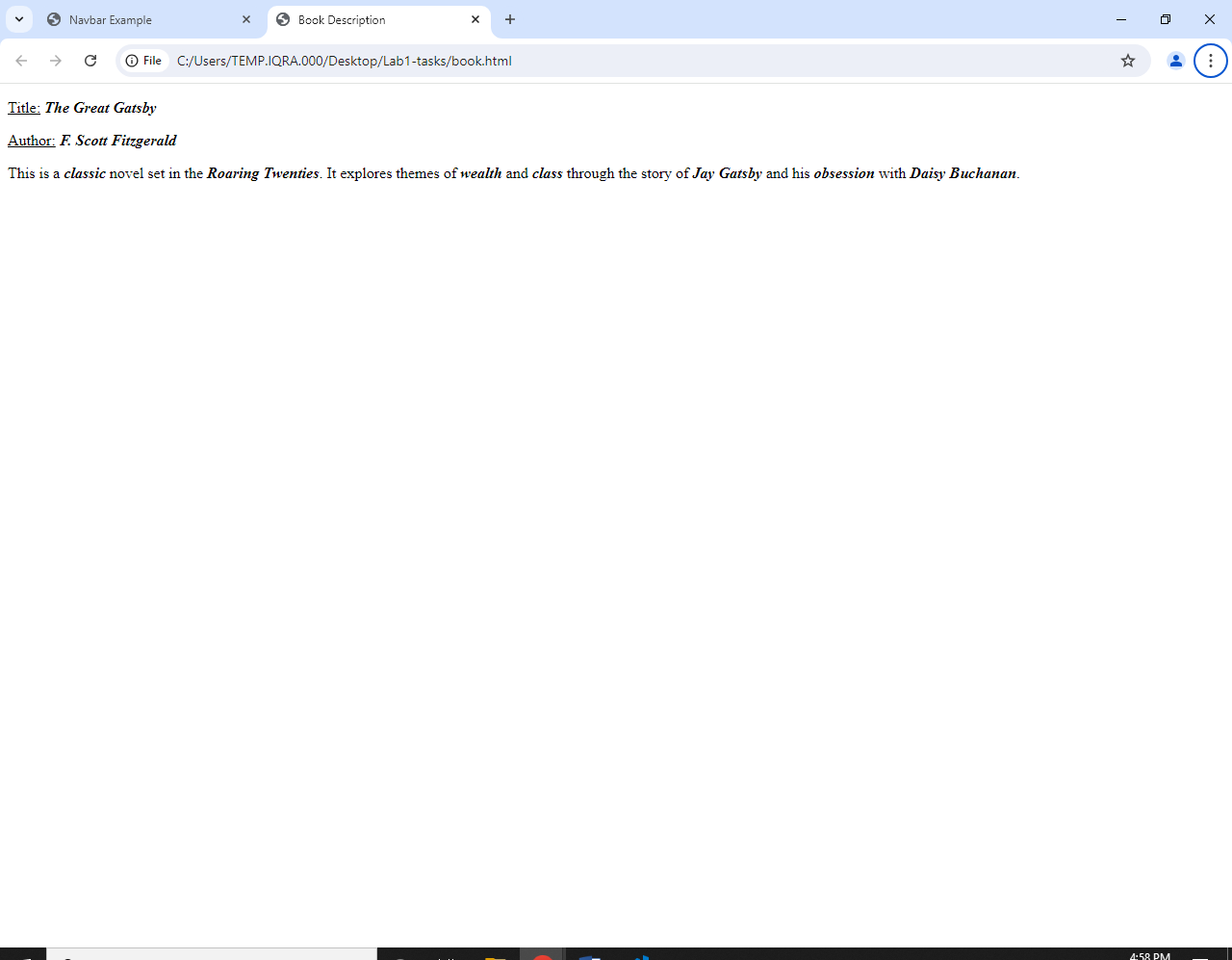
Emp empObj = db.Emps.First(e => e.Id == id);

return View(empObj);

}

Now Right Click inside Detail method and select Add View , this time select Detail Template.

