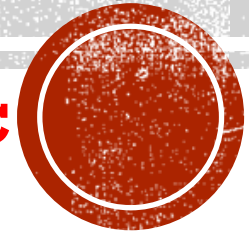
ASP.NET CORE MVC
SELECT TOPICS FROM MODULES 09-10
BOOTSTRAP, UNIT TESTING, TDD, MULTIPLE ENVIRONMENTS, LOGGING



Summer 2021 – Web Development using ASP .Net Core MVC

MAIN SOURCES FOR THESE SLIDED

- Unless otherwise specified, the main sources for these slides are:
  - https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications ←for homework
  - https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0 ←for "textbook"

INTRODUCTION TO BOOTSTRAP

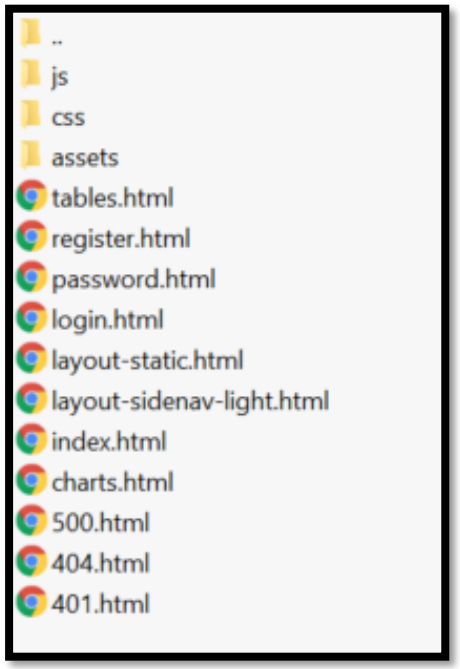
- Sources:
  - see ASP .Net Core documentation
  - https://getbootstrap.com/
  - https://www.w3schools.com/bootstrap4/bootstrap_ref_all_classes.asp
  - https://www.w3schools.com/bootstrap4/default.asp

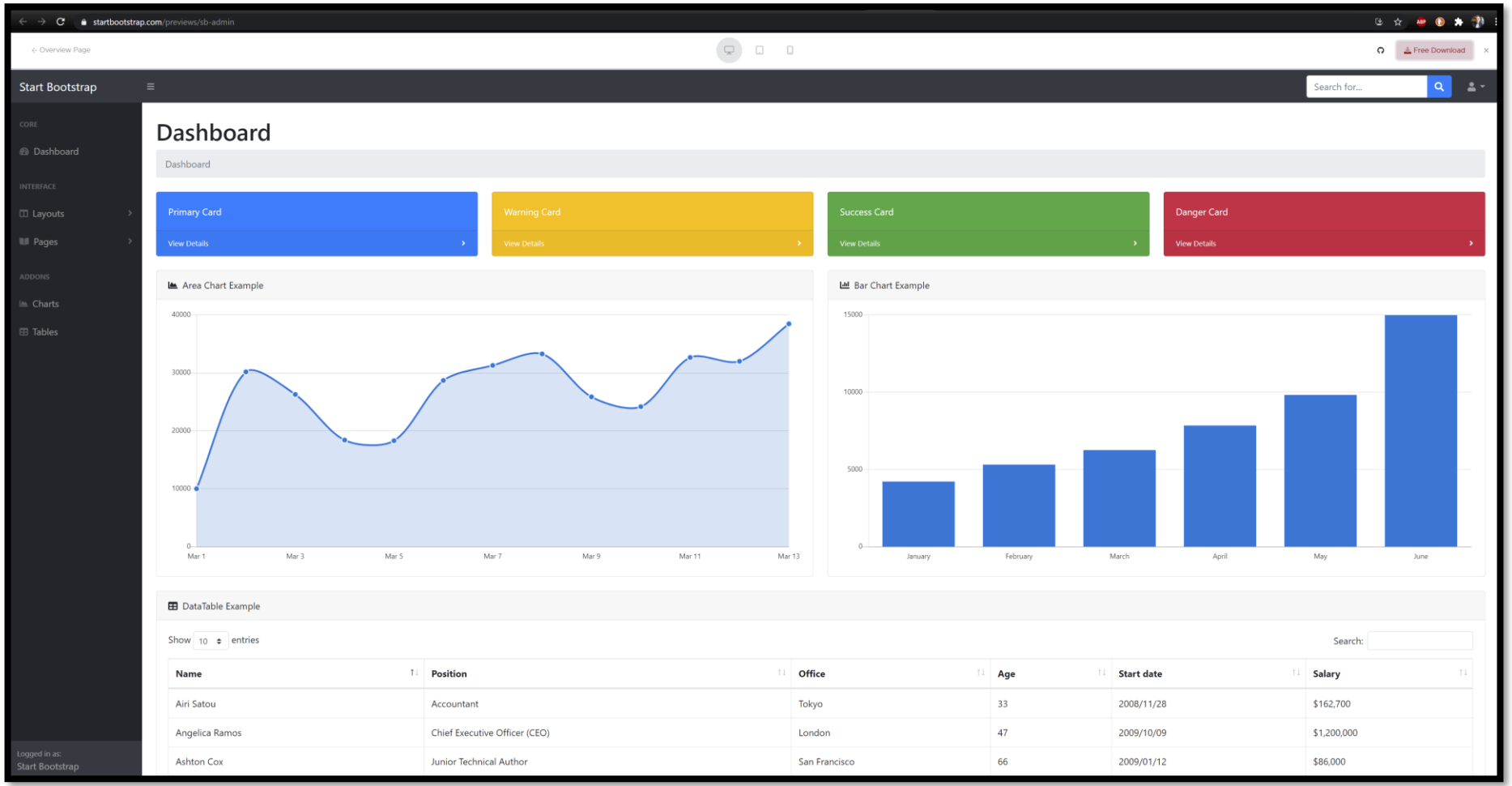
- Bootstrap is a free open-source front-end framework for building responsive web applications
  - It includes pre-built components and styles such as: forms, buttons, tables, navigations, modals, …
  - You'll get responsive pages … check out the following example … resize the width of the page to see how the page responds to this change

- To use it, you'll need to add the following to your project:
  - bootstrap.css: the **CSS** needed for Bootstrap needs to work:
    - `<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">`
    - Or install it in your own project (use **npm …**)
  - bootstrap.js: the **JavaScript** that Bootstrap needs to work:
    - `<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>`
    - Or install it in your own project (use **npm …**)
  - jQuery: Boostrap is dependent on **jQuery**, so you must also include this
    - `<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>`
    - Or install it in your own project (use **npm …**)
  - popper.js: needed for HTML elements that pop out such as **tooltips**
    - `<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>`
    - Or install it in your own project (use **npm …**)

EXISTING BOOTSTRAP THEMES

- To get a sense of what you can get with bootstrap, have a look at some of the themes
  - https://startbootstrap.com/ ← free and for pay
  - https://themes.getbootstrap.com/ ← for pay
- Here is a free theme: https://startbootstrap.com/previews/sb-admin-angular
  - https://startbootstrap.com/?showAngular=false&showVue=false&showPro=false





EXISTING BOOTSTRAP THEMES

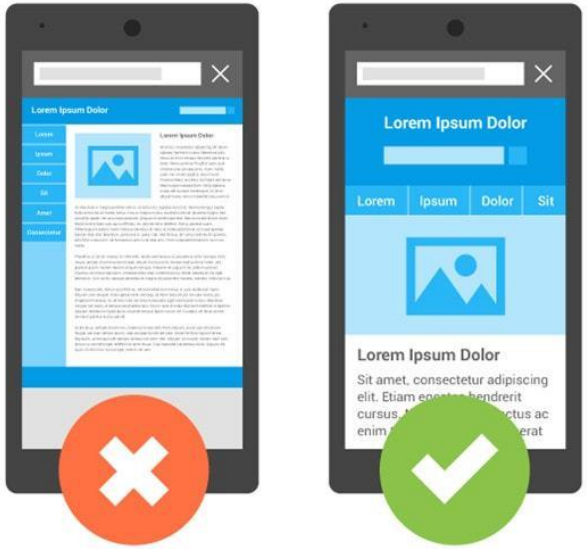
index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Dashboard - SB Admin</title>
        <link href="css/styles.css" rel="stylesheet" />
        <link href="https://cdn.datatables.net/1.10.20/css/dataTables.bootstrap4.min.css" rel="stylesheet" crossorigin="anonymous" />
        <script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/js/all.min.js" crossorigin="anonymous"></script>
    </head>
    <body class="sb-nav-fixed">
        <nav class="sb-topnav navbar navbar-expand navbar-dark bg-dark">
            <a class="navbar-brand" href="index.html">Start Bootstrap</a>
            <button class="btn btn-link btn-sm order-1 order-lg-0" id="sidebarToggle" href="#!"><i class="fas fa-bars"></i></button>
            <!-- Navbar Search-->
            <form class="d-none d-md-inline-block form-inline ml-auto mr-0 mr-md-3 my-2 my-md-0">
                <div class="input-group">
                    <input class="form-control" type="text" placeholder="Search for..." aria-label="Search" aria-describedby="basic-addon2" />
                    <div class="input-group-append">
                        <button class="btn btn-primary" type="button"><i class="fas fa-search"></i></button>
                    </div>
                </div>
            </form>
            <!-- Navbar-->
            <ul class="navbar-nav ml-auto ml-md-0">
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" id="userDropdown" href="#!" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"><i class="fas fa-user fa-fw"></i></a>
                </li>
                <div class="dropdown-menu dropdown-menu-right" aria-labelledby="userDropdown">
                    <a class="dropdown-item" href="#!">Settings</a>
                    <a class="dropdown-item" href="#!">Activity Log</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="login.html">Logout</a>
                </div>
            </ul>
        </nav>
        <div id="layoutSidenav">
            <div id="layoutSidenav_nav">
                <nav class="sb-sidenav accordion sb-sidenav-dark" id="sidenavAccordion">
                    <div class="sb-sidenav-menu">
                        <div class="nav">
                            <div class="sb-sidenav-menu-heading">Core</div>
                            <a class="nav-link" href="index.html">
                                <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt"></i></div>
                                Dashboard
                            </a>
                            <div class="sb-sidenav-menu-heading">Interface</div>
                            <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseLayouts" aria-expanded="false" aria-controls="collapseLayouts">
                                <div class="sb-nav-link-icon"><i class="fas fa-columns"></i></div>
                                Layouts
                                <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-down"></i></div>
                            </a>
                            <div class="collapse" id="collapseLayouts" aria-labelledby="headingOne" data-parent="#sidenavAccordion">
                                <nav class="sb-sidenav-menu-nested nav">
                                    <a class="nav-link" href="layout-static.html">Static Navigation</a>
                                    <a class="nav-link" href="layout-sidenav-light.html">Light Sidenav</a>
                                </nav>
                            </div>
                            <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapsePages" aria-expanded="false" aria-controls="collapsePages">
```

GETTING STARTED

- https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp
  - https://serpstat.com/blog/what-is-a-viewport-meta-tag-and-how-to-use-it/

1. Start with the a basic HTML5 document
2. To ensure proper rendering and touch zooming, add: <meta name="viewport" content="width=device-width, initial-scale=1">
3. Add the Bootstrap source links …
4. Bootstrap requires a container class in which elements to be added. Choose
   - .container class ← for a responsive **fixed width container**
   - .container-fluid class ← for a **full width container**, spanning the entire width of the viewport




