

Exercises: HTML5 and CSS

Problems for exercises and homework for the ["C# Web Basics" course @ SoftUni.](#)

After following the steps for this exercise, you should have built the base for a blog.

Note that this exercise uses bootstrap extensively, so every HTML code you write will be instantly stylized and will look good off the shelf. This is not the case when you write plain HTML without ready styles.

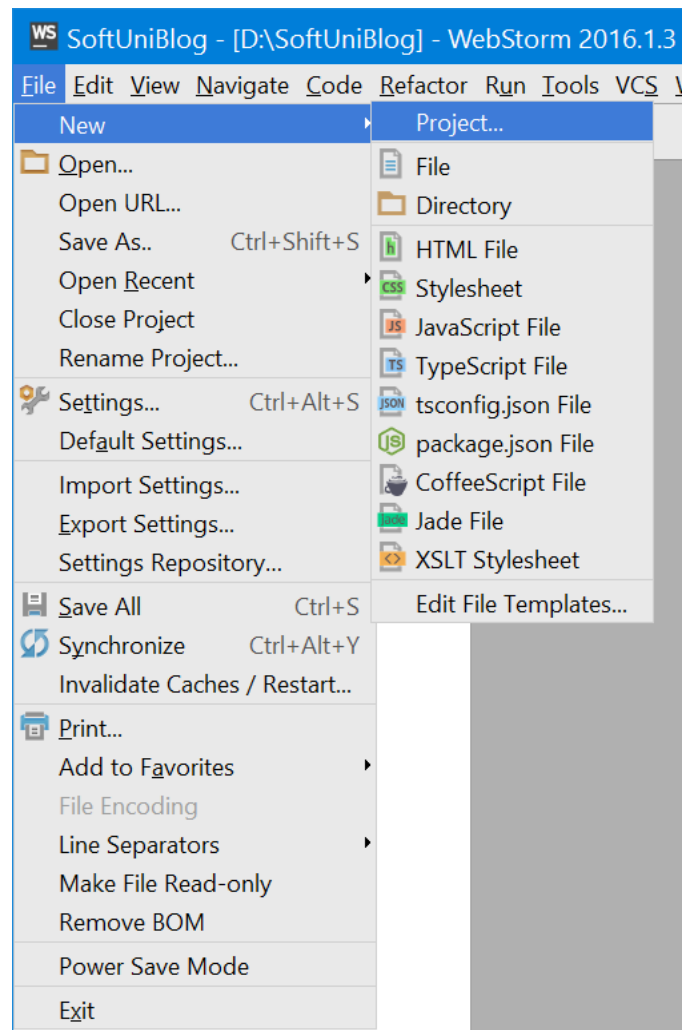
However, pay attention to the **<tags>** that we are using. Almost all **<tags>** that we are going to write use a **class=""** attribute (that is **<tag class="some class">**). The **class=""** attribute points to the corresponding selector in the **style.css** file. This is exactly the same as when you are writing your own styles but in our case, they are provided by the open source framework called [bootstrap](#).

With this in mind, let's get to work.

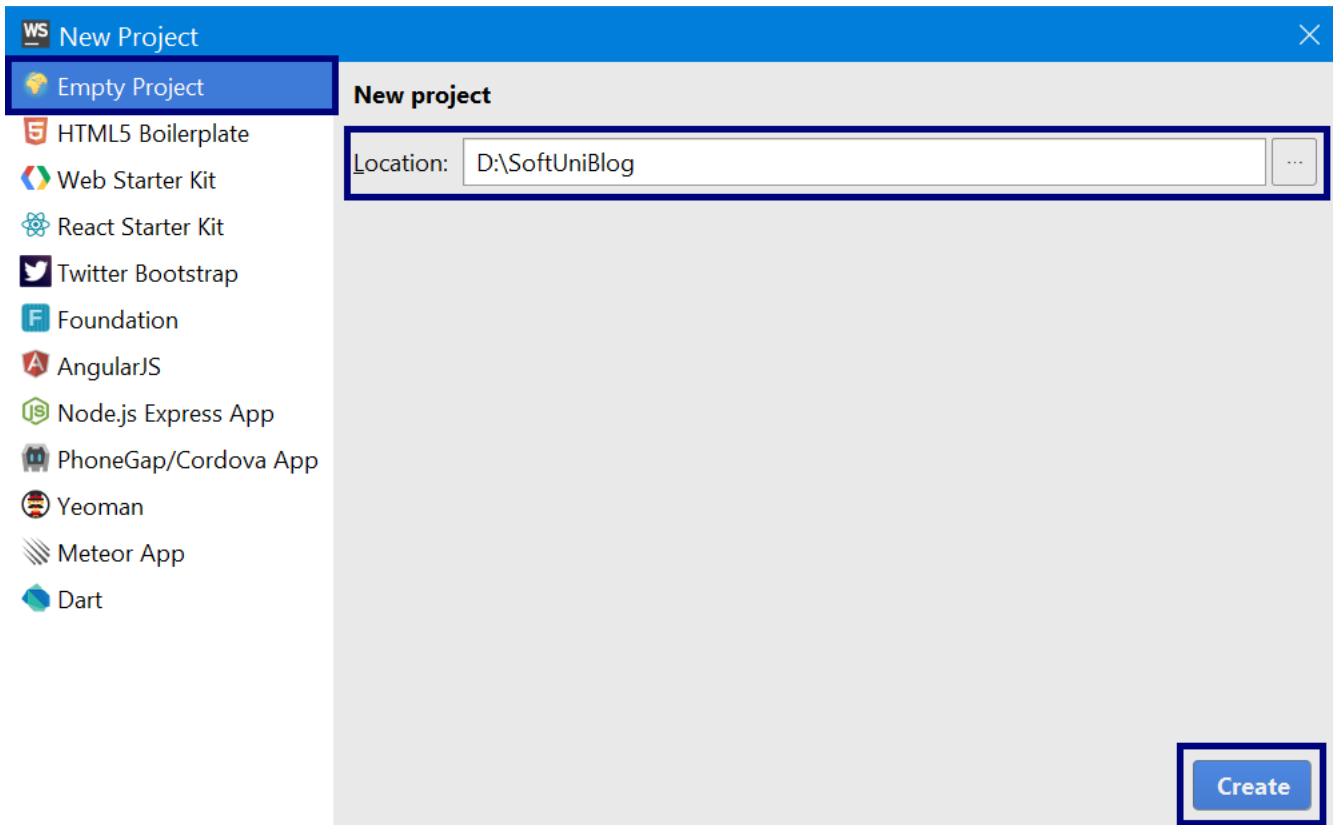
I. Initial Setup

1. Create New Project

Open WebStorm and create new project from **File -> New... -> Project**

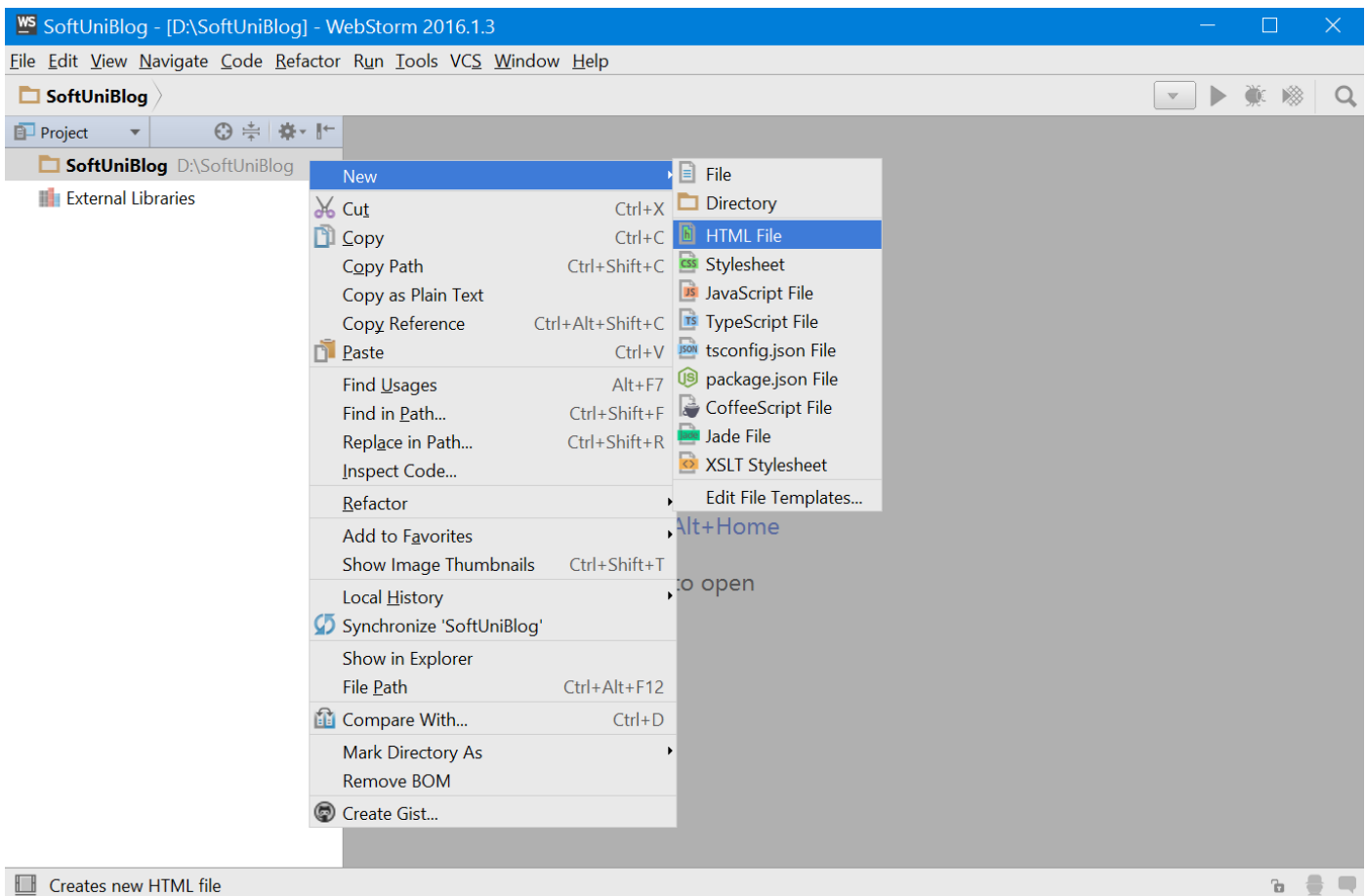


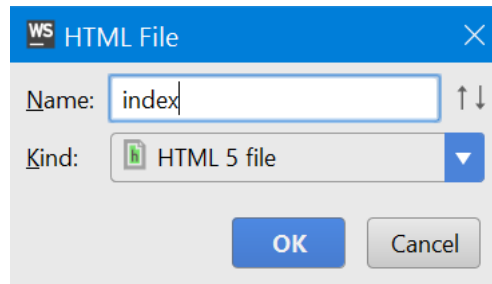
Now select **Empty Project** and choose a **destination folder** of your project