Lab 1 Tasks



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Book Description</title>

</head>

<body>

<p><u>Title:</u> <b><i>The Great Gatsby</i></b></p>

<p><u>Author:</u> <b><i>F. Scott Fitzgerald</i></b></p>

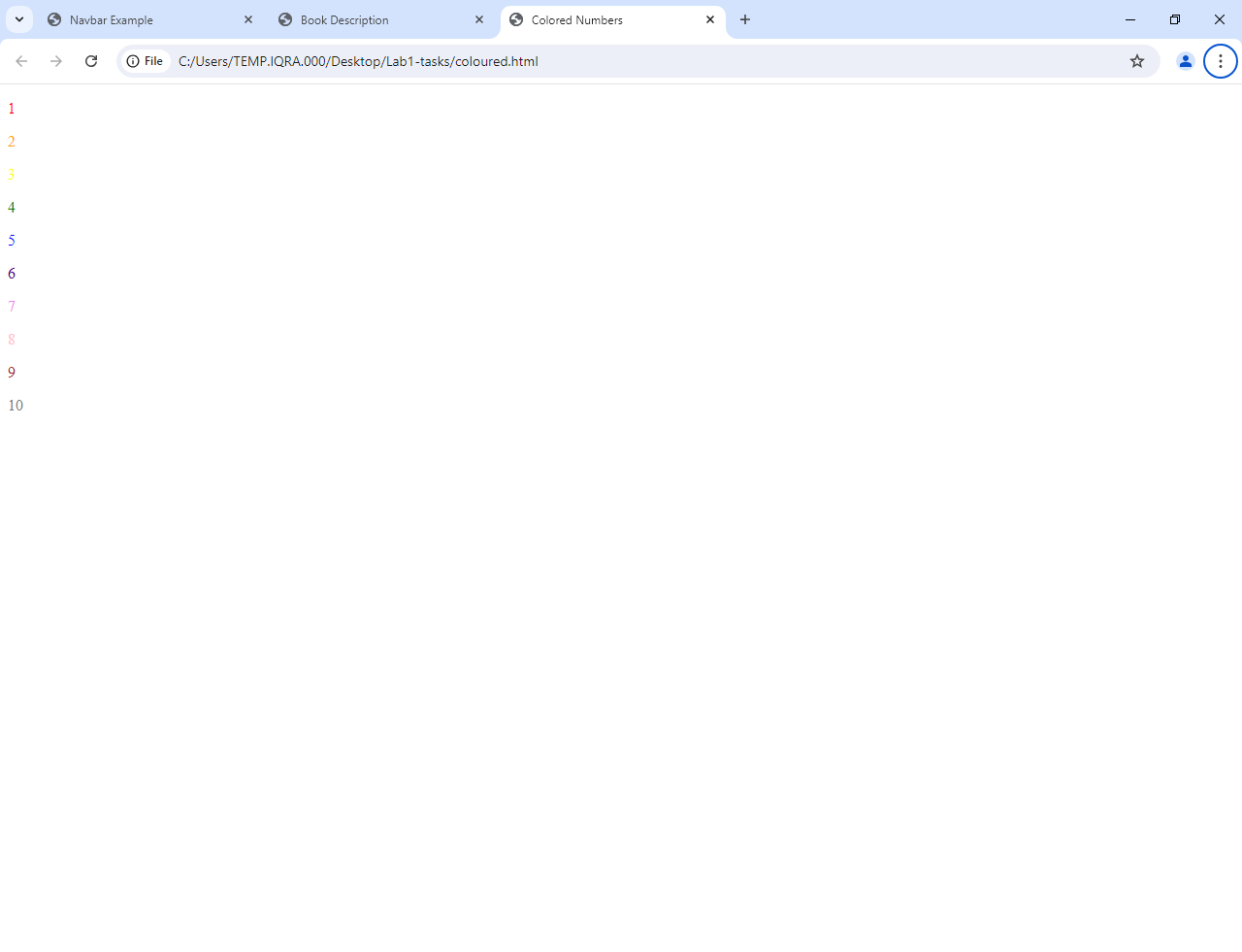
<p>This is a <b><i>classic</i></b> novel set in the <b><i>Roaring Twenties</i></b>. It explores themes of <b><i>wealth</i></b>

and <b><i>class</i></b> through the story of <b><i>Jay Gatsby</i></b> and his <b><i>obsession</i></b> with <b><i>Daisy

Buchanan</i></b>.</p>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Colored Numbers</title>

<style>

.color1 {

color: red;

