GETTING STARTED WITH FULL STACK DEVELOPER (FSD) IN DORDSTREAM

DORDSTREAM
FULL STACK DEVELOPERS

Getting Started with Full stack Developer (FSD) in DordStream

This tutorial teaches the basics of building a DordStream code library (Themes) for web software. We recommend you to host DordStream CMS software on your local server (Computer) (internet information services (IIS)). before starting this tutorial.

Prerequisites

Install the following:

- .NET Desktop Runtime 5.0.0.
- Evergreen Bootstrapper
- DordStream Studio.

Languages required:

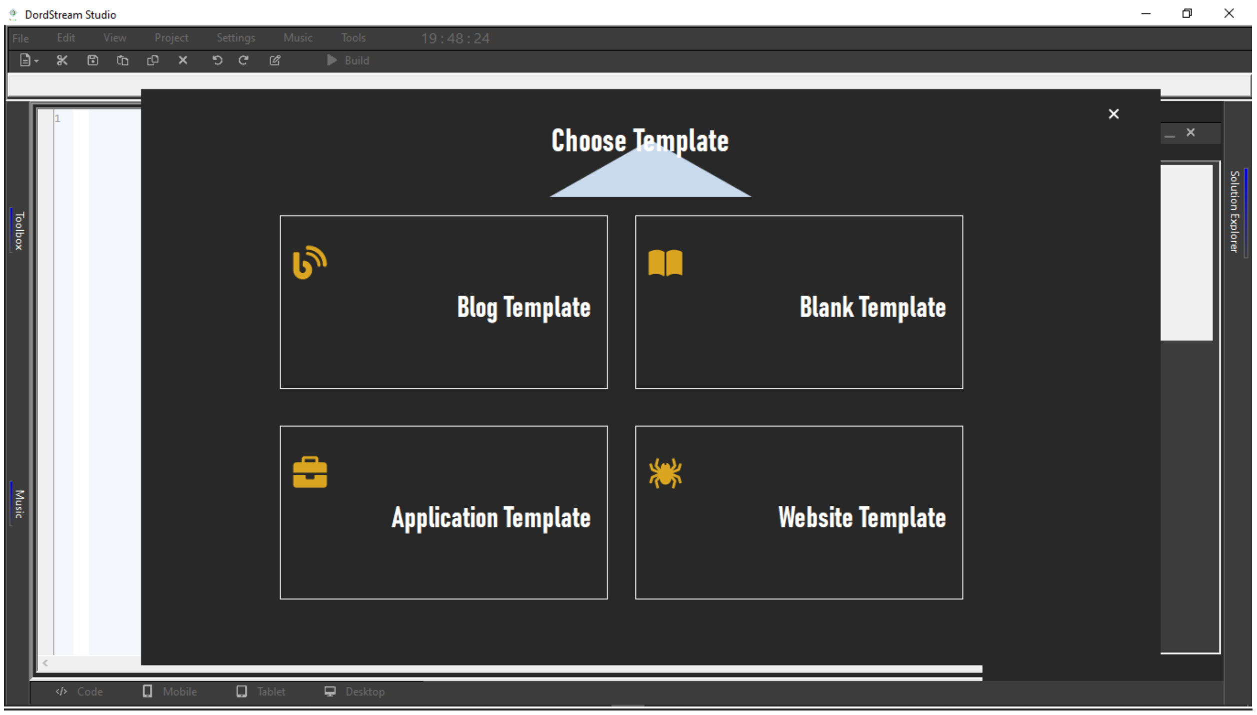
- Html
- Css
- Javascript
- C#

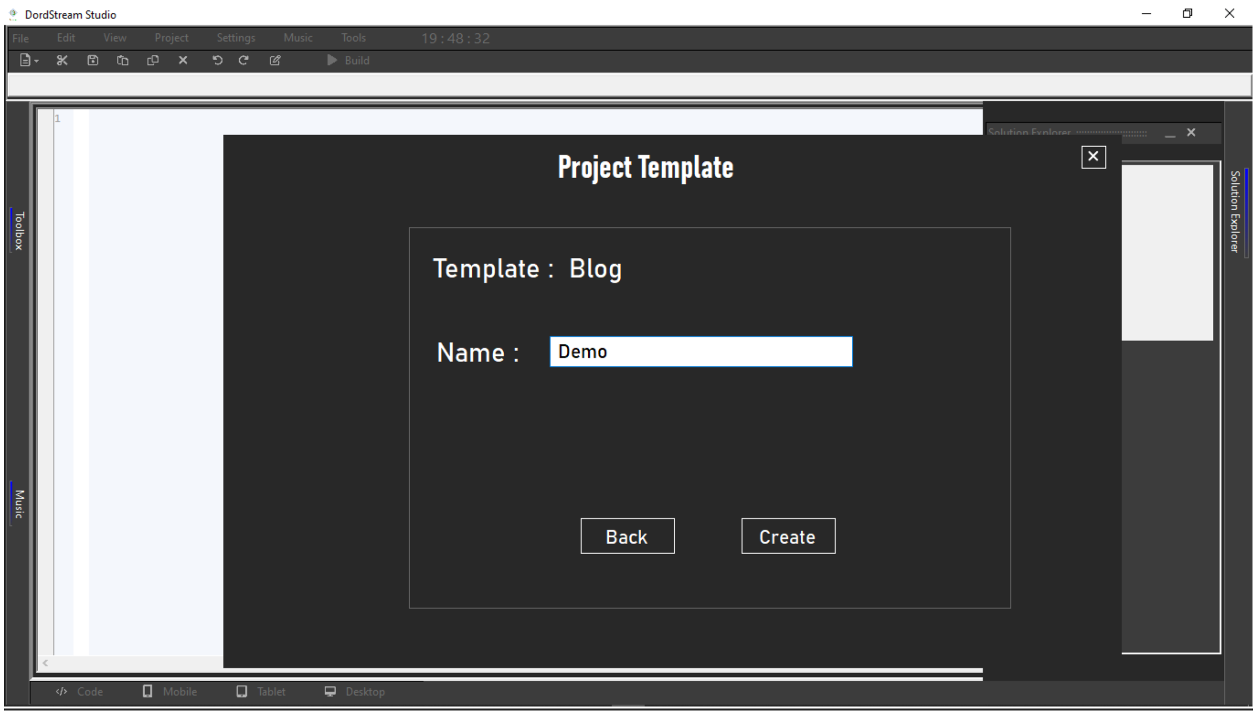
Create a DordStream Project

From the DordStream Studio **File** menu, select **New** > **Project**.

Create a new project and select project template. Name the project **Demo**. It's important to name the project *Demo* so the namespaces will match when publishing.

- Select the template, and name your project.