2. Add Blog Home Page

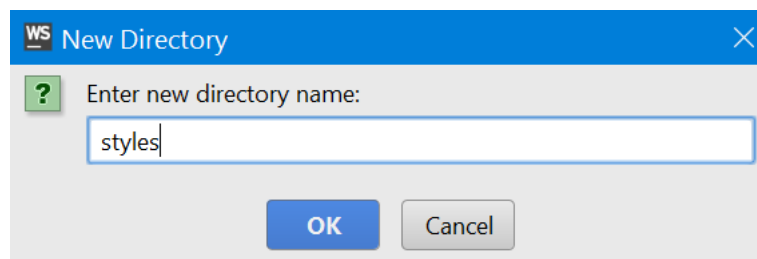
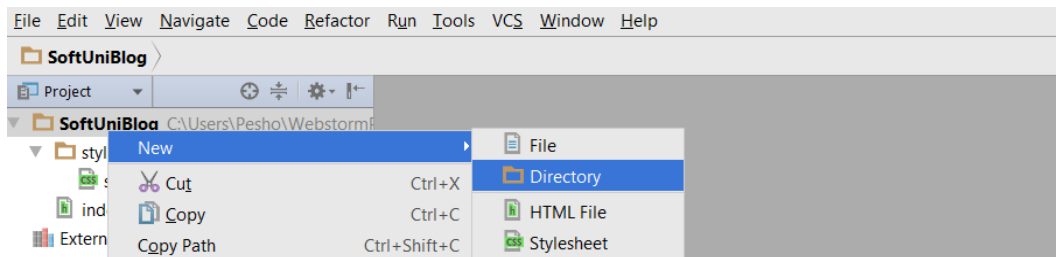
Create new HTML file called **index.html** that will be our **home page**.





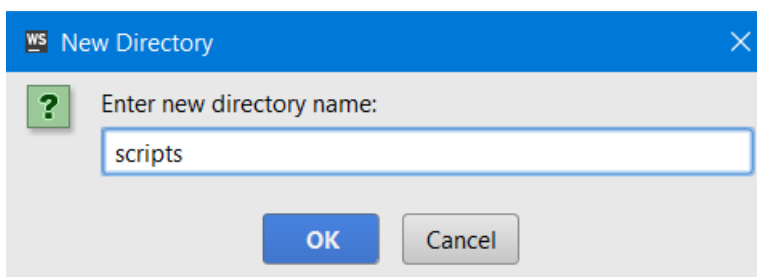
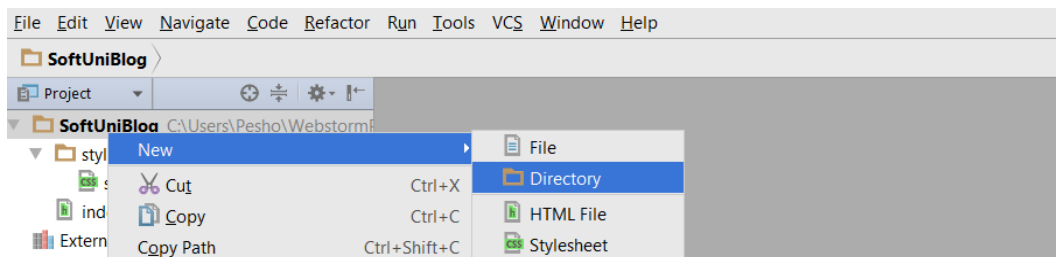
3. Create a Directory for Styles

Create **new directory** in our project that will keep our **CSS files** and name it **styles**



4. Create a Directory for Scripts

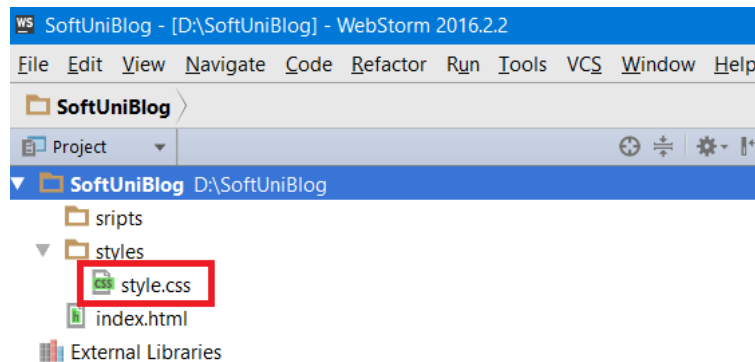
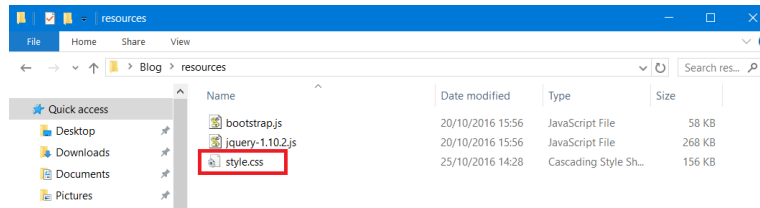
Create **new directory** in our project that will keep our **JS files** and name it **scripts**



5. Add CSS Files to the "styles" Folder

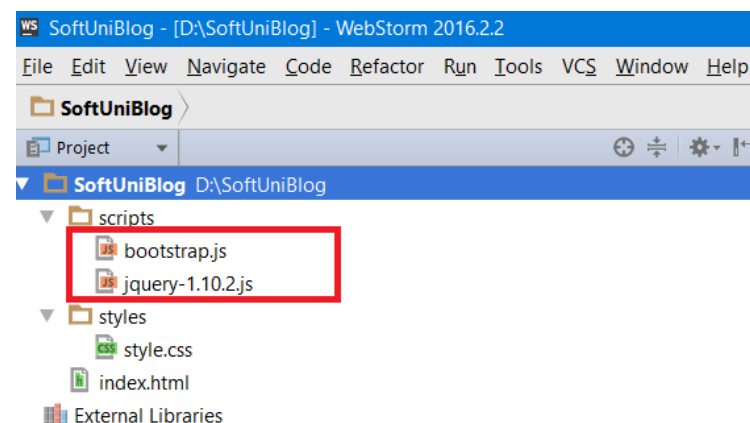
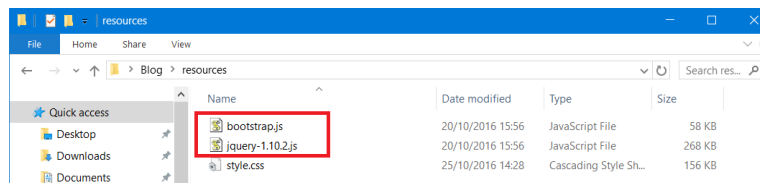
In the **resources folder** for this exercise, you can find some files that you will need for the blog. Head there and find the file "**style.css**". This file is a complete .css file that contains a modified bootstrap theme, meaning that it is basically a plug and play theme, **link your html files to it** and everything will be readily styled.

Just copy "style.css" from the resources folder and paste it in the WebStorm SoftUni Blog styles folder you just created:



6. Add Script Files to the "scripts" Folder

Head to the resources folder again, copy all .js files and paste them in the "scripts" folder:



7. Site Basic Structure

Now it's time to start writing some code. Create the **basic structure of a html** document:

- **<!DOCTYPE>** - this is an instruction to the web browser what version of HTML our page is written in. For HTML5 we should declare to be **<!DOCTYPE html>**

- **<html>** - this tag is the container of all other HTML elements in the document
- **<head>** - contains metadata like the title of the document, styles scripts, meta information and more.
- **<body>** - the actual content displayed in the browser

```
<!DOCTYPE html>
<html lang="en">
  <head>

  </head>
  <body>

  </body>
</html>
```

8. Link HTML and CSS Files

Now, we have the **HTML file** where will be the **structure of the blog** and the **CSS file** that will **make the blog look pretty**. But the **HTML file does not know** that the **CSS file exists**. So, let's link them. You will need to link every page for the website to this css file.

This can be done in the **<head>** part of the HTML file using **<link>** tag with **attributes**:

- **type** = text/css
- **rel** = stylesheet
- **href** = styles/styles.css

```
<head>
  <link type="text/css" rel="stylesheet" href="styles/style.css" />
</head>
```

9. Link HTML and jQuery Script Files

After we **linked the CSS file** we now have to **do the same for the jQuery and Bootstrap script files**. Linking script files is slightly different from linking a CSS file.

For the jQuery, in the **<head>** part of the HTML file using **<script></script>** tag with **attributes**:

- **src** = scripts/jquery-1.10.2.js

```
<head>
  <link type="text/css" rel="stylesheet" href="styles/style.css" />
  <script src="scripts/jquery-1.10.2.js"></script>
</head>
```

10. Link HTML and Bootstrap Script Files

For the bootstrap, in the **<head>** part of the HTML file using **<script></script>** tag with **attributes**:

- **src** = scripts/jquery-1.10.2.js

```

<head>
  <link type="text/css" rel="stylesheet" href="styles/style.css" />
  <script src="scripts/jquery-1.10.2.js"></script>
  <script src="scripts/bootstrap.js"></script>
</head>

```

11. jQuery Before Bootstrap

Quick Note: Make sure that jQuery link is before bootstrap link, otherwise scripts may not work properly.

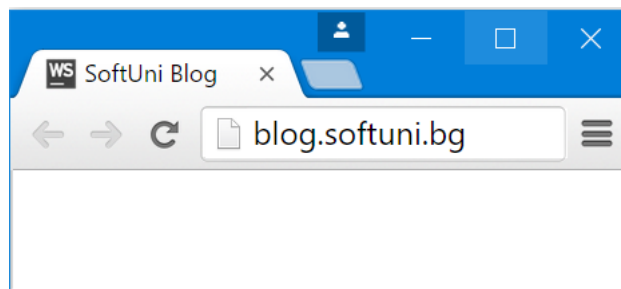
12. Change Blog Title

Let's give our page more **appropriate title** that will be displayed in the browser's tab.

```

<head>
  <title>SoftUni Blog</title>
  <link type="text/css" rel="stylesheet" href="styles/style.css" />
  <script src="scripts/jquery-1.10.2.js"></script>
  <script src="scripts/bootstrap.js"></script>
</head>

```



II. Creating the Navbar



13. Create a Document Structure

Now we are going to start modifying the contents of the body element.

```

1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <title>SoftUni Blog</title>
5      <link type="text/css" rel="stylesheet" href="styles/style.css" />
6      <script src="scripts/jquery-1.10.2.js"></script>
7      <script src="scripts/bootstrap.js"></script>
8    </head>
9    <body>
10     |
11  </body>
12 </html>

```

The **page content is usually located** in the **body** part. However, just putting the content there plainly is not very appropriate. So, we will **wrap the content** in semantic tags which help us **understand the role** of each part of the document.