```
<!DOCTYPE html>
<html>
<head>
  <title>Getting Started with Bootstrap</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
  <body>
  <div class="container">
   <h1>My First Bootstrap Page</h1>
   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam gravida dignissim est, eget lacinia ipsum varius sed. Curabitur sodales convallis pharetra. Ut feugiat turpis ipsum, in rutrum sem auctor a. Nulla viverra eleifend nibh, ut convallis justo ultricies pharetra. In commodo ornare lectus, eu placerat quam. Ut turpis turpis, faucibus ut sagittis sed, iaculis nec metus. In suscipit nec metus eget molestie. Curabitur dignissim condimentum augue, eget varius enim. Phasellus ultrices iaculis arcu. Curabitur eu lectus nec massa ornare tincidunt vel vel odio.</p>
   </div>
  <div class="container-fluid">
   <h1>My First Bootstrap Page</h1>
   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam gravida dignissim est, eget lacinia ipsum varius sed. Curabitur sodales convallis pharetra. Ut feugiat turpis ipsum, in rutrum sem auctor a. Nulla viverra eleifend nibh, ut convallis justo ultricies pharetra. In commodo ornare lectus, eu placerat quam. Ut turpis turpis, faucibus ut sagittis sed, iaculis nec metus. In suscipit nec metus eget molestie. Curabitur dignissim condimentum augue, eget varius enim. Phasellus ultrices iaculis arcu. Curabitur eu lectus nec massa ornare tincidunt vel vel odio.</p>
   </div>
  </body>
</html>
```

BOOTSTRAP PREBUILT COMPONENTS

- There are many prebuilt components.
- Next, we'll see a few of them

- Note: div defines a **division** or a **section** in an HTML document.
  - By default, browsers always place a line break before and after the element. However, this can be changed with CSS
  - Source: https://www.w3schools.com/tags/tag_div.ASP

BOOTSTRAP GRID SYSTEM

- Sources:
  - https://www.w3schools.com/bootstrap4/bootstrap_grid_basic.asp
  - https://www.w3schools.com/bootstrap4/bootstrap_grid_system.asp
  - https://www.w3schools.com/bootstrap4/bootstrap_grid_xsmall.asp
- Bootstrap's grid system allows up to 12 columns across the page.
  - ```

      .col-sm-4
      .col-sm-8

    ```
- Example to test
  - Try resizing the window
  - "The columns will automatically stack on top of each other when the screen is less than …" pixels➔

Extra small	Small	Medium	Large	Extra Large
.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
<576px	>=576px	>=768px	>=992px	>=1200px

- For a 33% - 67% split use col-4 and col 8
- For a 50% - 50% split use col-6 and col 6
- See also this example

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4				
span 4				span 8								
span 6						span 6						
span 12												
    ```

BOOTSTRAP GRID SYSTEM - SKIP

- Sources:
  - https://www.w3schools.com/bootstrap4/bootstrap_grid_examples.asp

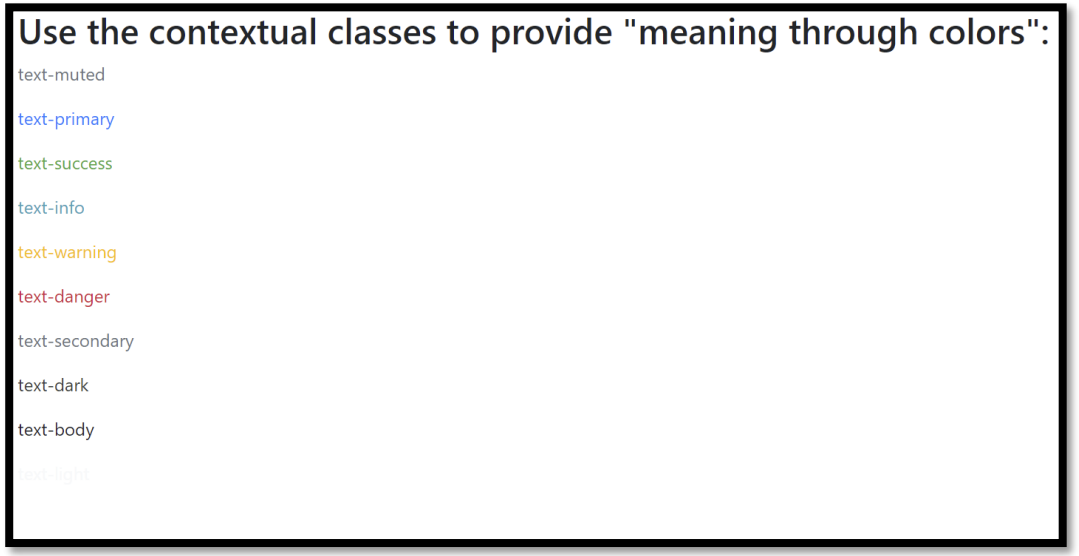
- Check out the examples in here. The following is a "nested columns" example

```
<div class="row">
  <div class="col-8 bg-warning">
    .col-8
    <div class="row">
      <div class="col-6 bg-light">.col-6</div>
      <div class="col-6 bg-secondary">.col-6</div>
    </div>
  </div>
  <div class="col-4 bg-success">.col-4</div>
 </div>
```

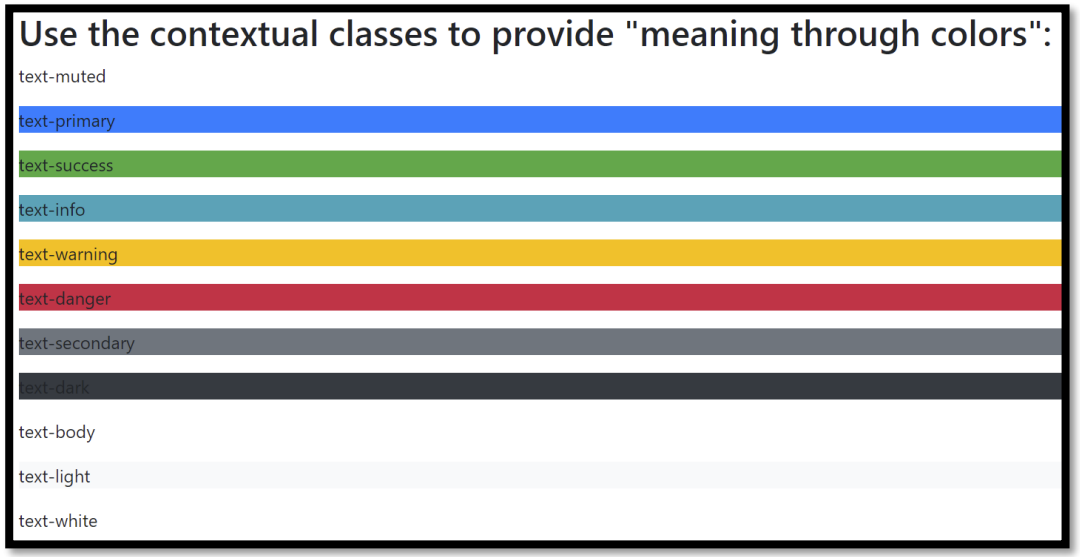
BOOTSTRAP COLORS

- Sources:
  - https://www.w3schools.com/bootstrap4/bootstrap_colors.asp
  - https://www.w3schools.com/bootstrap4/tryit.asp?filename=trybs_txt_colors&stacked=h

```
<div class="container">
  <h2>Use the contextual classes to provide "meaning through colors":</h2>
  <p class="text-muted">text-muted</p>
  <p class="text-primary">text-primary</p>
  <p class="text-success">text-success</p>
  <p class="text-info">text-info</p>
  <p class="text-warning">text-warning</p>
  <p class="text-danger">text-danger</p>
  <p class="text-secondary">text-secondary</p>
  <p class="text-dark">text-dark</p>
  <p class="text-body">text-body</p>
  <p class="text-light">text-light</p>
  <p class="text-white">text-white</p>
</div>
```




```
<div class="container">
  <h2>Use the contextual classes to provide "meaning through colors":</h2>
  <p class="bg-muted">text-muted</p>
  <p class="bg-primary">text-primary</p>
  <p class="bg-success">text-success</p>
  <p class="bg-info">text-info</p>
  <p class="bg-warning">text-warning</p>
  <p class="bg-danger">text-danger</p>
  <p class="bg-secondary">text-secondary</p>
  <p class="bg-dark">text-dark</p>
  <p class="bg-body">text-body</p>
  <p class="bg-light">text-light</p>
  <p class="bg-white">text-white</p>
</div>
```



BOOTSTRAP TABLES

- Sources:
  - https://www.w3schools.com/bootstrap4/bootstrap_tables.asp
  -
  - Combine .table-dark and .table-striped to create a dark, striped table:
  - <table class="table table-dark table-striped">
  - <thead>
  - <tr>
  - <th>Firstname</th>
  - <th>Lastname</th>
  - <th>Email</th>
  - </tr>
  - </thead>
  - <tbody>
  - <tr>
  - <td>John</td>
  - <td>Doe</td>
  - <td>john@example.com</td>
  - </tr>
  - <tr>
  - <td>Mary</td>
  - <td>Moe</td>
  - <td>mary@example.com</td>
  - </tr>
  - </tbody>
  - </table>
  -

- Also try:
  - <table class="table table-dark table-hover">

- See also:
  - https://www.w3schools.com/bootstrap4/tryit.asp?filename=trybs_table_contextual&stacked=h
  - <tr class="table-success">
  - <tr class="table-danger">
  - …