}

.color2 {

color: orange;

}

.color3 {

color: yellow;

}

.color4 {

color: green;

}

.color5 {

color: blue;

}

.color6 {

color: indigo;

}

.color7 {

color: violet;

}

.color8 {

color: pink;

}

.color9 {

color: brown;

}

.color10 {

color: gray;

}

</style>

</head>

<body>

<p class="color1">1</p>

<p class="color2">2</p>

<p class="color3">3</p>

<p class="color4">4</p>

<p class="color5">5</p>

<p class="color6">6</p>

<p class="color7">7</p>

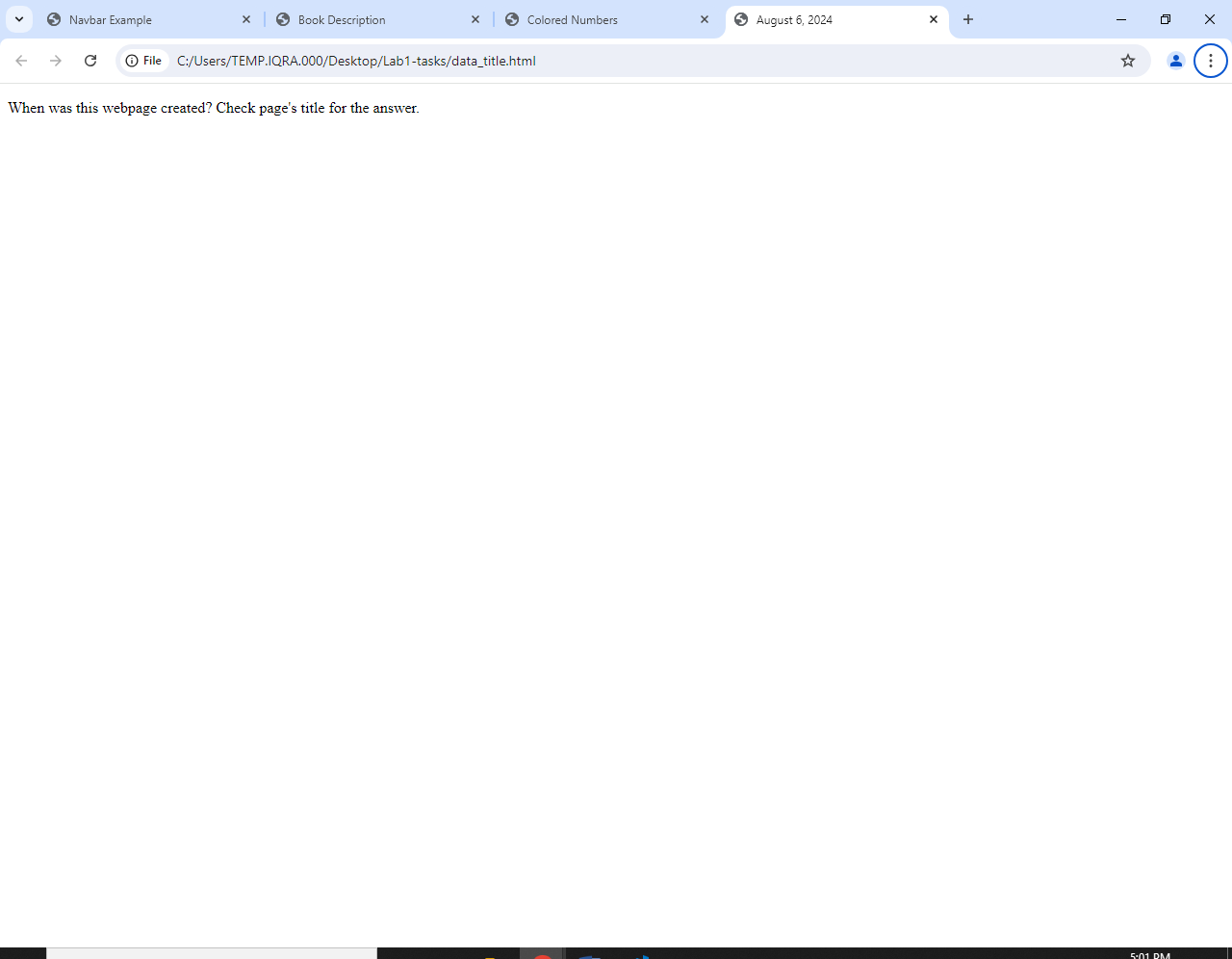
<p class="color8">8</p>

<p class="color9">9</p>

<p class="color10">10</p>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>August 6, 2024</title> <!-- Update this to the current date -->