We will start from the **<header>**, **<main>** and **<footer>** elements. As you can probably guess, the header will hold the header (or the navbar), the main will hold the page content and the footer will contain... you guessed it, the footer:

```
<body>
  <header>
  </header>

  <main>
  </main>

  <footer>
  </footer>
</body>
```

14. Create the Navbar

As our navbar will serve the role of a header, we will put it inside the header tag. First, create a **<div>...</div>** tag with attribute **class="navbar navbar-default navbar-fixed-top text-uppercase"**:

```
<header>
  <div class="navbar navbar-default navbar-fixed-top text-uppercase">

  </div>
</header>
```

The text inside the quotes means that this **<div>** will be a **"navbar"** with **"navbar-default"** theme, of type **"navbar-fixed-top"** so it will stay fixed at the top of the page and will have everything in it with upper case **"text-uppercase"**. This is a part of the **bootstrap syntax**.

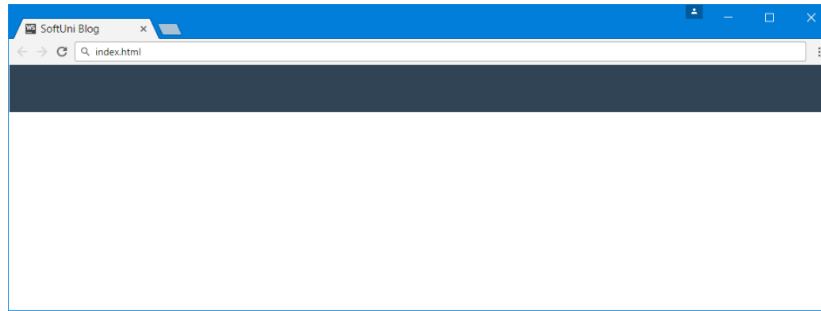
15. Create a Container for The Navbar Elements

In order for the **responsive design from bootstrap to work**, we need to use containers for elements. When the window size changes, basically the container changes its size. **<div>...</div>** container for all navbar elements:

```
<header>
  <div class="navbar navbar-default navbar-fixed-top text-uppercase">
    <div class="container">

    </div>
  </div>
</header>
```

If you start your project now (Default key combination: Ctrl + Shift + F10) you can see the navbar on top of your page:

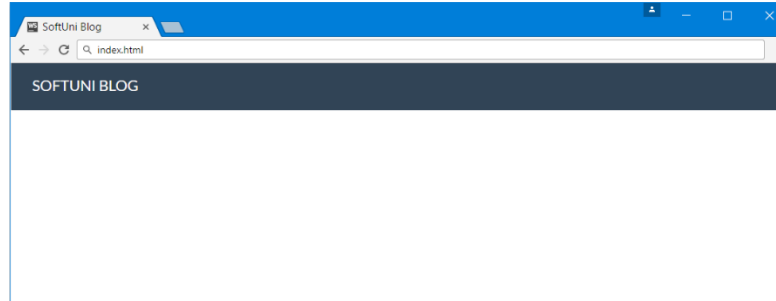


16. Create a Navbar Header Element

Now we need to **insert some content in the navbar**. We will start with the header of the blog, which will also serve the role of a home button. For it, create a `<div>` with class **"navbar-header"** inside the container `<div>`. And inside of the navbar-header `<div>` create a link tag `<a>` with text "SoftUni Blog" and a **href="index.html"** and a class **"navbar-brand"**:

```
<div class="navbar navbar-default navbar-fixed-top text-uppercase">
  <div class="container">
    <div class="navbar-header">
      <a href="index.html" class="navbar-brand">SoftUni Blog</a>
    </div>
  </div>
</div>
```

The **href** basically means that this link will point to the page we are currently building. You can check that what we just included appears in the header now:



If you click the link it should redirect you to the same page.

17. Create Navbar Buttons

Now we need to fill the rest of the navbar. On the home page, we will have **two buttons** on the right of the navbar - **Register** and **Login**. You should place them in the container we made earlier:

```
<div class="container">
```

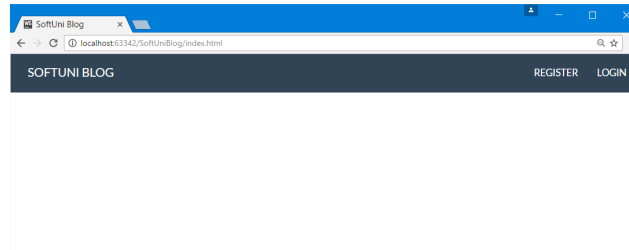
We should place them in a separate `<div>` which will have a class "navbar-collapse collapse". In the `<div>` we will place an unordered list `` with a class **"nav navbar-nav navbar-right"** and two `` elements - Register and Login. Each of them will be inside a link tag `<a>`. The **href=""** attribute should be left to **href="#"**, which means that it is a placeholder for an actual link. It will just do nothing for now:


```

<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav navbar-right">
    <li><a href="#">Register</a></li>
    <li><a href="#">Login</a></li>
  </ul>
</div>

```

Test if the buttons appear in the navbar:



18. Create a Responsive Dropdown Menu

The responsive dropdown is actually why we included both jQuery.js and bootstrap.js files, that is to make the navbar buttons collapse if the window become too small.

For this we will add a **<button>** to the navbar-header **<div>** we made earlier. Just after the home button, add a new **<button>** attributes:

- **type="button"**
- **class="navbar-toggle"**
- **data-toggle="collapse"**
- **data-target=".navbar-collapse"**

```

<div class="navbar-header">
  <a href="index.html" class="navbar-brand">SoftUni Blog</a>

  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
    </button>
</div>

```

Now add three **** with class **"icon-bar"**. The code should look like this:

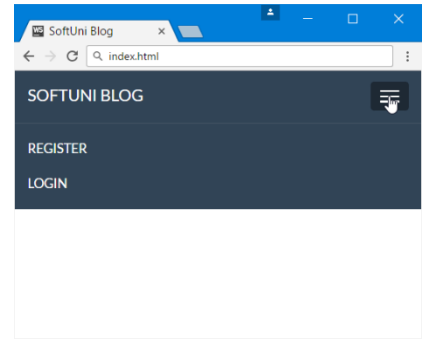
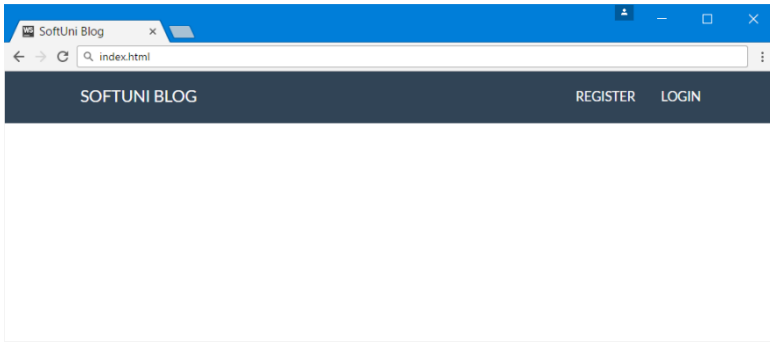
```

<div class="navbar-header">
  <a href="index.html" class="navbar-brand">SoftUni Blog</a>

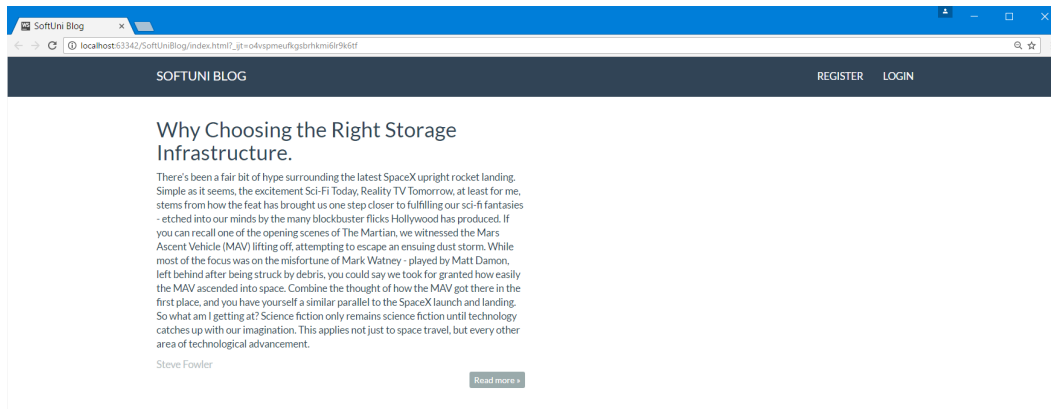
  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
  </button>
</div>

```

And ... we are ready with the navbar! **Test everything** to make sure it looks and works as intended. Shrink the window to see if it "responds", expand the menu, etc.:



III. Creating the Main Content



19. Wrap the Content in The Main

We are now going to work in the **<main>** part:

```
</header>

<main>
</main>

<footer>
</footer>
</body>
```

Again, we will need a **container** for the content there so we will create one. We can use a **<div>** with class "container body-content". Inside of it we will create a **<div>** with class "row" so our content is properly "responsive" (e.g. bootstrap reasons):

```
<main>
  <div class="container body-content">
    <div class="row">
    </div>
  </div>
</main>
```

20. Create a Wrapper for a Column

Working with columns inside a row will make sure that everything is **displayed properly** when **resizing the window** (you don't have to, but you can quickly glance through [this](#) article). So, we will make a **<div>** with **class="col-md-6"**, which basically means that we are creating a column that will have width 6 (bootstrap works with base width 12, as it is easily divisibly to even number of columns like 2, 3, 4, 6 ... etc.). Our home page will show two columns with articles:

```
<div class="container body-content">
  <div class="row">
    <div class="col-md-6">
    </div>
  </div>
</div>
```

Inside of it will be a single article.