Combine .table-dark and .table-striped to create a dark, striped table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com

Firstname	Lastname	Email
Default	Defaultson	def@somemail.com
Primary	Joe	joe@example.com
Success	Doe	john@example.com
Danger	Moe	mary@example.com
Info	Dooley	july@example.com
Warning	Refs	bo@example.com
Active	Activeson	act@example.com
Secondary	Secondson	sec@example.com
Light	Angie	angie@example.com

BOOTSTRAP ALERTS

- Source: https://www.w3schools.com/bootstrap4/bootstrap_alerts.asp
- Bootstrap alerts are created using the .alert class, followed by one of the **contextual classes** .alert-success, .alert-info, .alert-warning, .alert-danger, .alert-primary, .alert-secondary, .alert-light or .alert-dark

```
<div class="container">
  <div class="alert alert-success">
    <strong>Success!</strong> This alert box could indicate a successful or positive action.
  </div>
  <div class="alert alert-info">
    <strong>Info!</strong> This alert box could indicate a neutral informative change or action.
  </div>
  <div class="alert alert-warning">
    <strong>Warning!</strong> This alert box could indicate a warning that might need attention.
  </div>
  <div class="alert alert-danger">
   <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong>Danger!</strong> This alert box could indicate a dangerous or potentially negative action.
  </div>
  <div class="alert alert-primary">
   <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong>Primary!</strong> Indicates an important action.
  </div>
  <div class="alert alert-secondary">
    <strong>Secondary!</strong> Indicates a slightly less important action.
  </div>
  <div class="alert alert-dark">
    <strong>Dark!</strong> Dark grey alert.
  </div>
  <div class="alert alert-light">
    <strong>Light!</strong> Light grey alert.
  </div>
</div>
```

BOOTSTRAP BUTTONS

- Source: https://www.w3schools.com/bootstrap4/bootstrap_buttons.asp
  - The **button** classes can be used on <a>, <button>, or <input> elements

```
<div class="container">
  <h2>Button Colors</h2>
  <button type="button" class="btn">Basic</button>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-secondary">Secondary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-dark">Dark</button>
  <button type="button" class="btn btn-light">Light</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
```


```
<div class="container">
  <h2>Button Sizes</h2>
  <button type="button" class="btn btn-primary btn-lg">Large</button>
  <button type="button" class="btn btn-primary btn-md">Default</button>
  <button type="button" class="btn btn-primary btn-sm">Small</button>
</div>
```

```
<div class="container">
  <h2>Button States</h2>
  <button type="button" class="btn btn-primary active">Active Primary</button>
  <button type="button" class="btn btn-primary" disabled>Disabled Primary</button>
</div>
```

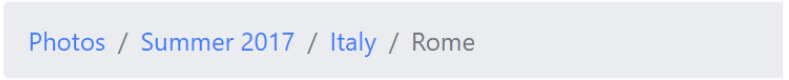
BOOTSTRAP PAGINATION

- Source: https://www.w3schools.com/bootstrap4/bootstrap_pagination.asp

```
<ul class="pagination">
  <li class="page-item">                <a class="page-link" href="#">Previous</a></li>
  <li class="page-item">                <a class="page-link" href="#">1</a></li>
  <li class="page-item active">         <a class="page-link" href="#">2</a></li>
  <li class="page-item">                <a class="page-link" href="#">3</a></li>
  <li class="page-item disabled">                 <a class="page-link" href="#">4</a></li>
  <li class="page-item">                <a class="page-link" href="#">Next</a></li>
</ul>
```

- the breadcrumb class can be used to indicate the current page's location within a navigational hierarchy

```
<ul class="breadcrumb">
  <li class="breadcrumb-item"><a href="#">Photos</a></li>
  <li class="breadcrumb-item"><a href="#">Summer 2017</a></li>
  <li class="breadcrumb-item"><a href="#">Italy</a></li>
  <li class="breadcrumb-item active">Rome</li>
</ul>
```



BOOTSTRAP CARDS

- Source: https://www.w3schools.com/bootstrap4/bootstrap_cards.asp

```
<div class="card" style="width:400px">
  <img class="card-img-top" src="img_avatar1.png" alt="Card image" style="width:100%">
  <div class="card-body">
    <h4 class="card-title">John Doe</h4>
    <p class="card-text">Some example text some example text. John Doe is an architect and engineer</p>
    <a href="#" class="btn btn-primary stretched-link">See Profile</a>
  </div>
</div>
```

- the stretched-link class to a link inside the **card**, and it will make the entire **card** clickable.

BOOTSTRAP NAVS

- Source: https://www.w3schools.com/bootstrap4/bootstrap_navs.asp

- For a simple horizontal menu, use the nav class on a <**ul**> element, then nav-item for each <**li**> and add the nav-link class to their links:

```
<div class="container">
 <ul class="nav">
  <li class="nav-item">          <a class="nav-link" href="#">Link 1</a>                  </li>
  <li class="nav-item">          <a class="nav-link" href="#">Link 2</a>                  </li>
  <li class="nav-item">          <a class="nav-link" href="#">Link 3</a>                  </li>
  <li class="nav-item">          <a class="nav-link disabled" href="#">Link 4</a>         </li>
 </ul>
</div>
```

- Add the flex-column class to create a **vertical** nav:

```
<ul class="nav flex-column">
```

- Use the nav-tabs class to create navigation **tabs**.
  - Add the active class to indicate the active/current link.

- Use the nav-pills class to create…

BOOTSTRAP NAVS - SKIP

- Source: https://www.w3schools.com/bootstrap4/bootstrap_navs.asp

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#">Dropdown</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Link 1</a>
      <a class="dropdown-item" href="#">Link 2</a>
      <a class="dropdown-item" href="#">Link 3</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

BOOTSTRAP NAVBARS/NAVIGATION BARS

- Source: https://www.w3schools.com/bootstrap4/bootstrap_navbar.asp
- A navigation bar/navbar is a **navigation header** placed at the top of the page:

- Try different classes for the <nav> element
  - class="navbar navbar-dark bg-dark"
  - class="navbar navbar-dark bg-primary"
  - class="navbar navbar-light"

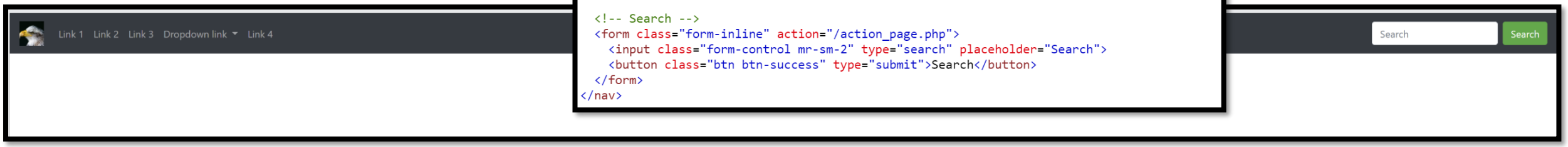
  - class="navbar navbar-light fixed-bottom"
  - sticky-top example

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <!-- Brand/logo -->
  <a class="navbar-brand" href="#">
    <img src="bird.jpg" alt="logo" style="width:40px;">
  </a>

  <!-- Links -->
  <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="#">Link 1</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link 2</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link 3</a>
    </li>

    <!-- Dropdown -->
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-toggle="dropdown">
        Dropdown link
      </a>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#">Link 5</a>
        <a class="dropdown-item" href="#">Link 6</a>
        <a class="dropdown-item" href="#">Link 7</a>
      </div>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link 4</a>
    </li>
  </ul>

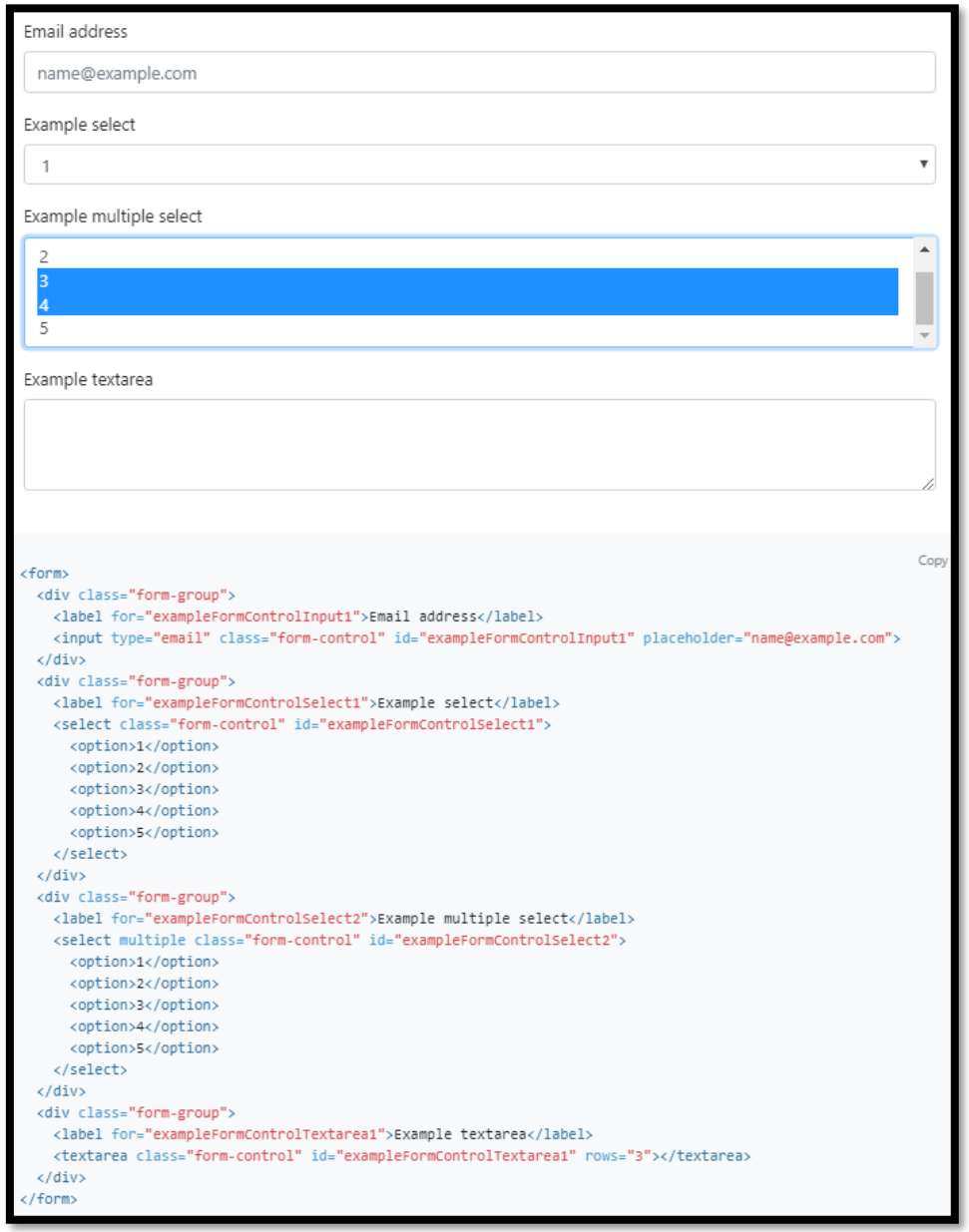
  <!-- Search -->
  <form class="form-inline" action="/action_page.php">
    <input class="form-control mr-sm-2" type="search" placeholder="Search">
    <button class="btn btn-success" type="submit">Search</button>
  </form>
</nav>
```



BOOTSTRAP FORMS - SKIPPED

- Sources:
  - https://www.w3schools.com/bootstrap4/bootstrap_forms.asp
  - https://getbootstrap.com/docs/4.1/components/forms/

- Start with a **div** with the **form-group** class.
  - Each input control will have the **form-control** class.



BOOTSTRAP MODAL

- Sources:
  - https://www.w3schools.com/bootstrap4/bootstrap_navs.asp
  - https://getbootstrap.com/docs/4.1/components/modal/

- Modal windows are dialog boxes/popup windows that are display on top of the current page