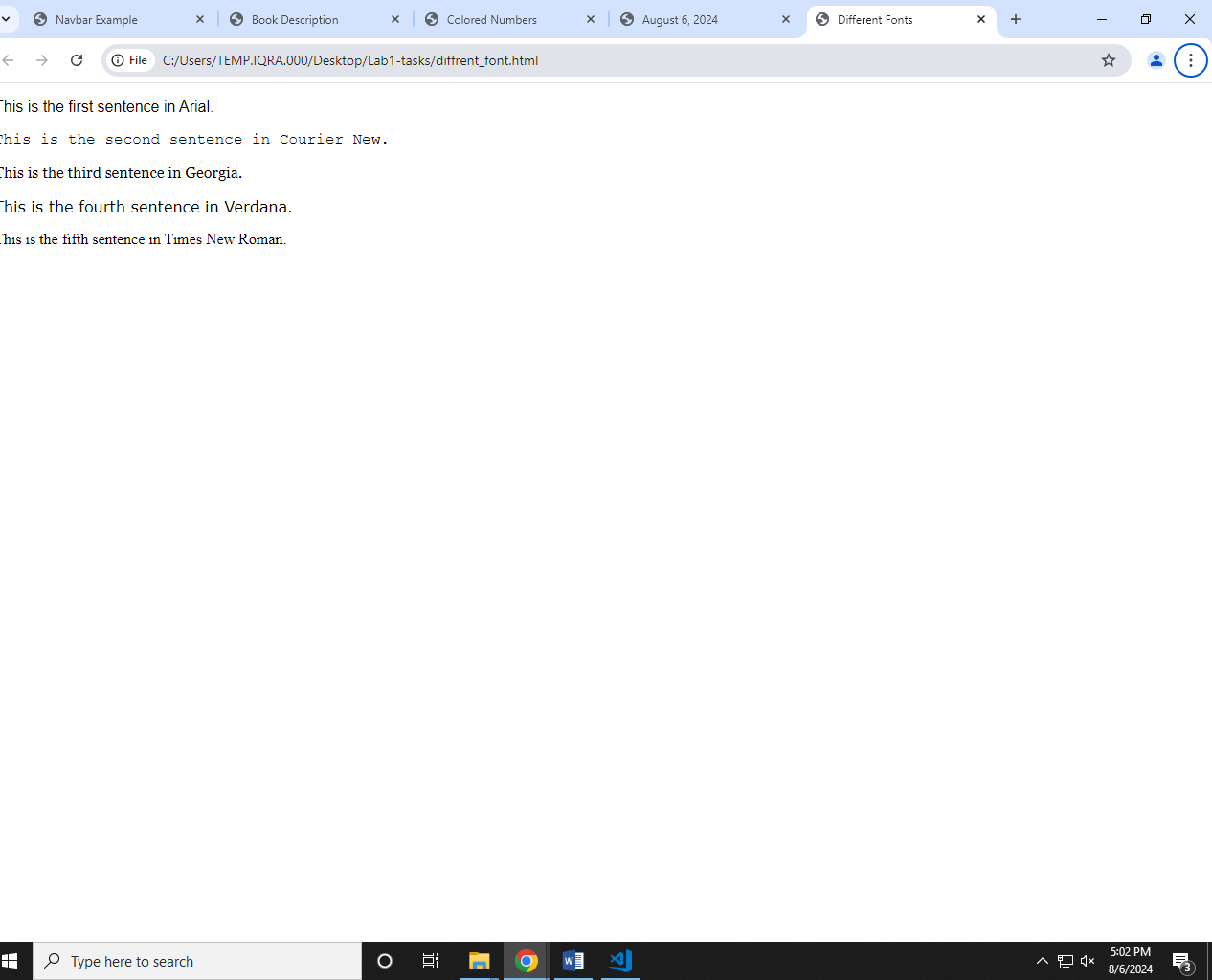
</head>

<body>

<p>When was this webpage created? Check page's title for the answer.</p>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Different Fonts</title>