21. Creating Structure for a Single Article

Inside the column we have just created, we are going to create the **template** which **every article** on the home page will use. For this, we are going to use the semantic tag `<article>`, and inside of it every article will have a `<header>`, `<p>`, author and `<footer>`:

```
<div class="col-md-6">
  <article>
    <header>
    </header>

    <p>
    </p>

    <small class="author">
    </small>

    <footer>
    </footer>
  </article>
</div>
```

22. Creating Article Footer

Within the footer, we are going to have a button that links to the whole article. Inside a `<div>` create a link `<a>` with the following attributes:

- `class="btn btn-default btn-xs"`
- `href="#"`

Make the `<div>` **"pull-right"** and put text inside the button **"Read more »"**:

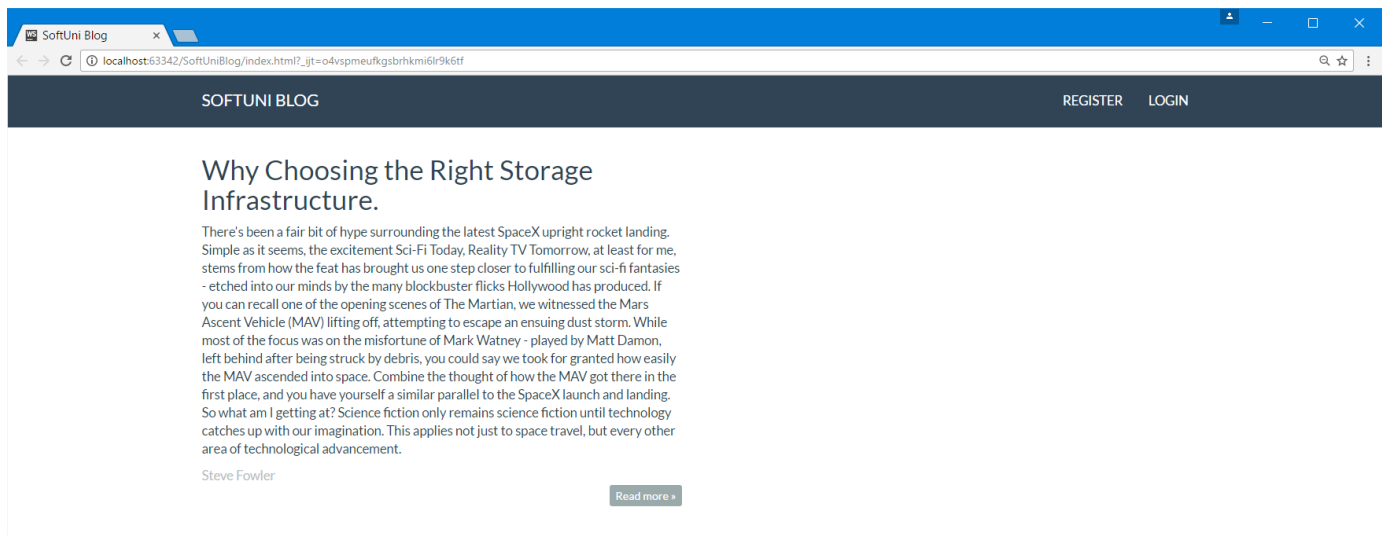
```
<footer>
  <div class="pull-right">
    <a class="btn btn-default btn-xs" href="#">Read more &raquo;</a>
  </div>
</footer>
```

23. Fill Article Content and Test

Fill the `<header>` with a sample header inside a `<h2>` tag and some sample text:

```
<header>
  <h2>Why Choosing the Right Storage Infrastructure.</h2>
</header>
```

Then fill the `<p>` tag, the author and start your project. It should look like this:



IV. Creating the Footer

24. Create the Footer

We are now working here:

```
    </main>
  <footer>
</footer>
</body>
</html>
```

This will be easy. The footer of the page needs only a single paragraph `<p>` and it will be placed in the `<footer>` tag inside a `<div class="container modal-footer">` in it, containing the following text "**©** 2016 - Software University Foundation":

```
<footer>
  <div class="container modal-footer">
    <p>&copy; 2016 - Software University Foundation</p>
  </div>
</footer>
```

And now you are done with the main page! If you want, you can multiply the articles (copy and paste the whole `<div class="col-md-6">`), just to see how the blog will look once it has some content in it:

Why Choosing the Right Storage Infrastructure.

There's been a fair bit of hype surrounding the latest SpaceX upright rocket landing. Simple as it seems, the excitement Sci-Fi Today, Reality TV Tomorrow, at least for me, stems from how the feat has brought us one step closer to fulfilling our sci-fi fantasies - etched into our minds by the many blockbuster flicks Hollywood has produced. If you can recall one of the opening scenes of The Martian, we witnessed the Mars Ascent Vehicle (MAV) lifting off, attempting to escape an ensuing dust storm. While most of the focus was on the misfortune of Mark Watney - played by Matt Damon, left behind after being struck by debris, you could say we took for granted how easily the MAV ascended into space. Combine the thought of how the MAV got there in the first place, and you have yourself a similar parallel to the SpaceX launch and landing. So what am I getting at? This applies not just to space travel, but every other area of technological advancement.

Steve Fowler

[Read more »](#)

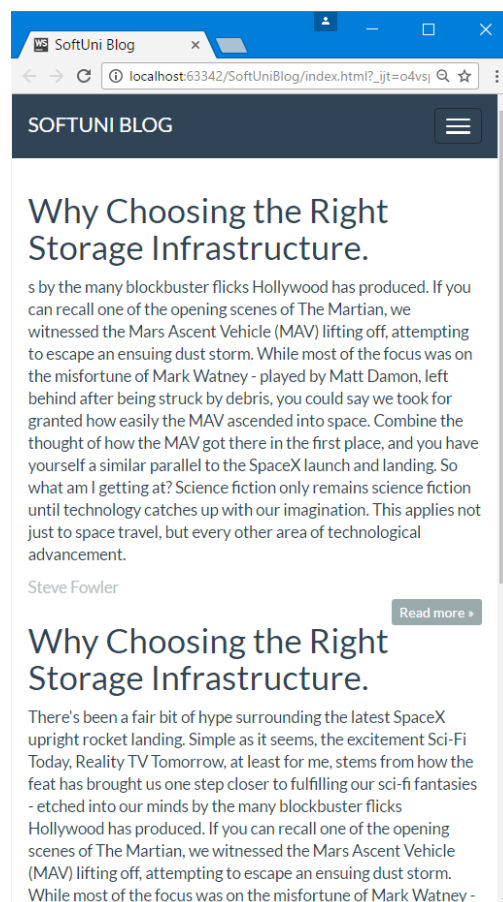
Infrastructure Right Why the Storage Choosing.

There's been a fair bit of hype surrounding the latest SpaceX upright rocket landing, you could say we took for granted how easily the MAV ascended into space. Combine the thought of how the MAV got there in the first place, and you have yourself a similar parallel to the SpaceX launch and landing. So what am I getting at? This applies not just to space travel, but every other area of technological advancement.

Fowler Steve

[Read more »](#)

© 2016 - Software University Foundation

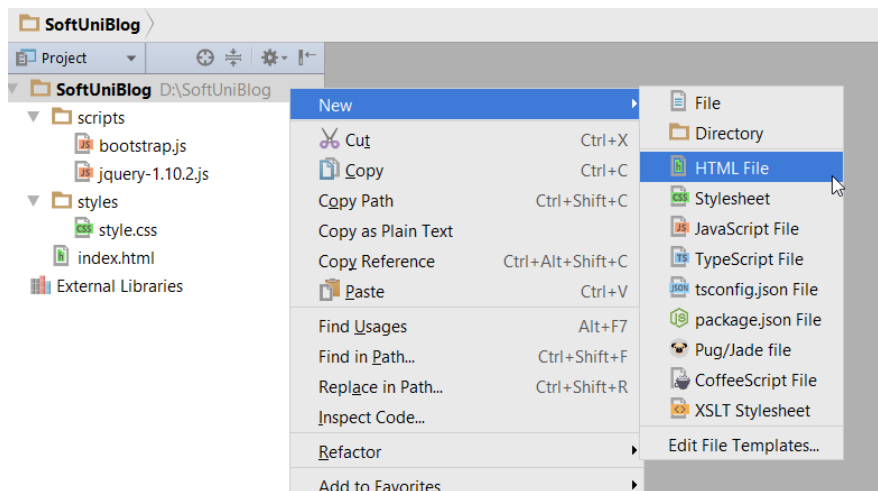


V. Creating the Logged in View

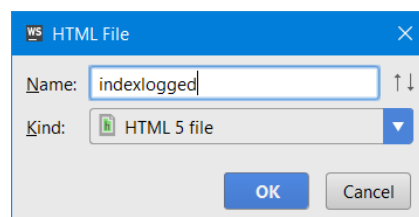
25. Initial Setup

We need a **variation of the home page** in which **the header is slightly different**. It is logical, when you are not logged in the site, to have a button for just that, and contrary, when you are logged in, you need to have a **button for creating a new post** and a **button for logging out**. This is what we are going to do now.

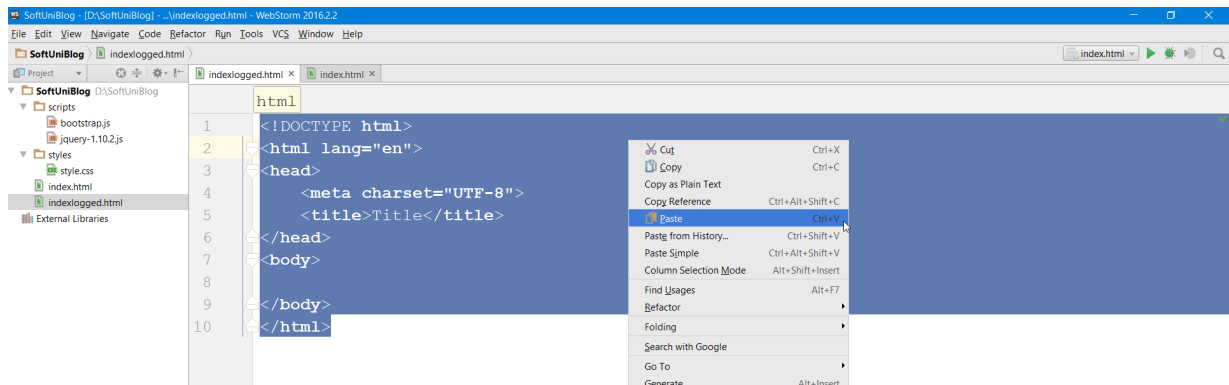
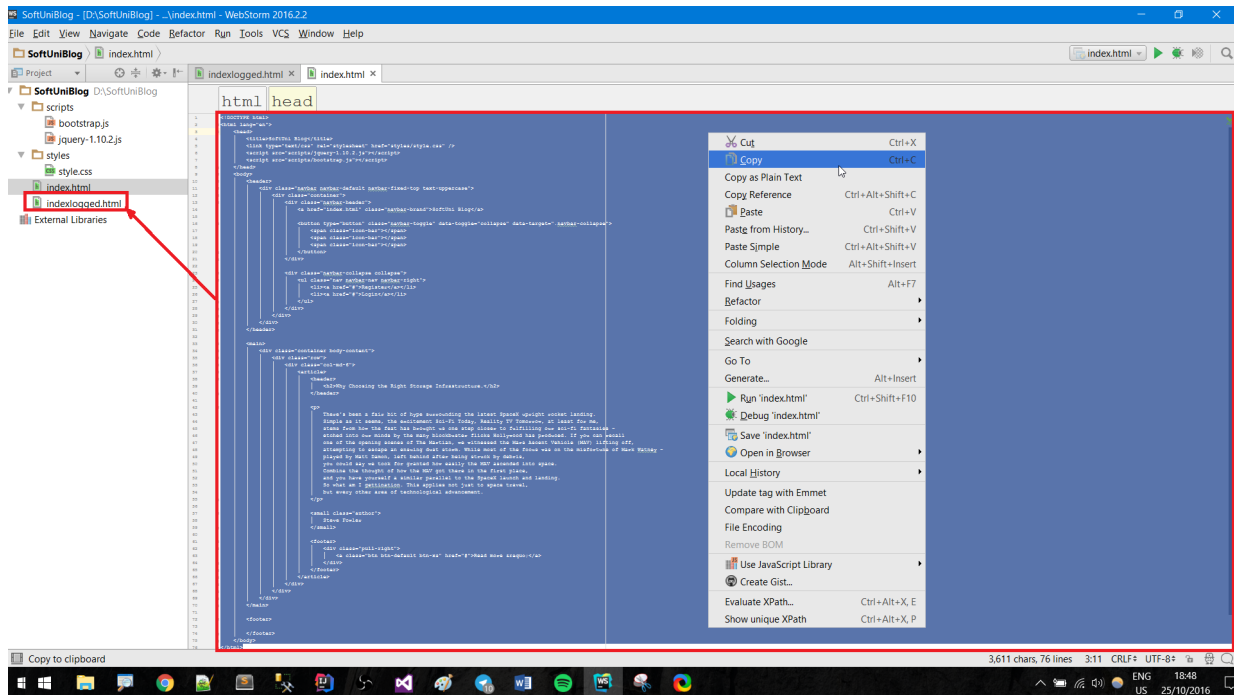
First you will need a different page, so create one:



Call it "indexlogged":



And just copy the whole HTML from the index.html page and paste it in indexlogged.html, overriding all code that WebStorm generates:



Now you should have **two pages that are exactly the same except the name**. This is convenient, because everything is the same, except for two buttons.

26. Edit indexlogged.html

Now we need to **edit the new page "indexlogged.html"**. You should look for the navbar buttons in the header:


```

</header>
<div class="navbar navbar-default navbar-fixed-top text-uppercase">
  <div class="container">
    <div class="navbar-header">
      <a href="index.html" class="navbar-brand">SoftUni Blog</a>

      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
    </div>

    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#">Register</a></li>
        <li><a href="#">Login</a></li>
      </ul>
    </div>
  </div>
</div>
</header>

```

We need to add **one more button** that will be called "Welcome(User)". It will be placed before the other two. For now, "User" is just a placeholder that will be replaced by the actual username in the future. The button will be exactly like the other two, a list item `...` with a link inside `...<a>` and text in it "Welcome(User)":

```

<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav navbar-right">
    <li><a href="#">Welcome (User) </a></li>
    <li><a href="#">Register</a></li>
    <li><a href="#">Login</a></li>
  </ul>
</div>

```

Now just edit the text in the other two links: edit **"Register" into "New Post"**, and **"Login" into "Logout"**:

```

<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav navbar-right">
    <li><a href="#">Welcome (User) </a></li>
    <li><a href="#">New Post</a></li>
    <li><a href="#">Logout</a></li>
  </ul>
</div>

```

The new page's navbar should look like this now:



Why Choosing the Right Storage

VI. Creating the View Post Page

This is a page for viewing a single post. The "read more" button should redirect to this page.

Why Choosing the Right Storage Infrastructure.

There's been a fair bit of hype surrounding the latest SpaceX upright rocket landing. Simple as it seems, the excitement Sci-Fi Today, Reality TV Tomorrow, at least for me, stems from how the feat has brought us one step closer to fulfilling our sci-fi fantasies - etched into our minds by the many blockbuster flicks Hollywood has produced. If you can recall one of the opening scenes of The Martian, we witnessed the Mars Ascent Vehicle (MAV) lifting off, attempting to escape an ensuing dust storm. While most of the focus was on the misfortune of Mark Watney - played by Matt Damon, left behind after being struck by debris, you could say we took for granted how easily the MAV ascended into space. Combine the thought of how the MAV got there in the first place, and you have yourself a similar parallel to the SpaceX launch and landing. So what am I getting at. This applies not just to space travel, but every other area of technological advancement.

Steve Fowler

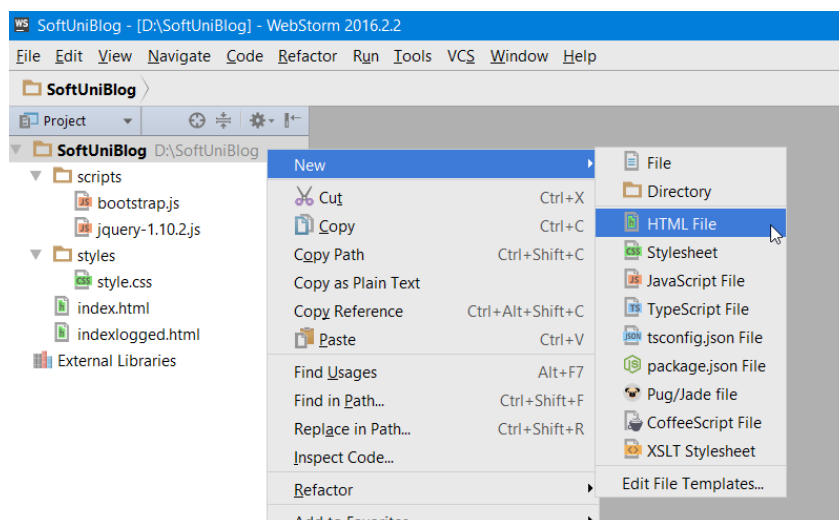
back »

© 2016 - Software University Foundation

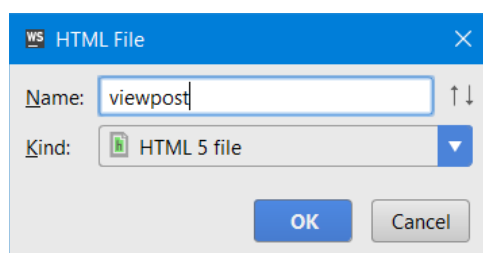
27. Initial Setup

This one is easy. You can just copy and paste the index.html or indexlogged.html and just modify the code a little bit.

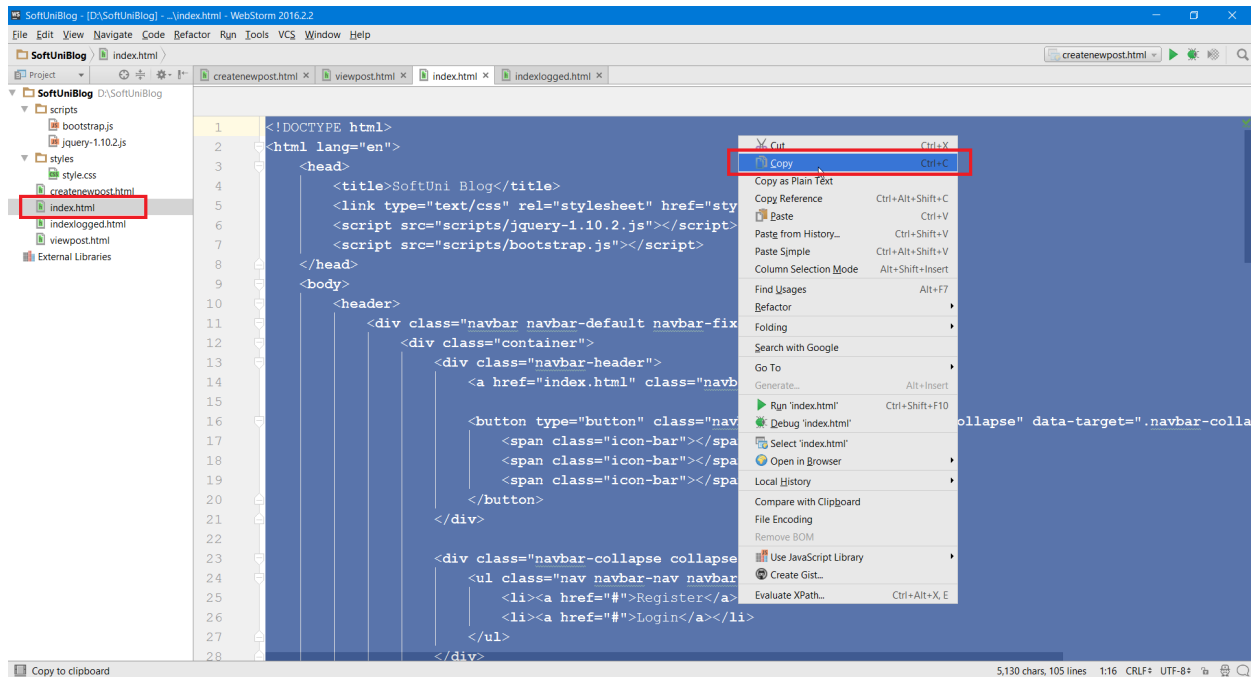
First create a new HTML file:



Name it "viewpost":



Then copy and paste the whole index.html into the new file, overwriting everything in the new file:



Now in newpost.html leave only one article (that is the content of a single **<div class="col-md-6">**) and modify these:

- Make the class of the **<div>** wrapper "**col-md-12**". This will make the article to be **presented on the whole page**
- **Replace the text** in the button from "Read more" to "back"

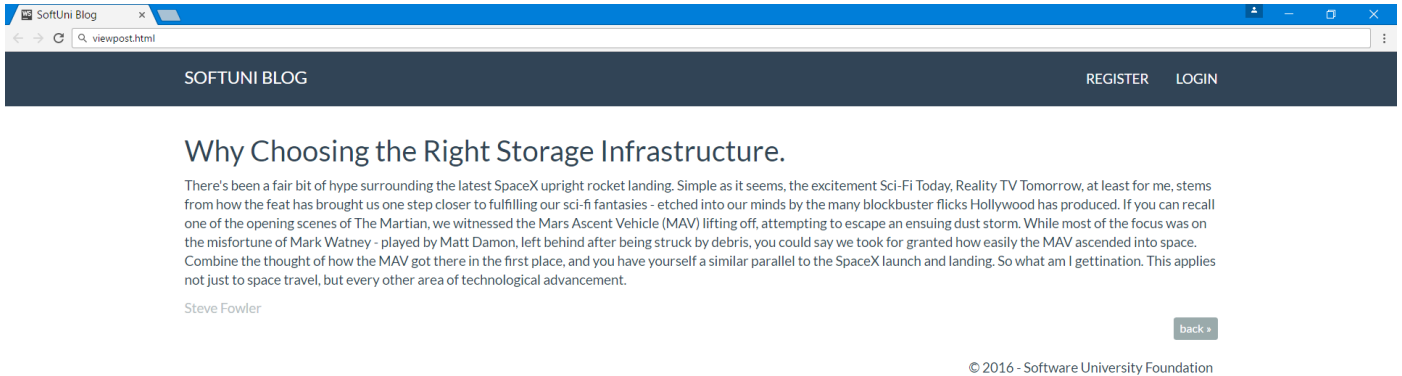
```
<div class="col-md-12">
  <article>
    <header>
      <h2></h2>
    </header>

    <p>
    </p>

    <small class="author">
    </small>

    <footer>
      <div class="pull-right">
        <a class="btn btn-default btn-xs" href="#">back &raquo;</a>
      </div>
    </footer>
  </article>
</div>
```