```
<!-- Button to Open the Modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModalAlex">
  Open modal window
</button>

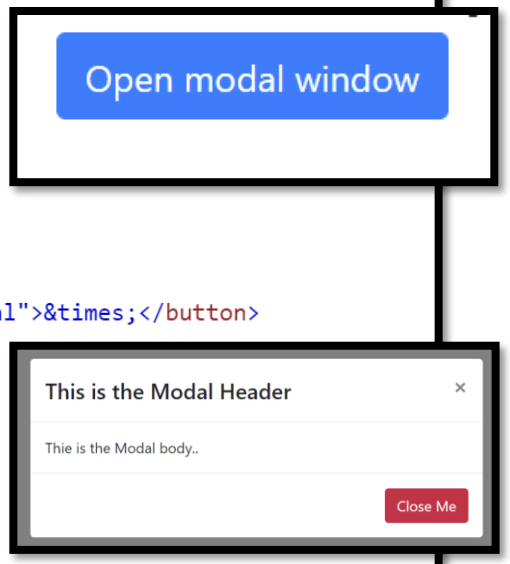
<!-- The Modal -->
<div class="modal" id="myModalAlex">
  <div class="modal-dialog">
    <div class="modal-content">

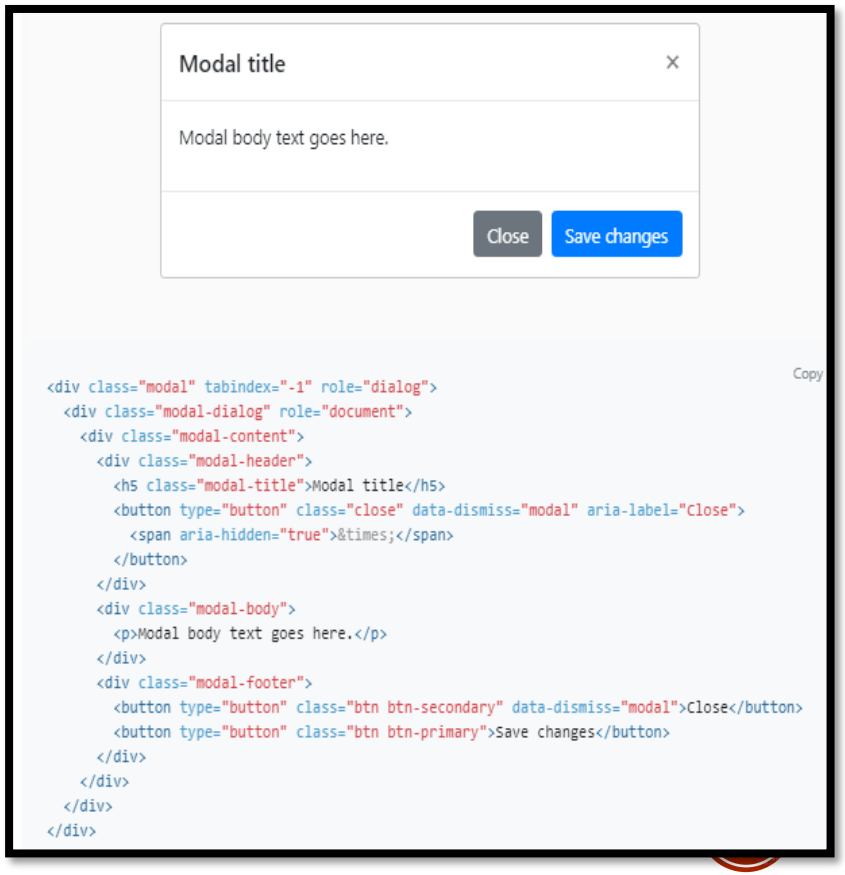
      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">This is the Modal Header</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>

      <!-- Modal body -->
      <div class="modal-body">
        Thie is the Modal body..
      </div>

      <!-- Modal footer -->
      <div class="modal-footer">
        <button type="button" class="btn btn-danger" data-dismiss="modal">Close Me</button>
      </div>

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





CSS MEDIA RULES - EXTRA

- Source:
  - https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

- Media queries allow conditional application of CSS styles, based on the device conditions
- Example: https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_media_bg

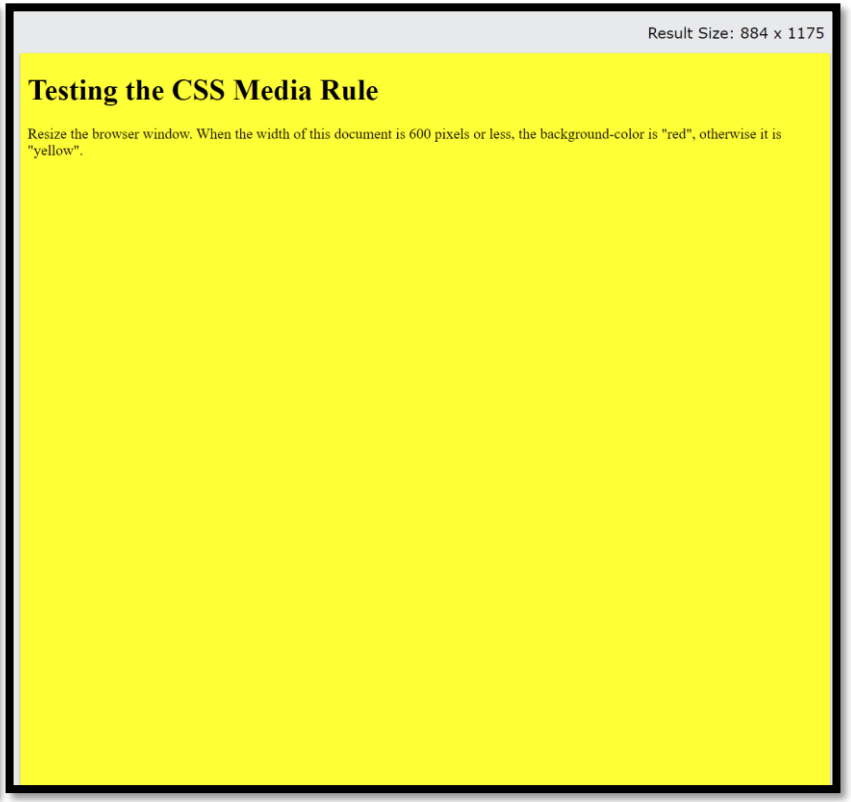
```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {
  background-color: yellow;
}

@media only screen and (max-width: 600px) {
  body {
    background-color: red;
  }
}
</style>
</head>
<body>

<h1>Testing the CSS Media Rule</h1>

<p>Resize the browser window. When the width of this document is 600 pixels
or less, the background-color is "red", otherwise it is "yellow".</p>

</body>
</html>
```



IN-CLASS DEMO

Demonstration: How to Work with Bootstrap

- Source/Steps
  - https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD09_DEMO.md#demonstration-how-to-work-with-bootstrap

IN-CLASS DEMO

Demonstration: How to Use the Bootstrap Grid System

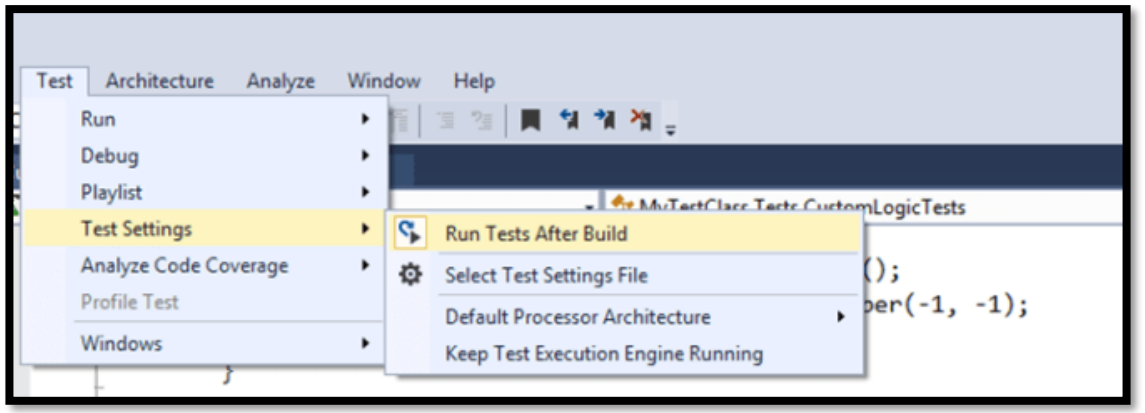
- Source/Steps
  - https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD09_DEMO.md#demonstration-how-to-use-the-bootstrap-grid-system

TESTING YOUR CODE

- Source: https://www.perfecto.io/resources/types-of-testing

- There are many different types of testing, including:
  - Accessibility testing
  - Acceptance testing
  - Black box testing
  - Functional testing
  - Integration testing
  - Load testing
  - Non functional testing
  - Performance testing
  - Regression testing
  - Sanity testing
  - Security testing
  - Stress testing
  - Unit testing
  - White box testing
  - …

- Testing can be done **manually,** or it can be **automated**.