<style>

.font1 {

font-family: Arial, sans-serif;

}

.font2 {

font-family: 'Courier New', monospace;

}

.font3 {

font-family: 'Georgia', serif;

}

.font4 {

font-family: Verdana, sans-serif;

}

.font5 {

font-family: 'Times New Roman', serif;

}

</style>

</head>

<body>

<p class="font1">This is the first sentence in Arial.</p>

<p class="font2">This is the second sentence in Courier New.</p>

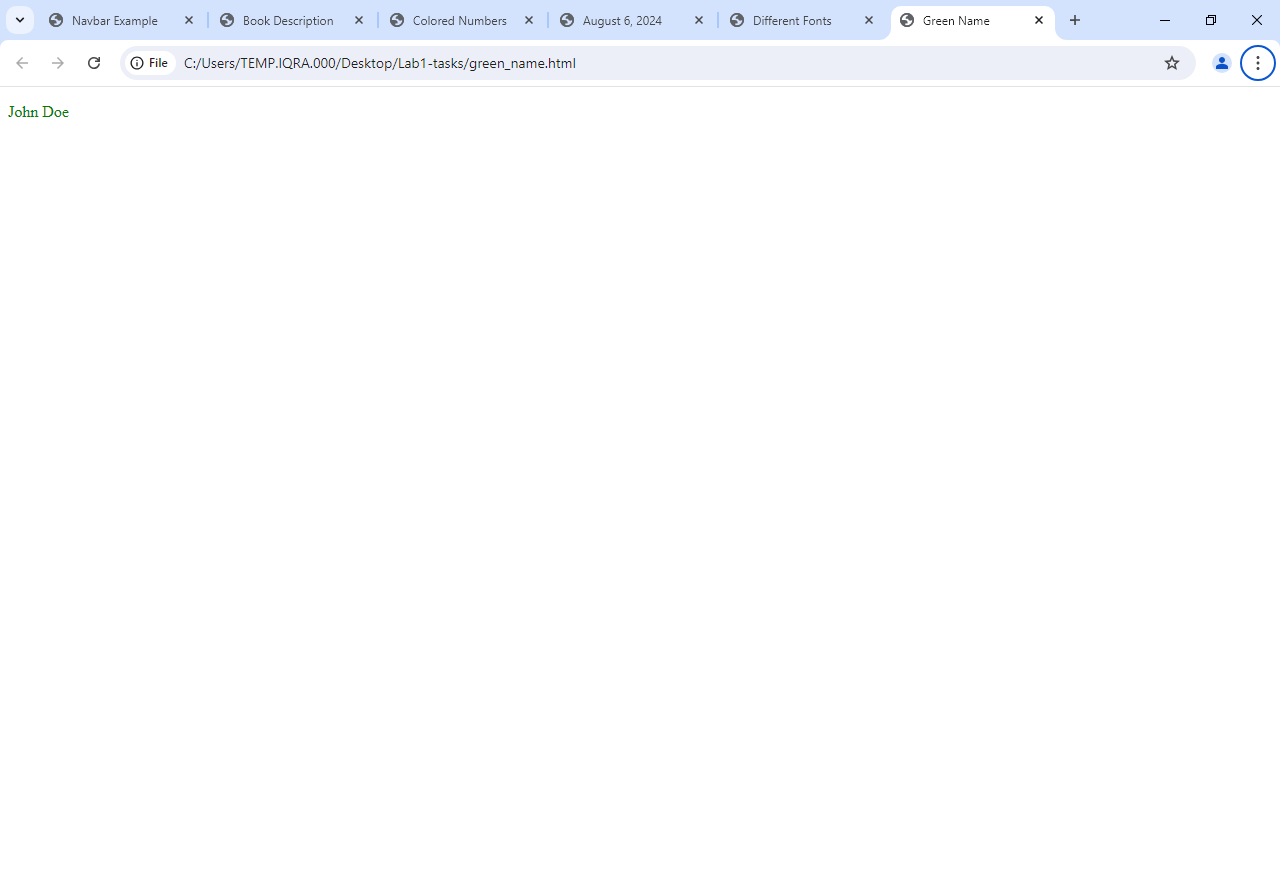
<p class="font3">This is the third sentence in Georgia.</p>