Of course, in the **<header>** and **<p>** elements you should have your own article text, as well as in the "author" element. So... that was it, you now have a view single post page:

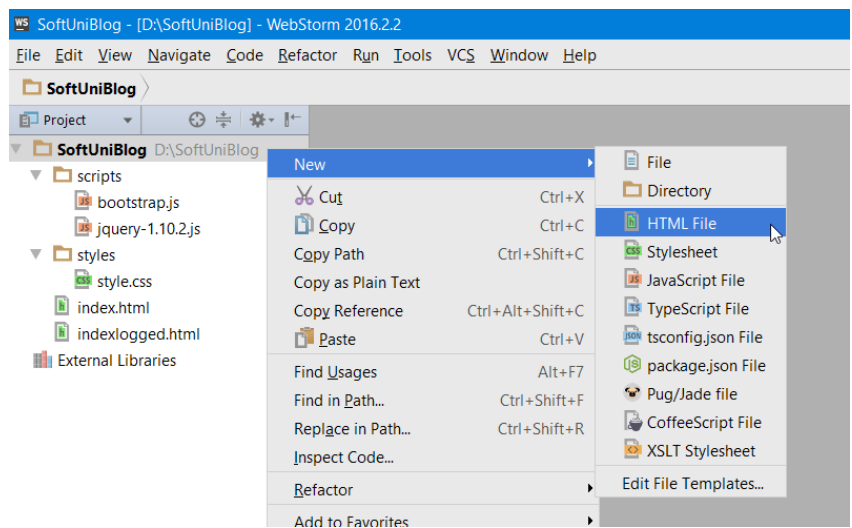


VII. Creating the New Post Page

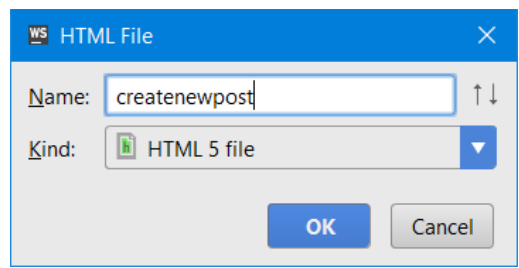
28. Initial Setup

If you are feeling adventurous, you can rewrite the navbar that we've already made. But for the sake of saving space and time, we are going to just copy and paste it.

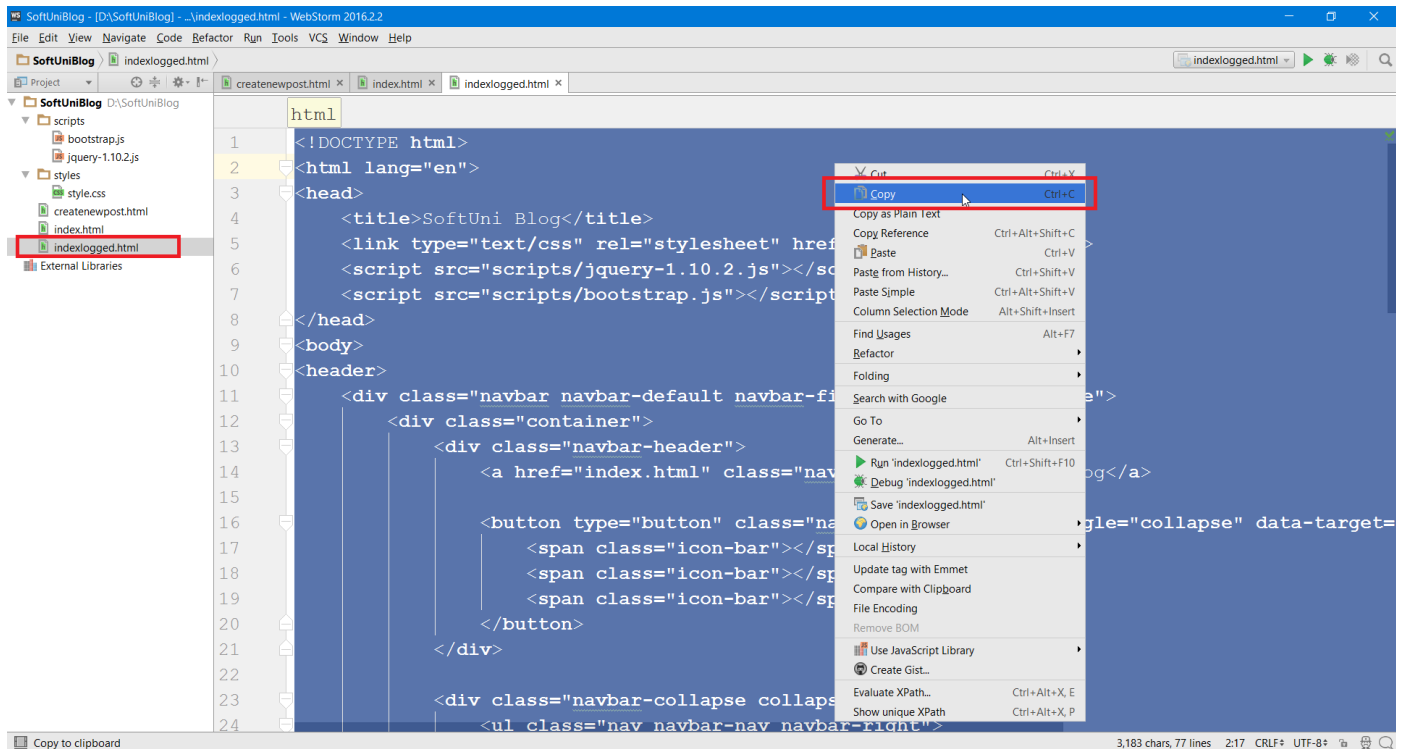
First create a new HTML file:



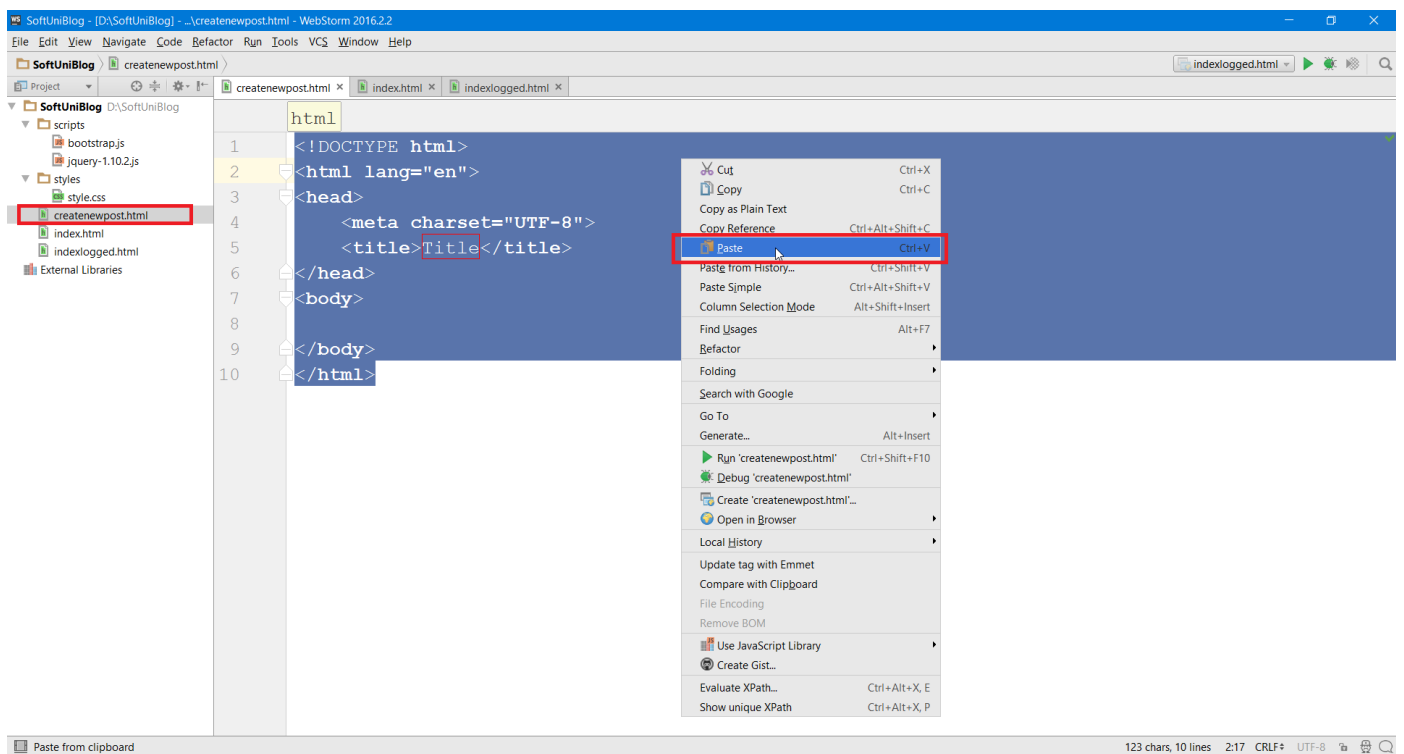
Name it "createnewpost":



Again, copy and paste everything, this time from **indexlogged.html** into **createnewpost.html**:



Overwrite everything inside createnewpost.html:



29. Delete All Content, Except the Navbar

You can now just delete the segment storing all posts, that is everything inside the **<main>** tag:

```

</header>

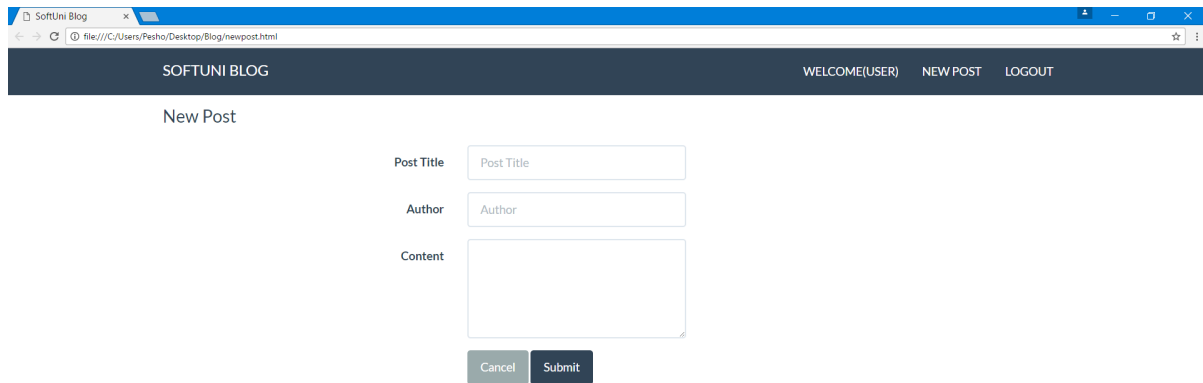
<main>
</main>

<footer>

```

30. Creating the New Post Content

For a **new post to be created** we need **three text boxes** and a **pair of buttons**. This should give you an idea of what will the page look like:



First of all, head to the `<main>` and insert a `<div>` with a class **"container body-content span=8 offset=2"**. This will be the **container for all elements**. Span = 8 means that it will occupy 8 columns of 12 total (remember that bootstrap works with 12 column system) and offset=2 means that it will start after 2 columns on the left (so 2 on the left, 8 in the middle and 2 on the right, for total of 12).