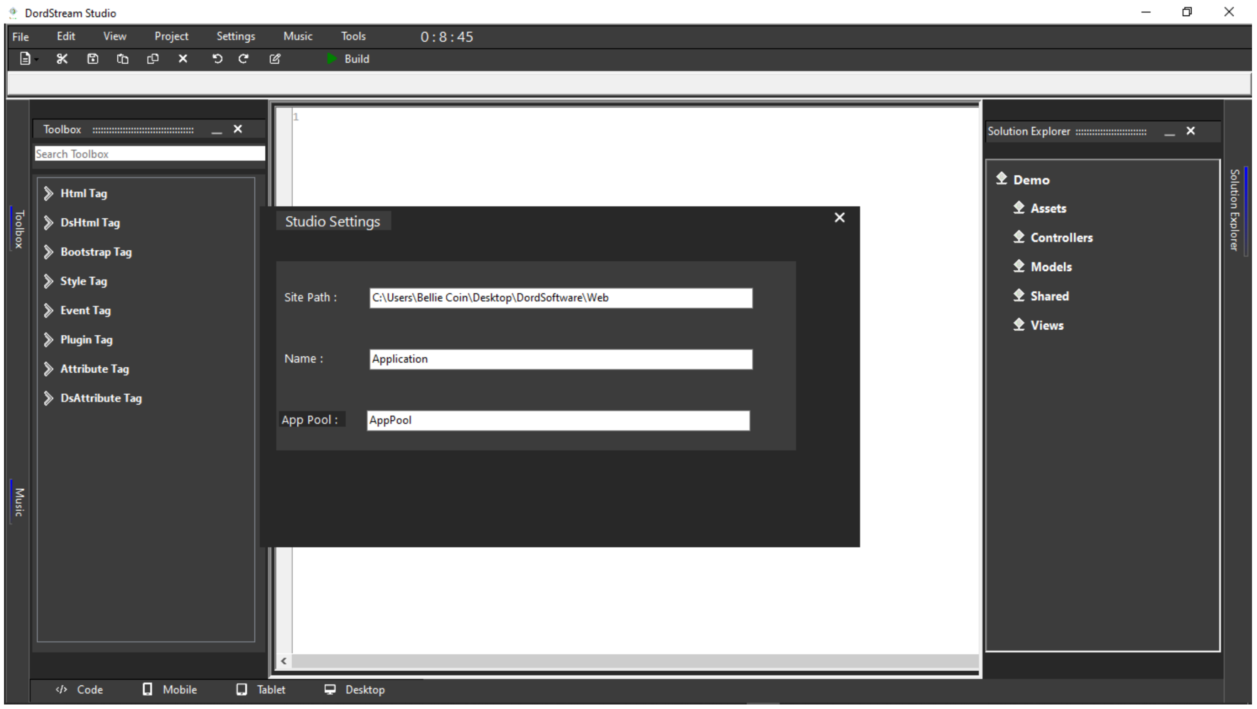


Project Template

Template : Blog

Name : Demo

- Connect **DordStream Studio** with the **DordStream CMS** hosted on Internet Information Service (IIS). From the DordStream Studio **Settings** menu