<p class="font4">This is the fourth sentence in Verdana.</p>

<p class="font5">This is the fifth sentence in Times New Roman.</p>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Green Name</title>

<style>

.green-text {

color: green;

}

</style>

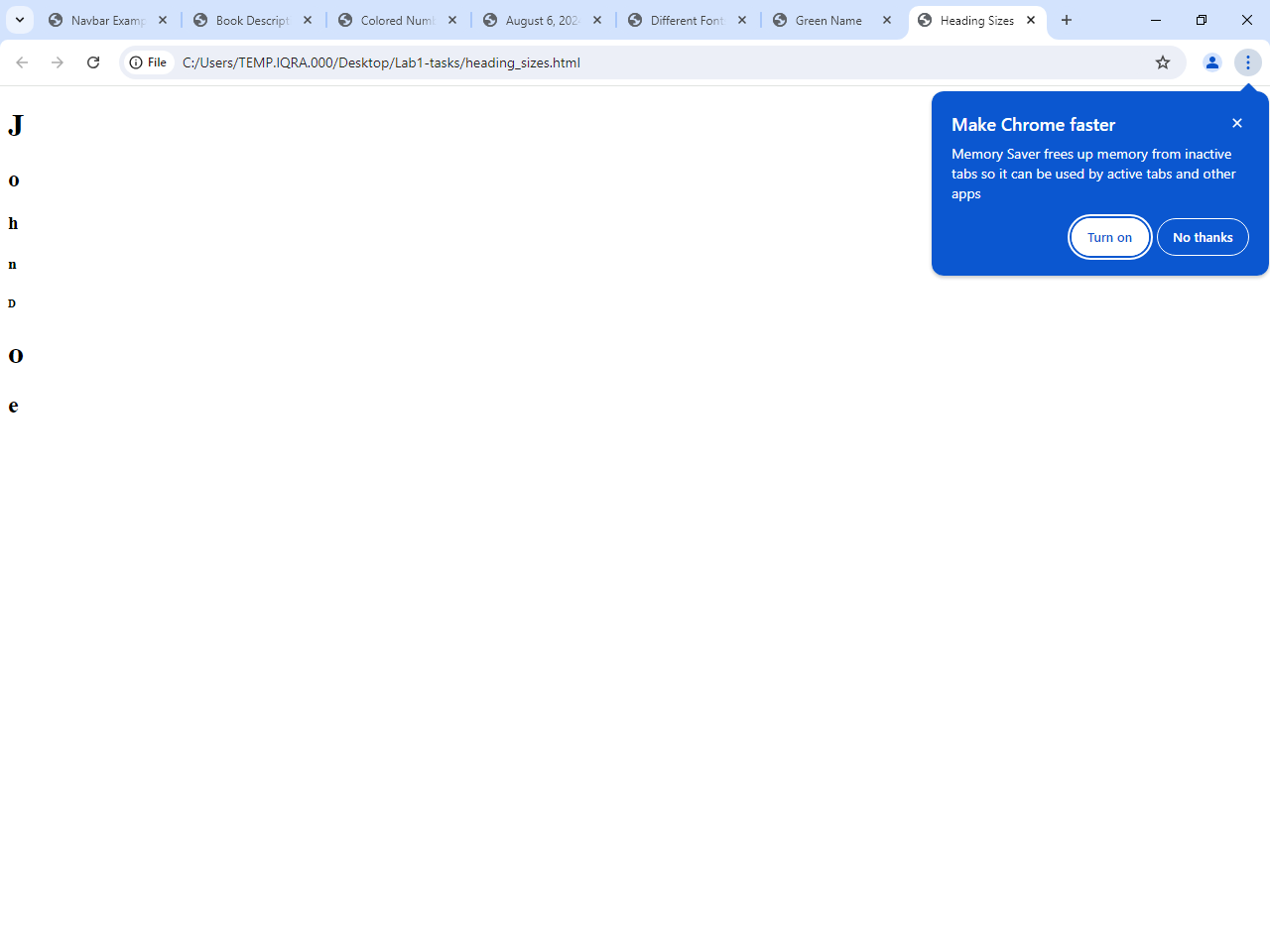
</head>

<body>

<p class="green-text">John Doe</p>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Heading Sizes</title>