Here is the element:

```
<main>
  <div class="container body-content span=8 offset=2">
  </div>
</main>
```

Inside of it, we need a `<form>` tag with class **"form-horizontal"** and a nested `<fieldset>` tag:

```
<main>
  <div class="container body-content span=8 offset=2">
    <form class="form-horizontal">
      <fieldset>
      </fieldset>
    </form>
  </div>
</main>
```

Now we can insert a `<legend>`, which is the "New Post" title in the upper left corner:

```
<fieldset>
  <legend>New Post</legend>
</fieldset>
```

31. Text Box Template

Now we **need a template for every text box**. Every text box needs a **container**, a in it label and the actual text box.

We will start with the container. This will be a `<div>` with a **class="form-group"** that will group a form element:

```

<fieldset>
  <legend>New Post</legend>

  <div class="form-group">
  </div>

</fieldset>

```

Place the **<label>** in it with **class="col-sm-4 control-label"** and text **"Post Title"**. The **"col-sm-4"** is part of bootstrap. It means that when the window is small (sm), the column (col) will have width 4 of 12:

```

<div class="form-group">
  <label class="col-sm-4 control-label">Post Title</label>
</div>

```

By far you should see this:



Now create one more **<div>** after the **<label>** with a **class="col-sm-4"** and in it create an **<input>** with the following attributes:

- **type="text"**
- **class="form-control"**
- **id="postTitle"**
- **placeholder="Post Title"**

```

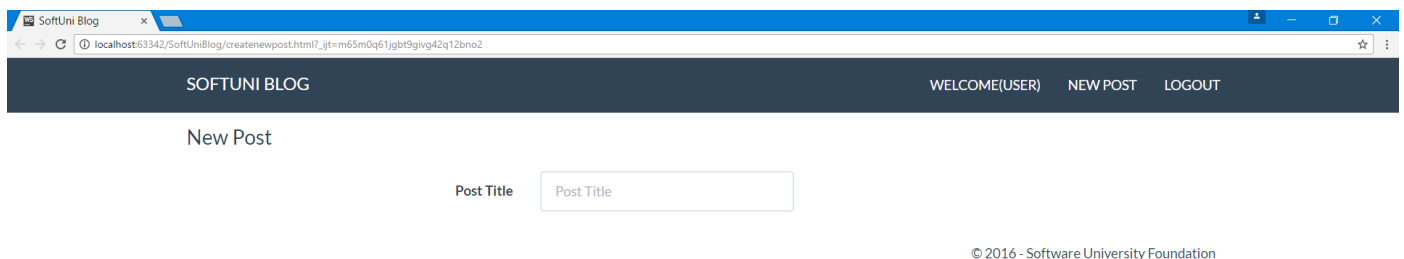
<fieldset>
  <legend>New Post</legend>

  <div class="form-group">
    <label class="col-sm-4 control-label">Post Title</label>
    <div class="col-sm-4 ">
      <input type="text" class="form-control" id="postTitle" placeholder="Post Title">
    </div>
  </div>

</fieldset>

```

And now we have a text box:



The left "Post Title" is the **<label>** and the "Post Title" inside the text box is the **placeholder="Post Title"**.

32. Multiply Text Boxes

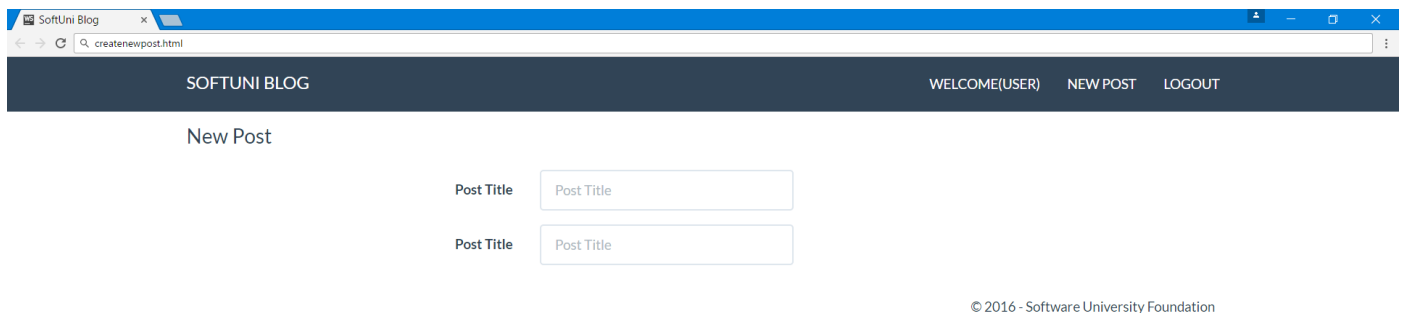
You can now copy the code for a single text box (the whole `<div class="form-group">`) and paste it one more time:

```
<fieldset>
  <legend>New Post</legend>

  <div class="form-group">
    <label class="col-sm-4 control-label">Post Title</label>
    <div class="col-sm-4 ">
      <input type="text" class="form-control" id="postTitle" placeholder="Post Title">
    </div>
  </div>

  <div class="form-group">
    <label class="col-sm-4 control-label">Post Title</label>
    <div class="col-sm-4 ">
      <input type="text" class="form-control" id="postTitle" placeholder="Post Title">
    </div>
  </div>
</fieldset>
```

This should create one more text box with the same properties:



You just need to **rename everything appropriately**. The label for the second box would be "Author", the placeholder "Author" and the id - "postAuthor":

```
<div class="form-group">
  <label class="col-sm-4 control-label">Post Title</label>
  <div class="col-sm-4 ">
    <input type="text" class="form-control" id="postTitle" placeholder="Post Title">
  </div>
</div>

<div class="form-group">
  <label class="col-sm-4 control-label">Author</label>
  <div class="col-sm-4 ">
    <input type="text" class="form-control" id="postAuthor" placeholder="Author">
  </div>
</div>
```

33. Content Text Box

Now the content box is a little bit different. For the `<div>`, instead of `class="col-sm-4"`, you should have `class="col-sm-6"` and `<input>` you need a `<textarea>` with the following attributes:

- `class="form-control"`
- `rows="5"`
- `id="postContent"`


```

<div class="form-group">
  <label class="col-sm-4 control-label">Content</label>
  <div class="col-sm-6">
    <textarea class="form-control" rows="5" id="postContent"></textarea>
  </div>
</div>

```

This will create the content text area:

34. Submit Buttons

The only thing left on this page is the submit and cancel buttons.

You need one more **<div>** with **class="form-group"**. Inside of it place a **<div>** with **class="col-sm-4 col-sm-offset-4"**. And inside of it place the two buttons:

```

<div class="form-group">
  <div class="col-sm-4 col-sm-offset-4">
    <!-- button -->
    <!-- button -->
  </div>
</div>

```

The first **<button>** will be of **type="reset"** and **class="btn btn-default"**:

```
<button type="reset" class="btn btn-default">Cancel</button>
```

The second **<button>** will be of **type="submit"** and **class="btn btn-primary"**:

```
<button type="submit" class="btn btn-primary">Submit</button>
```

We are done! We have a create new post page:

© 2016 - Software University Foundation

VIII. Register Page

We are going to build the following:

SOFTUNI BLOG REGISTER LOGIN

Register

Email

Full Name

Password

Confirm Password

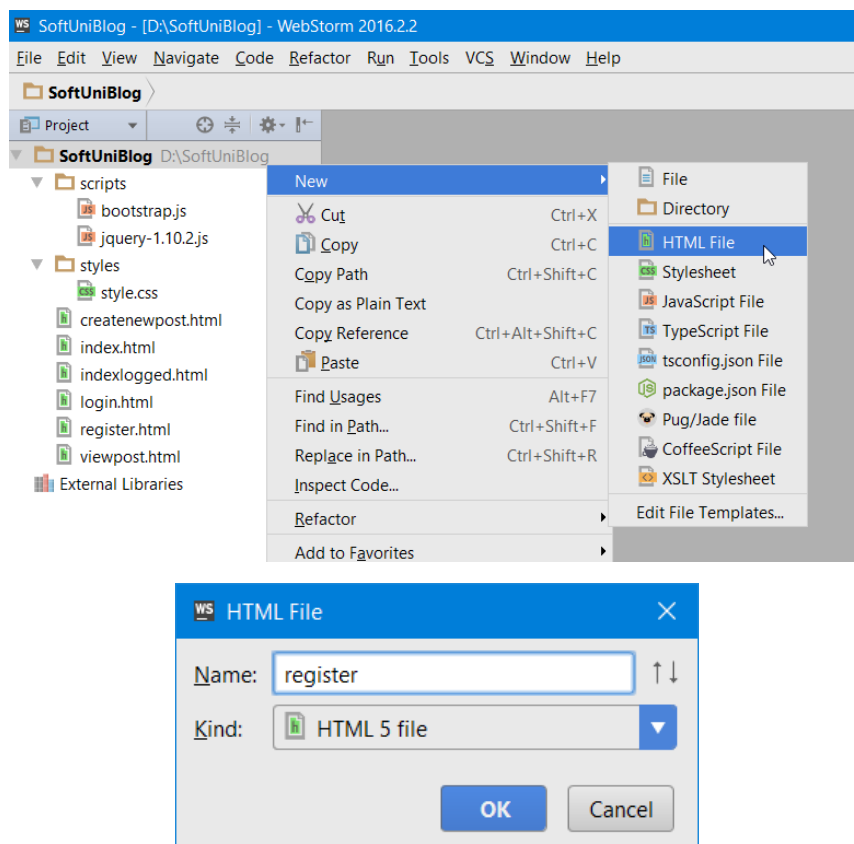
© 2016 - Software University Foundation

By now you should know everything you need to create the page above. You already have used every element that is present in this view so if you feel confident, you can build it yourself. If not, continue reading.