Studio Settings

Site Path : C:\Users\Bellie Coin\Desktop\DordSoftware\Web

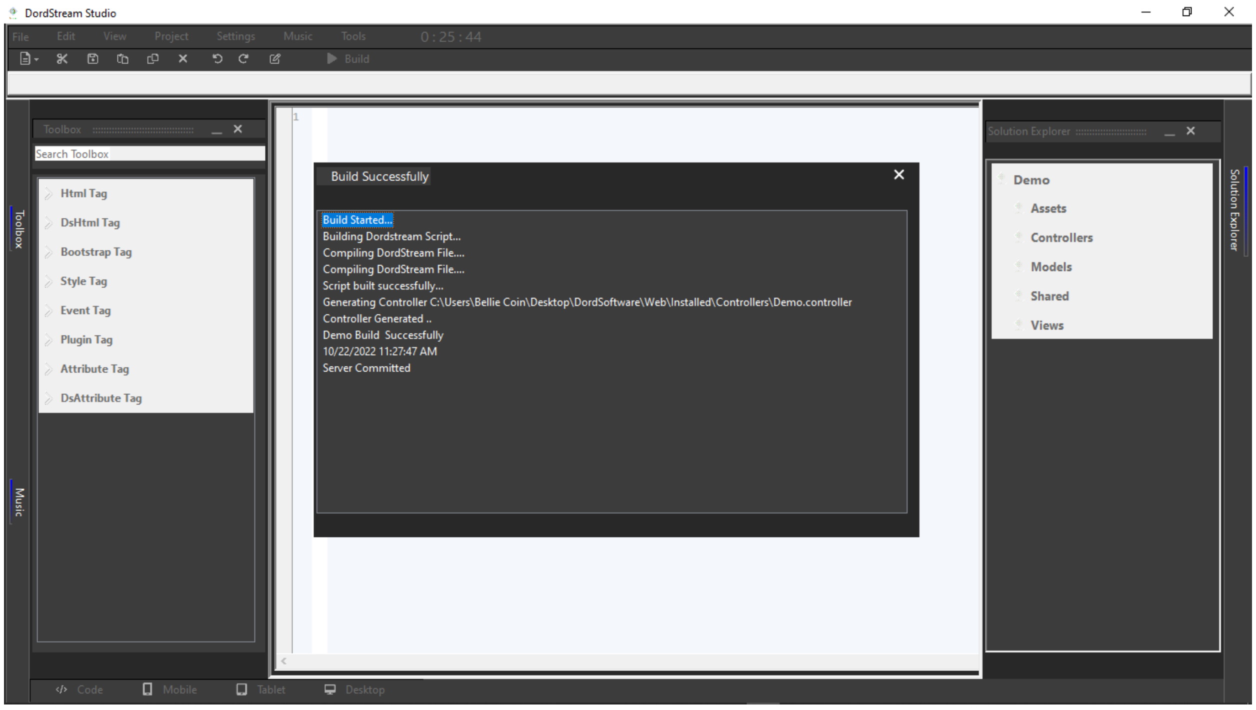
Name : Application

App Pool : AppPool

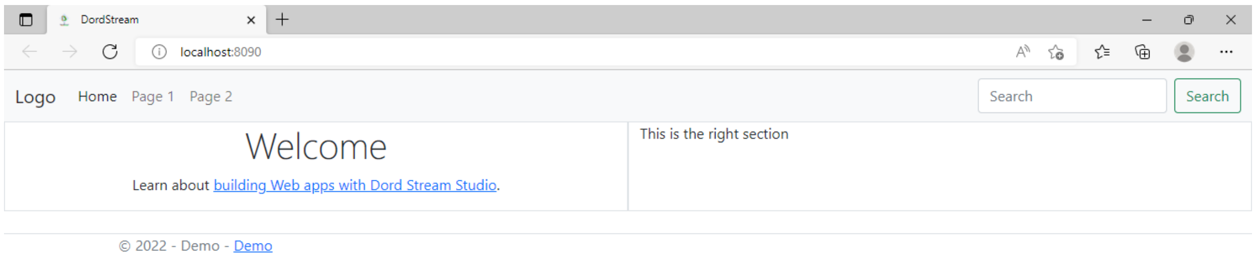
- **Site Path** is the path to the software (**DordStream CMS).** double click on the textbox to select the path to the software then press **Enter Key** to save the path.

- **Name** is the site name given to the hosted software (**DordStream CMS)** on Internet information services (IIS). After inputting the name then press **Enter Key** to save the site name.
- **App Pool** is the application pool name of the hosted software (**DordStream CMS)** on Internet information services (IIS). After inputting the application pool name then press **Enter Key** to save the application pool.

- Click on **Build** to run the app in debug mode.



```
Build Started...
Building Dordstream Script...
Compiling DordStream File....
Compiling DordStream File....
Script built successfully...
Generating Controller C:\Users\Bellie Coin\Desktop\DordSoftware\Web\Installed\Controllers\Demo.controller
Controller Generated ..
Demo Build  Successfully
10/22/2022 11:27:47 AM
Server Committed
```