UNIT TESTING

- Sources:
  - https://smartbear.com/learn/automated-testing/what-is-unit-testing/
  - https://en.wikipedia.org/wiki/Test-driven_development
  - https://www.justinweiss.com/articles/writing-better-tests-with-the-three-phase-pattern/

- "A **unit test** is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property. "
  - We'll put multiple related unit tests into a class, and we'll call that class a test fixture.
  - Each **unit test** has three steps:
    - Arrange: creates an instance of the class you're testing. Set up any data needed to run the test.
    - Act: runs the functionality you want to test. For example, call a method.
    - Assert: checks the **obtained result** against the **expected result**.

- Why we care about **unit testing**?
  - Because tests can be done in isolation from the other parts of system, we can test the code as soon as we write the code.
  - Once you write them, you can rerun those tests as many times as needed (e.g. after every major code changes).

- Test Driven Development (TDD)
  - "is a software development process relying on software requirements being converted to test cases before software is fully developed, and tracking all software development by repeatedly testing the software against all test cases."
  - So in TDD you start with the tests, then you write the code and make sure it passes those tests.

TESTING THE MVC APPLICATION

- Models are usually independent classes so they should be easy to test.
  - Arrange: create an instance, set up some values needed for testing
  - Act: call a method
  - Assert: check the result of the method against the expected result

- Controllers are more challenging because they may involve data coming from a database.
  - To perform unit testing on controllers, you need to isolate your application from the underlying database.
  - For this, one can make use of the **repository pattern**.
    - This way, we can test with "fake" in-memory data, rather than data from a database.

- Views [EXTRA]: … unit testing in here is different … "*In many projects, the UI is taken for granted to work ok if the backend code is tested ok, only thing people test is the alignment and layouts and look and feel. This cannot be tested by a test case.*"
  - https://www.c-sharpcorner.com/forums/how-to-unit-test-mvc-views

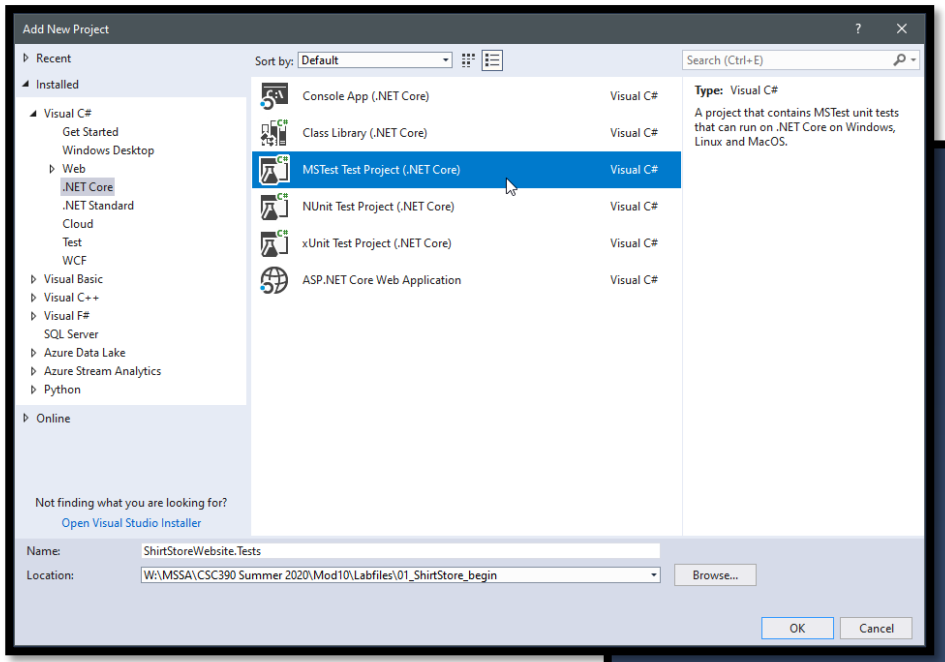
MSTEST TEST PROJECT (.NET CORE)

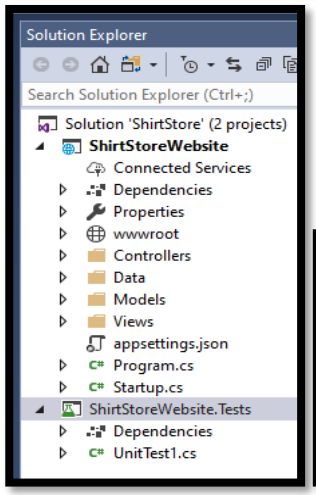
- To add **unit tests** to your application, you can **add a test project** based on the **MSTest Test Project (.NET Core)** template
  - To your **test project** you will need to add a reference to your **MVC web application project** so that the test code can access those classes.
  - You'll also need to install the **Microsoft.AspNetCore.Mvc** package in the **test project**.

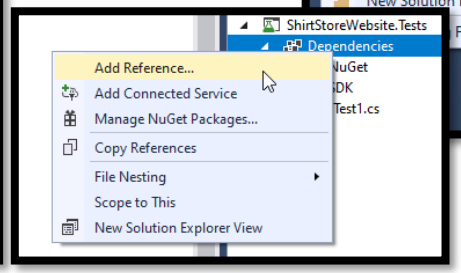
- In a MSTest test project you will then create
  - **test fixtures** ← classes marked with the **[TestClass]** attribute.
  - **unit tests** ← void returning methods marked with the **[TestMethod]** attribute

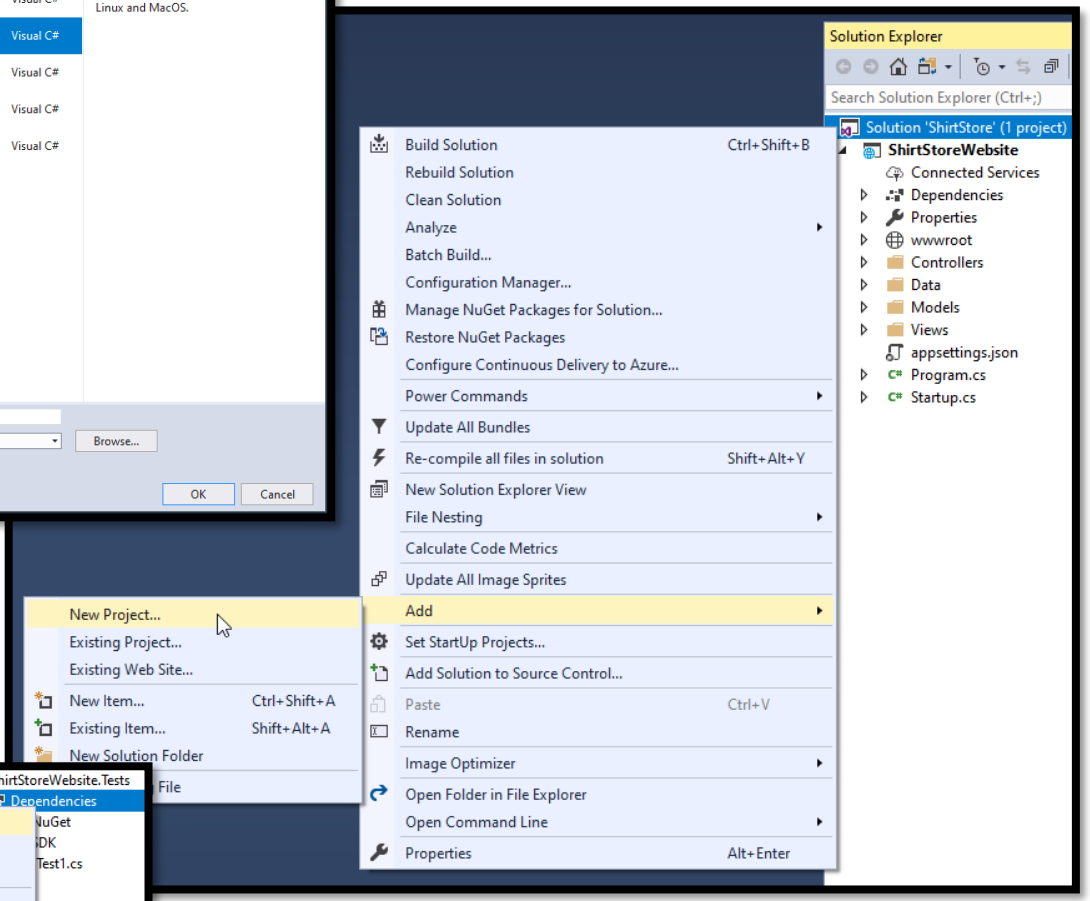
MSTEST TEST PROJECT (.NET CORE)

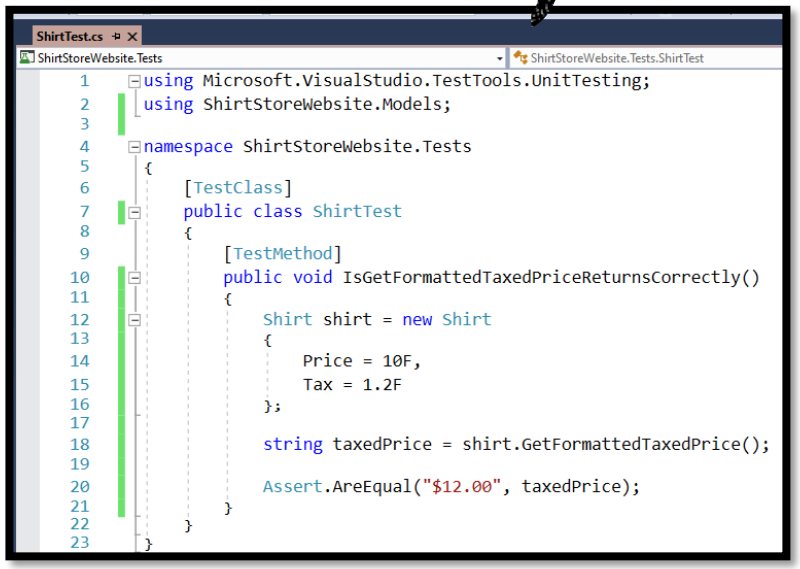
- Adding a Unit Testing Project … Part of your homework…