35. Initial Setup

If you couldn't have guessed it... well yeah, we are going to copy-paste again.

Create a new page "register.html":



Copy the code from index.html so you will have all you need right from the start.

Then delete everything inside the **<main>** tag and start working there:

```
    </div>
  </div>
</header>

<main>
</main>

<footer>
  <div class="container modal-footer">
    <p>&copy; 2016 - Software University Foundation</p>
  </div>
</footer>
```

36. Container for Content

In the **<main>**, insert a **<div>** with a class **"container body-content span=8 offset=2"**. This will be the container for all textbox elements:

```
<main>
  <div class="container body-content span=8 offset=2">
  </div>
</main>
```

Inside of it, we need a **<form>** tag with class **"form-horizontal"** and a nested **<fieldset>** tag:

```
<main>
  <div class="container body-content span=8 offset=2">
    <form class="form-horizontal">
      <fieldset>
      </fieldset>
    </form>
  </div>
</main>
```

Now we can insert a **<legend>**, which will be the "Register" title in the upper left corner:

```
<main>
  <div class="container body-content span=8 offset=2">
    <form class="form-horizontal">
      <fieldset>
        <legend>Register</legend>
      </fieldset>
    </form>
  </div>
</main>
```

37. Text Box Template

We need a template for every text box. Every text box needs a container, and in it - a label and the actual text box.

We will start with the container. This will be a **<div>** with a **class="form-group"** that will group a form element:

```

<fieldset>
  <legend>Register</legend>

  <div class="form-group">
    </div>

</fieldset>

```

Place the **<label>** in it with **class="col-sm-4 control-label"** and text **"Email"**:

```

<div class="form-group">
  <label class="col-sm-4 control-label">Email</label>
</div>

```

Now create one more **<div>** after the **<label>** with a **class="col-sm-4"** and in it create an **<input>** with the following attributes:

- **type="text"**
- **class="form-control"**
- **id="inputEmail"**
- **placeholder="Email"**

```

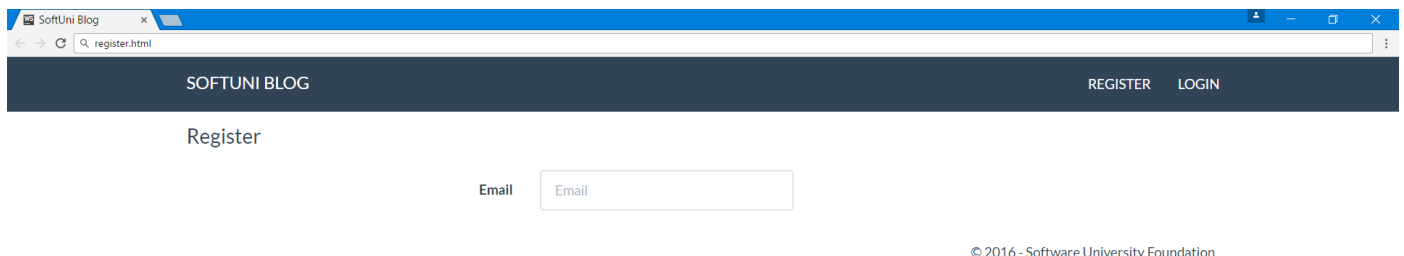
<fieldset>
  <legend>Register</legend>

  <div class="form-group">
    <label class="col-sm-4 control-label">Email</label>
    <div class="col-sm-4 ">
      <input type="text" class="form-control" id="inputEmail" placeholder="Email">
    </div>
  </div>

</fieldset>

```

And now we have a text box:



38. Multiply Text Boxes

You can now **copy the code for a single text box** (the whole **<div class="form-group">**) and paste it **three more times**:

```

<fieldset>
  <legend>Register</legend>

  <div class="form-group">
    <label class="col-sm-4 control-label">Email</label>
    <div class="col-sm-4 ">
      <input type="text" class="form-control" id="inputEmail" placeholder="Email">
    </div>
  </div>

  <!--insert copies here-->

</fieldset>

```

This should create **three more text boxes** with the same properties:

You just need to **rename everything appropriately**:

- The **label** for the **second box** would be "Full Name", the **placeholder** "Full Name" and the **id** - "inputFullName"
- The **label** for the **third box** would be "Password", the **placeholder** "Password" and the **id** - "inputPassword"
- The **label** for the **fourth box** would be "Confirm Password", the **placeholder** "Confirm Password" and the **id** - "inputConfirmPassword"

39. Submit Buttons

The only thing left on this page is the **register** and **cancel buttons**.

You need one more `<div>` with **class="form-group"**. Inside of it place a `<div>` with **class="col-sm-4 col-sm-offset-4"**. And inside of it place the two buttons:

```
<div class="form-group">
  <div class="col-sm-4 col-sm-offset-4">
    <!-- button -->
    <!-- button -->
  </div>
</div>
```

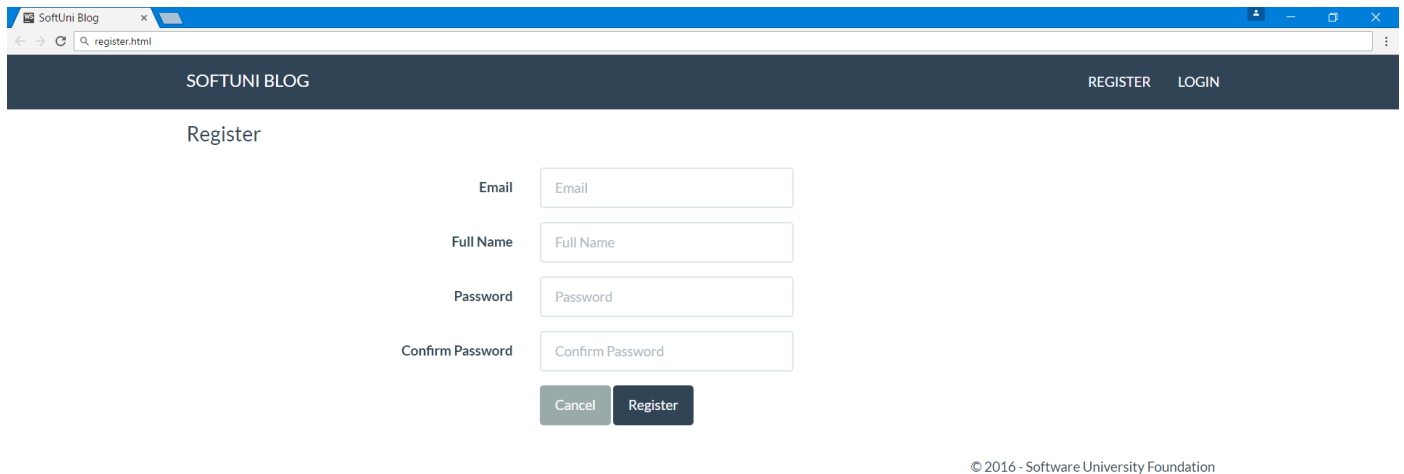
The first `<button>` will be of **type="reset"** and **class=" btn btn-default"**:

```
<button type="reset" class="btn btn-default">Cancel</button>
```

The second `<button>` will be of **type="submit"** and **class=" btn btn-primary"**:

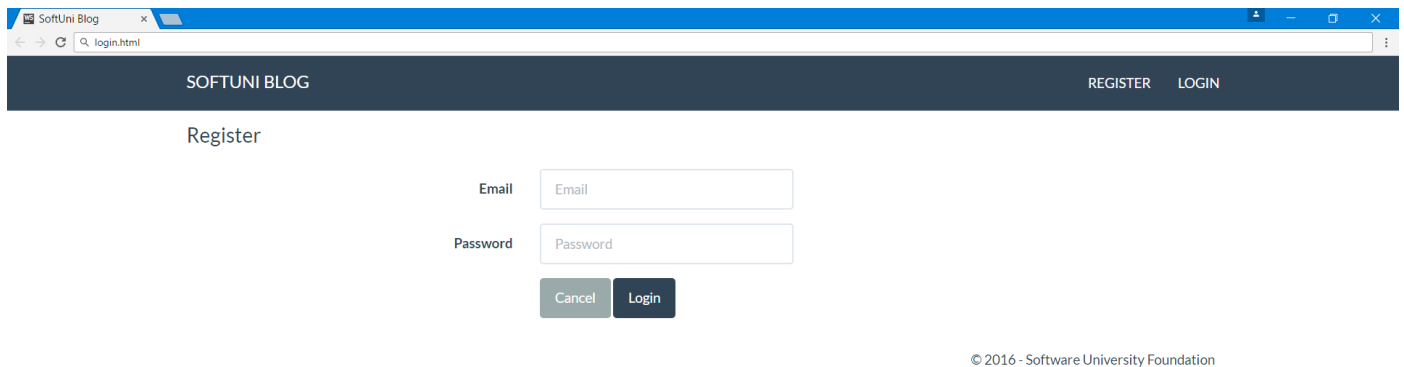
```
<button type="submit" class="btn btn-primary">Register</button>
```

We are done! We have a create the register page:



IX. Login Page

This is the **final page** that you are going to create:



As it is practically the **same as the register page** and is even easier, this job will be solely for you to finish.

Hints

- Use the Register page and just delete the two textboxes that you don't need
- Rename the Register button to "Login"

X. Link Pages

If you want, you can play a little bit with the static web site you have just created. You can **modify every link** in it to **point to another page** and this will make your site feel real, even if it doesn't yet have the functionality to actually store user info and blog posts.

You can **modify the links** in the `<a>` tags. For example, in the file `index.html`, you can set the **href=""** of the register button to point to the `register.html` page and the **href=""** of the login button to point to the `login.html` page:

```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav navbar-right">
    <li><a href="register.html">Register</a></li>
    <li><a href="login.html">Login</a></li>
  </ul>
</div>
```

This way if you **click on** the login or register button you will be **directed** to the corresponding page.

You can link the other pages as well or even create links in the register and cancel buttons, but this is **yours to experiment with**.

Hints

- Link "SoftUni Blog" button to direct to index.html
- Link "Logout" button to direct to index.html
- Link "Read More" button to direct to viewpost.html
- Link "New Post" button to direct to createnewpost.html
- Link "Back" button to direct to index.html

XI. Congratulations!

Congratulations, you have **completed** the exercise for **HTML5** and **CSS**!