</head>

<body>

<h1>J</h1>

<h2>o</h2>

<h3>h</h3>

<h4>n</h4>

<h5> </h5>

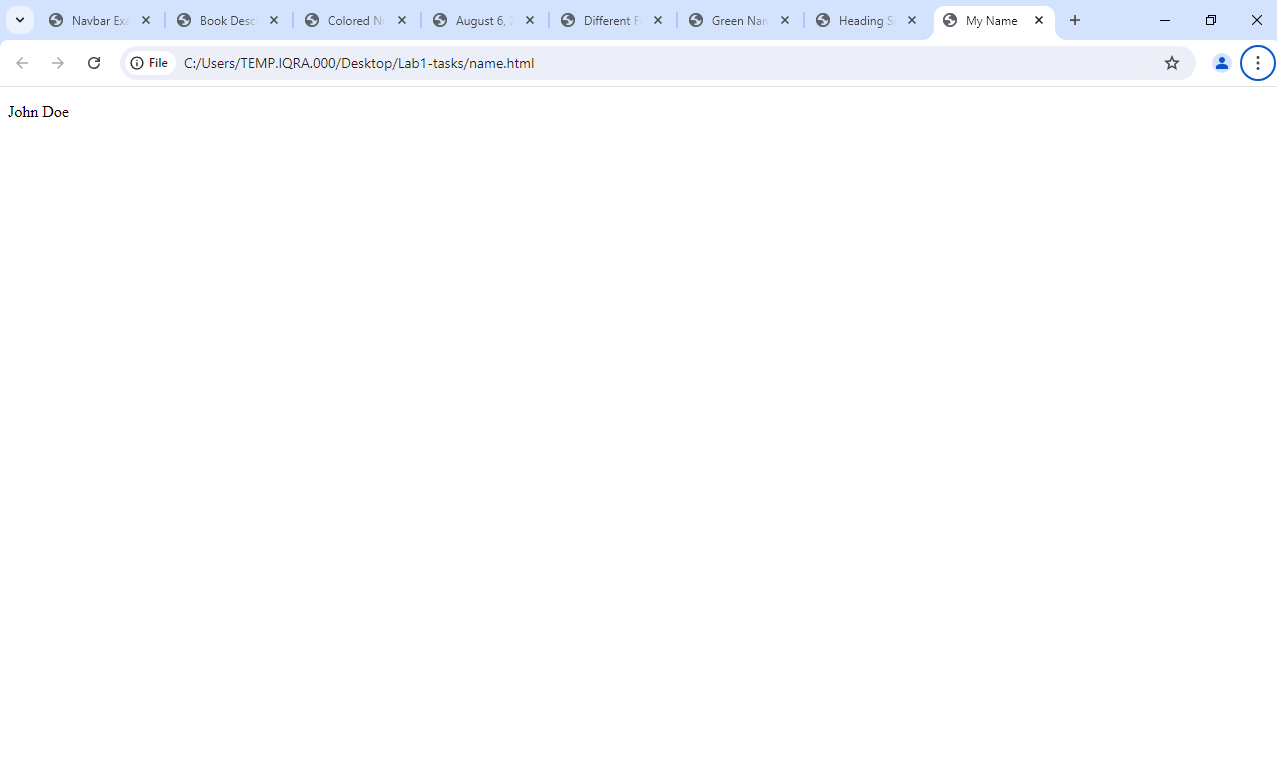
<h6>D</h6>

<h1>o</h1>

<h2>e</h2>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My Name</title>

</head>

<body>

<p>John Doe</p>

</body>

</html>

No Head

<!DOCTYPE html>

<html lang="en">

<body>

<p>This is a webpage without a head section.</p>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Numbers 1-10</title>

</head>

<body>

<p>1</p>

<p>2</p>

<p>3</p>

<p>4</p>

<p>5</p>

<p>6</p>

<p>7</p>

<p>8</p>

<p>9</p>

<p>10</p>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Tahoma Name</title>

<style>

.tahoma-font {

font-family: Tahoma, sans-serif;

}

</style>

</head>

<body>

<p class="tahoma-font">John Doe</p>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>This is a webpage</title>

</head>

<body>

<p>Welcome to the webpage!</p>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Webpage with Head</title>

</head>

<body>

<p>This is a webpage with a head section.</p>

</body>

</html>