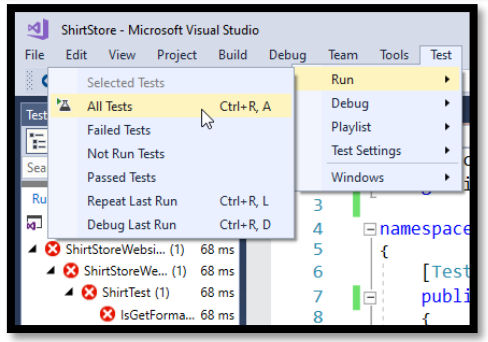


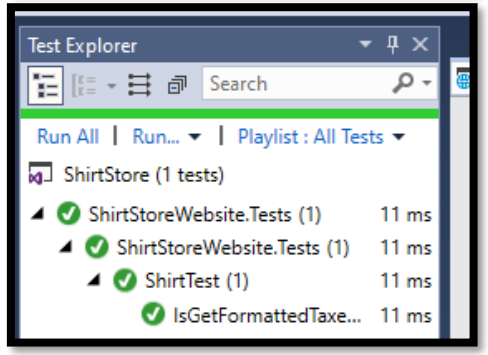












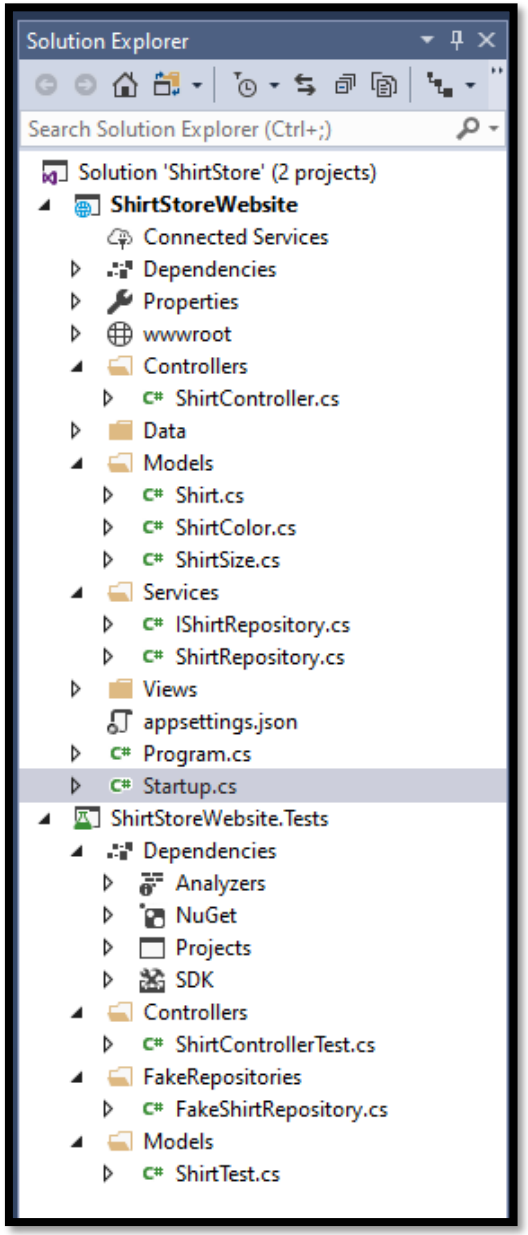
TESTING MODELS

- From your homework

ShirtTest.cs

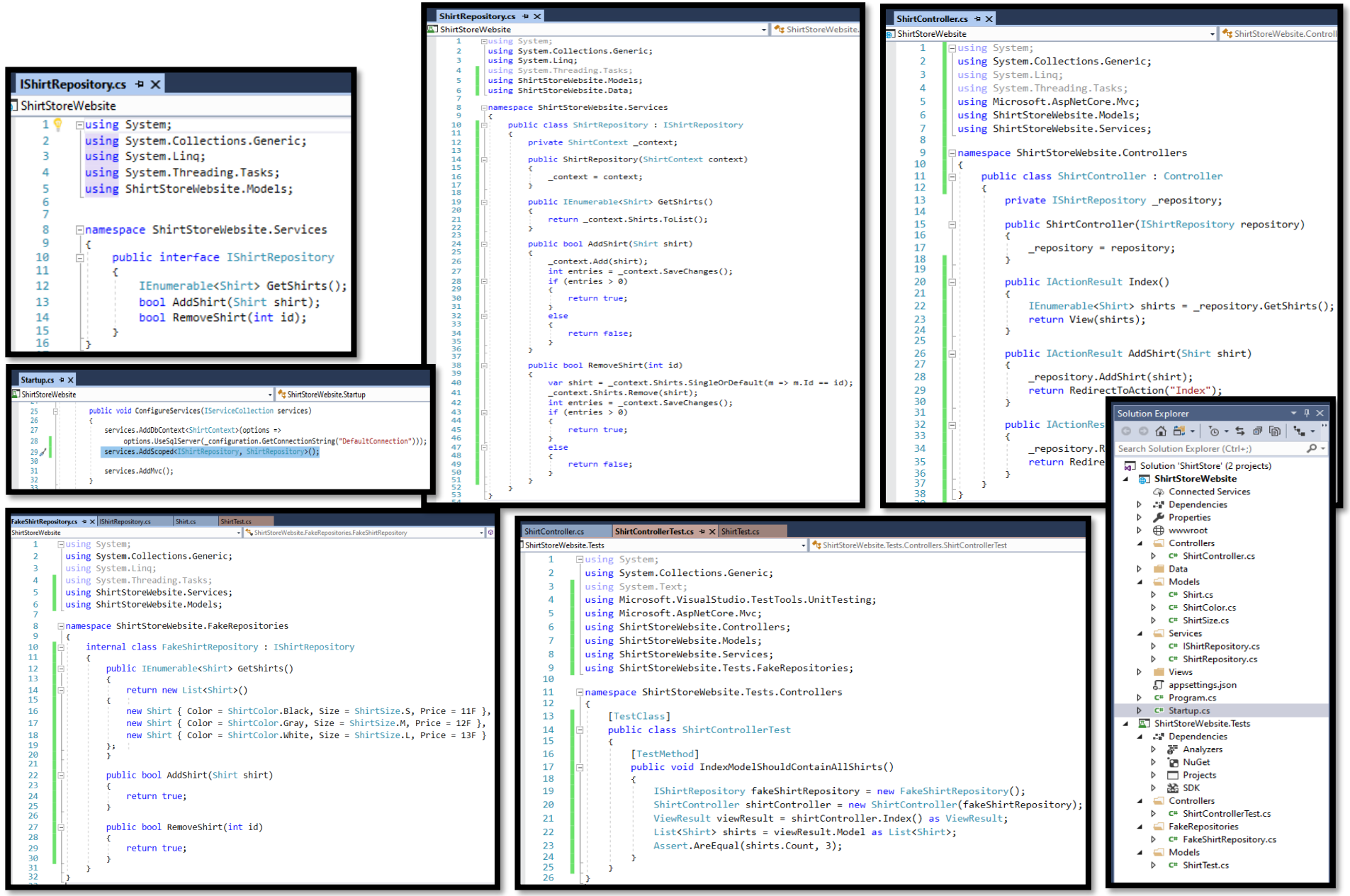
ShirtStoreWebsite.Tests | ShirtStoreWebsite.Tests.ShirtTest

```
1   using Microsoft.VisualStudio.TestTools.UnitTesting;
2   using ShirtStoreWebsite.Models;
3
4   namespace ShirtStoreWebsite.Tests
5   {
6       [TestClass]
7       public class ShirtTest
8       {
9           [TestMethod]
10          public void IsGetFormattedTaxedPriceReturnsCorrectly()
11          {
12              Shirt shirt = new Shirt
13              {
14                  Price = 10F,
15                  Tax = 1.2F
16              };
17
18              string taxedPrice = shirt.GetFormattedTaxedPrice();
19
20              Assert.AreEqual("$12.00", taxedPrice);
21          }
22      }
23  }
```



TESTING CONTROLLERS

- From your homework



IShirtRepository.cs
ShirtStoreWebsite

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ShirtStoreWebsite.Models;


namespace ShirtStoreWebsite.Services
{
    public interface IShirtRepository
    {
        IEnumerable<Shirt> GetShirts();
        bool AddShirt(Shirt shirt);
        bool RemoveShirt(int id);
    }
}
```

ShirtRepository.cs
ShirtStoreWebsite

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ShirtStoreWebsite.Models;
using ShirtStoreWebsite.Data;

namespace ShirtStoreWebsite.Services
{
    public class ShirtRepository : IShirtRepository
    {
        private ShirtContext _context;

        public ShirtRepository(ShirtContext context)
        {
            _context = context;
        }

        public IEnumerable<Shirt> GetShirts()
        {
            return _context.Shirts.ToList();
        }

        public bool AddShirt(Shirt shirt)
        {
            _context.Add(shirt);
            int entries = _context.SaveChanges();
            if (entries > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public bool RemoveShirt(int id)
        {
            var shirt = _context.Shirts.SingleOrDefault(m => m.Id == id);
            _context.Shirts.Remove(shirt);
            int entries = _context.SaveChanges();
            if (entries > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

ShirtController.cs
ShirtStoreWebsite

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using ShirtStoreWebsite.Models;
using ShirtStoreWebsite.Services;

namespace ShirtStoreWebsite.Controllers
{
    public class ShirtController : Controller
    {
        private IShirtRepository _repository;

        public ShirtController(IShirtRepository repository)
        {
            _repository = repository;
        }

        public IActionResult Index()
        {
            IEnumerable<Shirt> shirts = _repository.GetShirts();
            return View(shirts);
        }

        public IActionResult AddShirt(Shirt shirt)
        {
            _repository.AddShirt(shirt);
            return RedirectToAction("Index");
        }

        public IActionRes
        {
            _repository.R
            return Redire
        }
    }
}
```