Welcome

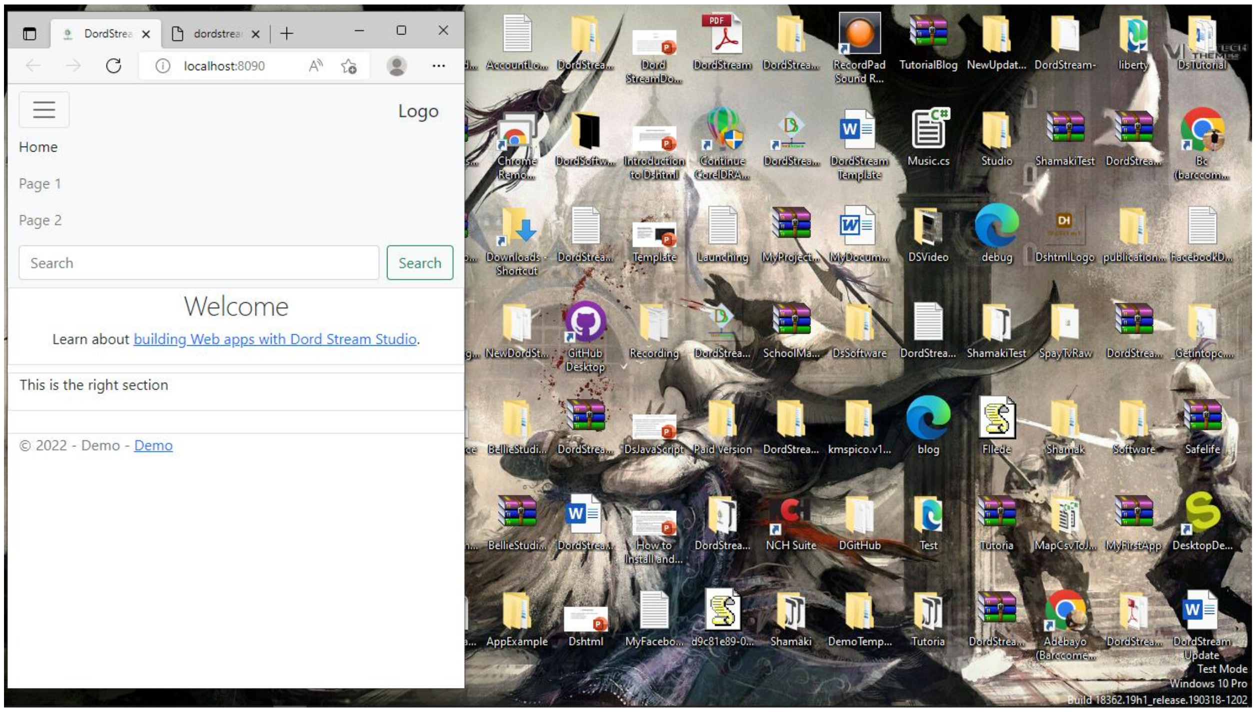
Learn about building Web apps with Dord Stream Studio.

This is the right section

© 2022 - Demo - Demo

- Run web browser and access the localhost host on IIS. The address bar shows `localhost:port#` and not something like `example.com`. That's because `localhost` is the standard hostname for your local computer. Localhost only serves web requests from the local computer. When Visual Studio creates a web project, a random port is used for the web server. In the preceding image, the port number is 8090. When you run the app, you'll see a different port number.

The default template creates **Demo**, **Home**, **Page 1** and **Page 2** links and pages. Depending on the size of your browser window, you might need to click the navigation icon to show the links.



Welcome

Learn about building Web apps with Dord Stream Studio.

This is the right section

© 2022 - Demo - Demo

Project files and folders

The following table lists the files and folders in the project. For this tutorial

Folder	Purpose
Assets	This is a folder that contains set of files generally text content, graphics, image, audio, video, Javascript, Css, etc. which server does not change when sending to users.

Folder	Purpose
Shared	This is a folder that contains set of files that will be shared throughout the application. Shared folder has a required (compulsory) files and optional files. compulsory files are layout.html, landing.html, readme.txt, banner.png | banner.jpg and error.html optional files depend on usage. they are login.html, register.html, post.html, category.html and page.html depending on the project template • Layout (layout.html). This is a file that contain the hypertext markup language which defines the project layout. • Landing (landing.html). This is a file that contain the hypertext markup language which defines the landing page interface (first page to be loaded when running the application). • ReadMe (readme.txt). This is a file that contain project description. • Error (error.html). This is a file that contain hypertext markup language which defines the error page interface. • Login (login.html). This is a file that contain hypertext markup language which defines the login page interface. • Register (register.html). This is a file that contain hypertext markup language which defines the register page interface. • Banner (banner.png | banner.jpg). This is the project banner. (image file). • Category (category.html). This is a file that contain hypertext markup language which defines the blog post category page interface. • Post (post.html). This is a file that contain hypertext markup language which defines the post page interface (display post). • Page (page.html). This is a file that contain hypertext markup language which defines the blog page interface (display blog page).