Startup.cs
ShirtStoreWebsite
ShirtStoreWebsite.Startup

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ShirtContext>(options =>
        options.UseSqlServer(_configuration.GetConnectionString("DefaultConnection")));
    services.AddScoped<IShirtRepository, ShirtRepository>();

    services.AddMvc();
}
```

Solution Explorer

```
Search Solution Explorer (Ctrl+;)
Solution 'ShirtStore' (2 projects)
  ShirtStoreWebsite
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
      ShirtController.cs
    Data
    Models
      Shirt.cs
      ShirtColor.cs
      ShirtSize.cs
    Services
      IShirtRepository.cs
      ShirtRepository.cs
    Views
    appsettings.json
    Program.cs
    Startup.cs
  ShirtStoreWebsite.Tests
    Dependencies
      Analyzers
      NuGet
      Projects
      SDK
    Controllers
      ShirtControllerTest.cs
    FakeRepositories
      FakeShirtRepository.cs
    Models
      ShirtTest.cs
```

FakeShirtRepository.cs | IShirtRepository.cs | Shirt.cs | ShirtTest.cs
ShirtStoreWebsite
ShirtStoreWebsite.FakeRepositories.FakeShirtRepository

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ShirtStoreWebsite.Services;
using ShirtStoreWebsite.Models;

namespace ShirtStoreWebsite.FakeRepositories
{
    internal class FakeShirtRepository : IShirtRepository
    {
        public IEnumerable<Shirt> GetShirts()
        {
            return new List<Shirt>()
            {
                new Shirt { Color = ShirtColor.Black, Size = ShirtSize.S, Price = 11F },
                new Shirt { Color = ShirtColor.Gray, Size = ShirtSize.M, Price = 12F },
                new Shirt { Color = ShirtColor.White, Size = ShirtSize.L, Price = 13F }
            };
        }

        public bool AddShirt(Shirt shirt)
        {
            return true;
        }

        public bool RemoveShirt(int id)
        {
            return true;
        }
    }
}
```

ShirtController.cs | ShirtControllerTest.cs | ShirtTest.cs
ShirtStoreWebsite.Tests
ShirtStoreWebsite.Tests.Controllers.ShirtControllerTest

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Microsoft.AspNetCore.Mvc;
using ShirtStoreWebsite.Controllers;
using ShirtStoreWebsite.Models;
using ShirtStoreWebsite.Services;
using ShirtStoreWebsite.Tests.FakeRepositories;

namespace ShirtStoreWebsite.Tests.Controllers
{
    [TestClass]
    public class ShirtControllerTest
    {
        [TestMethod]
        public void IndexModelShouldContainAllShirts()
        {
            IShirtRepository fakeShirtRepository = new FakeShirtRepository();
            ShirtController shirtController = new ShirtController(fakeShirtRepository);
            ViewResult viewResult = shirtController.Index() as ViewResult;
            List<Shirt> shirts = viewResult.Model as List<Shirt>;
            Assert.AreEqual(shirts.Count, 3);
        }
    }
}
```

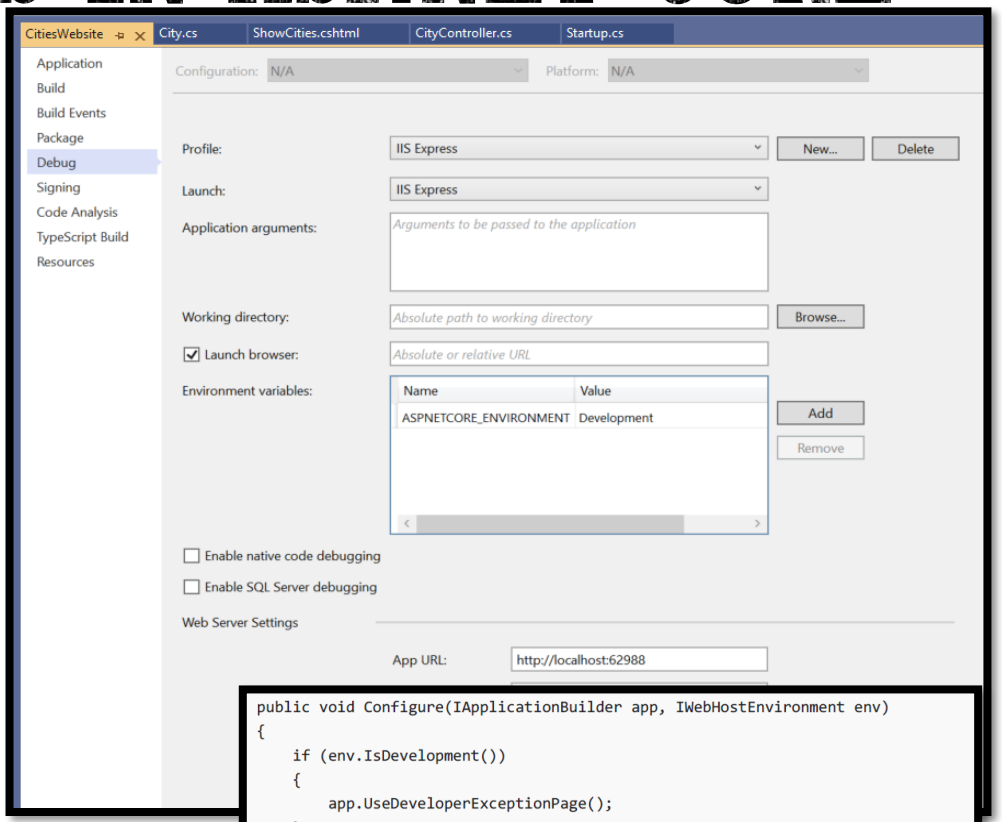
IN-CLASS DEMO

Demonstration: How to Run Unit Tests

- Source/Steps
  - https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD10_DEMO.md#demonstration-how-to-run-unit-tests

USE MULTIPLE ENVIRONMENTS IN ASP.NET CORE

- See page 1184
- Debug > Properties > use **ASPNETCORE_ENVIRONMENT** to set the current env.
  - Typical values: **Development**, **Staging**, **Production** (default if not value set)



- To check for the value of the environment, one can use the **IWebHostingEnvironment** service (former **IHostingEnvironment** service). Useful methods:
  - IsDevelopment()
  - IsStaging()
  - IsProduction()
  - IsEnvironment("Put some Environment Name here")

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    if (env.IsProduction() || env.IsStaging() || env.IsEnvironment("Staging_2"))
    {
        app.UseExceptionHandler("/Error");
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```

- We'll talk about **UseExceptionHandler** and **UseDeveloperExceptionPage** on the next slides…

DEVELOPER EXCEPTION PAGE

- If you add the **UseDeveloperExceptionPage** middleware to the **Configure** method, you will be able to get a page that has detailed information about the exception that occurred.
- Beware: never use this in **production** environment! Why?

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    if (env.IsProduction() || env.IsStaging() || env.IsEnvironment("Staging_2"))
    {
        app.UseExceptionHandler("/Error");
    }

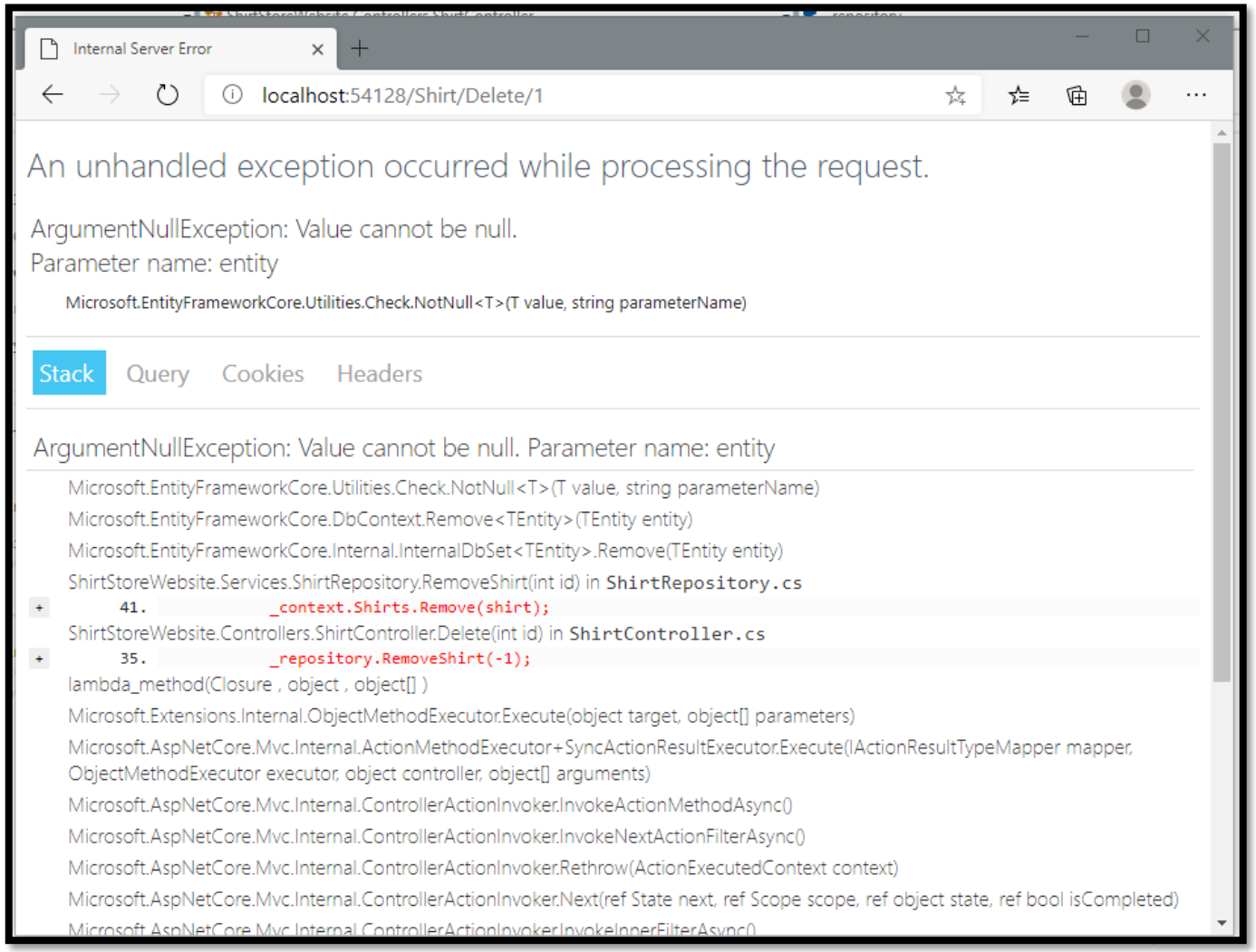
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```

- Part of your homework



CUSTOM EXCEPTION HANDLER PAGE

- See also:
  - https://www.tutorialsteacher.com/core/aspnet-core-exception-handling

- A friendly alternative to **UseDeveloperExceptionPage** middleware is the **UseExceptionHandler** middleware.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

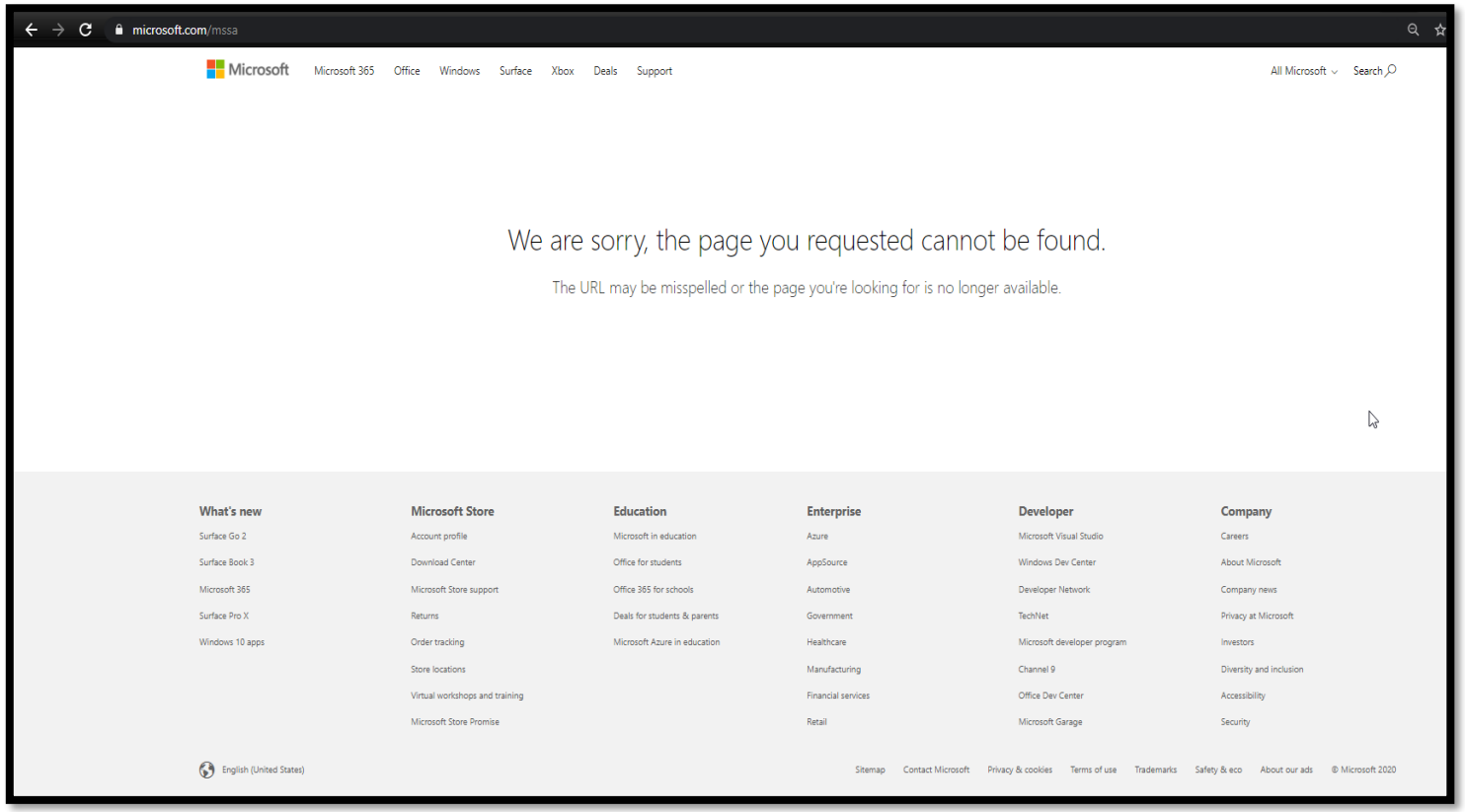
    if (env.IsProduction() || env.IsStaging() || env.IsEnvironment("Staging_2"))
    {
        app.UseExceptionHandler("/Cupcake/Error");
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

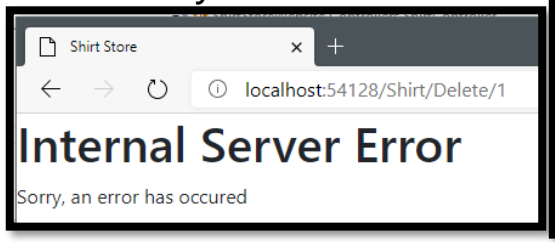
    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```



- Part of your homework



CUSTOM EXCEPTION HANDLER PAGE

- See also:
  - https://www.infoworld.com/article/3545304/how-to-handle-404-errors-in-aspnet-core-mvc.html

- To capture the 404 (not found) errors add code similar to the one below before routing:

```
app.Use(async (context, next) =>
{
    await next();
    if (context.Response.StatusCode == 404)
    {
        context.Request.Path = "/Cupcake/Error";
        await next();
    }
});
```

- This is code base on module 7 demo code … ➔ ➔ ➔ ➔ ➔ ➔

- You can also use: app.UseStatusCodePages();

```
public void Configure(IApplicationBuilder app, CupcakeContext cupcakeContext, ILogger<Startup> log)
{

    log.LogError($"startup at {DateTime.Now}");

    app.UseExceptionHandler("/Cupcake/Error");

    app.Use(async (context, next) =>
    {
        await next();
        if (context.Response.StatusCode == 404)
        {
            context.Request.Path = "/Cupcake/Error";
            await next();
        }
    });

    app.UseStaticFiles();

    cupcakeContext.Database.EnsureCreated();

    app.UseMvc(routes =>
    {
        routes.MapRoute(
            name: "CupcakeRoute",
            template: "{controller}/{action}/{id?}",
            defaults: new { controller = "Cupcake", action = "Index" },
            constraints: new { id = "[0-9]+" });
    });
}
```

IN-CLASS DEMO

Demonstration: How to Configure Exception Handling

- Source/Steps
  - https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD10_DEMO.md#demonstration-how-to-configure-exception-handling

LOGGING IN .NET CORE AND ASP.NET CORE

- Source:
  - For this and the remaining slides, use Page 1208+ and the Demo shown below

- How do you debug an application that's in production? Can you use break points and stop it?
- What information would you log?

- In here we'll see **logging providers**. Logging providers are used to **store** logs (one exception: Console provider **displays** logs)

STEP 1: CONFIGURE LOGGING PROVIDERS

- First step you'll need to set up which providers to use. In the Program.cs > CreateWebHostBuilder ← Call the ConfigureLogging method

- To override the default set of logging providers added by Host.CreateDefaultBuilder , call **ClearProviders** and add the logging providers you want to use.

- Call the add*Provider Name*() method – then you're all set to use it:
  - **Console** ← logs messages to the application's console window.
  - **Debug** ← logs to the application's debug window.
  - **EventSource** …
  - **EventLog** …
  - **TraceSource** …
  - **AzureAppServices** …
  - **File** …

- Logging configuration is typically provided by the **Logging** section of **appsettings.{Environment}.json** files
  - The **LogLevel** specifies the minimum level to log for selected categories.
  - If no **LogLevel** is specified, logging defaults to the **Information** level.

- Note: one can add multiple providers

```
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
    .ConfigureLogging((hostingContext, logging) =>
    {
        var env = hostingContext.HostingEnvironment;
        var config = hostingContext.Configuration.GetSection("Logging");

        logging.ClearProviders();

        if (env.IsDevelopment())
        {
            logging.AddConfiguration(config);
            logging.AddConsole();
        }
        else
        {
            logging.AddFile(config);
        }
    })
        .UseStartup<Startup>();
```

STEP A: USE LOGGING PROVIDERS

- **inject** a logger where you need it. Use ILogger<anyType>
  - Note: anyType can be any class (typically the name of class where it is used)

- then call the Log*Level Name*() method to log information. Options:
- Trace (level 0): Contain the most detailed messages. Should not be enabled in production
- Debug (1): used for debugging and development
- Information (2): tracks general flow of the application.
- Warning (3): use if for abnormal and unexpected events
- Error (4): use it for exceptions cannot be handled
- Critical (5): use if for failures that require immediate attention
- None (6): … see oage 1215.

- We can pass a "string/error message"
- We can also pass an ID (always the first parameter)
- We can also pass an exception object (come after ID, if any ID)

- Also see page 1245+

```
public class HomeController : Controller
{
    IDivisionCalculator _numberCalculator;
    ICounter _counter;
    ILogger _logger;

    public HomeController(IDivisionCalculator numberCalculator, ICounter counter, ILogger<HomeController> logger)
    {
        _counter = counter;
        _numberCalculator = numberCalculator;
        _logger = logger;
    }


    public IActionResult GetDividedNumber(int id)
    {
        ViewBag.CounterSucceeded = false;
        try
        {
            _counter.IncrementNumberCount(id);
            ViewBag.NumberOfViews = _counter.NumberCounter[id];
            ViewBag.CounterSucceeded = true;
            _logger.LogError("GetDividedNumber - Success ");
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occured while trying to increase or retrieve the page display count. Number parameter is: {id}");
        }
```

IN-CLASS DEMO

Demonstration: How to Log an MVC Application

- Source/Steps
  - https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD10_DEMO.md#demonstration-how-to-log-an-mvc-application

LAB/HOMEWORK: CLIENT-SIDE DEVELOPMENT

- **Module 09**
  - ~~Exercise 1: Using gulp to Run Tasks~~
  - ~~Exercise 2: Styling Using Sass~~
  - Exercise 3: Using Bootstrap

- You will find the **high-level** steps on the following page:
  https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD09_LAB_MANUAL.md
- You will find the **detailed** steps on the following page:
  https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD09_LAK.md
- For your homework submit one zipped folder with your complete solution.

LAB/HOMEWORK: TESTING AND TROUBLESHOOTING

- **Module 10**
  - Exercise 1: Testing a Model
  - Exercise 2: Testing a Controller using a Fake Repository
  - Exercise 3: Implementing a Repository in the MVC Project
  - Exercise 4: Adding Exception Handling
  - Exercise 5: Adding Logging

- You will find the **high-level** steps on the following page:
  https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD10_LAB_MANUAL.md
- You will find the **detailed** steps on the following page:
  https://github.com/MicrosoftLearning/20486D-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486D_MOD10_LAK.md

- For your homework submit one zipped folder with your complete solution.