Folder	Purpose

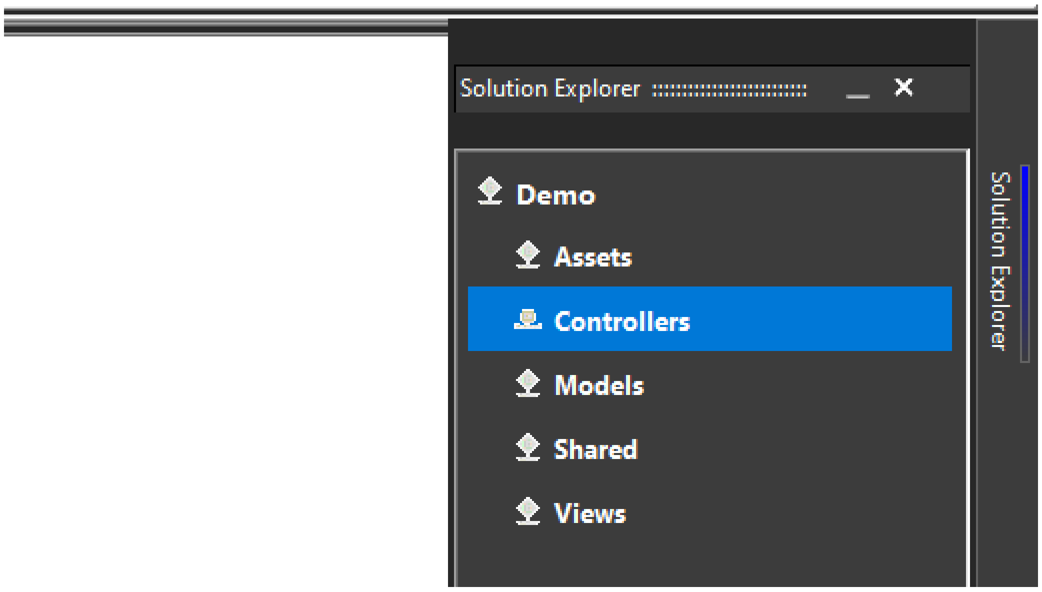
	• AppConfiguration (appconfig.json). This is a file that contain application configuration and can also use to store data by assigning key. It can be access using ConfigurationManager class.
	ConfigurationManager manager = new ConfigurationManager();
	object value = manager[“Key Name”];

Views	This is a folder that contains html file that is visible to the user. this is the visible part which user can access through web browser. The files serve as route on server. The file can be access by everyone on browser through domain/ route/ filename without extension. E.g. www.example.com/route/index. View is being replace with route on sever. View folder consist the following folders • **Admin.** This is the folder that contain the files which can be access only by users assigned the role of admin on server. It can be access on server by domain/route/admin/filename without extension. e.g. www.example.com/route/admin/index. • **Author.** This is the folder that contain the files which can be access only by users assigned the role of author on server. It can be access on server by domain/route/author/filename without extension. e.g. www.example.com/route/author/index. • **User.** This is the folder that contain the files which can be access only by users assigned the role of user on server. It can be access on server by domain/route/user/filename without extension. e.g. www.example.com/route/user/index. • **Auth.** This is the folder that contain the files which can be access only by authorized users on server. It can be access on server by domain/route/author/filename without extension. e.g. www.example.com/route/auth/index. **Note:** accessing folder on browser without specifying the filename will check for index.html file, if the index file not will redirect to the error page. E.g. accessing www.example.com/route or www.example.com/route/admin or www.example.com/route/author or

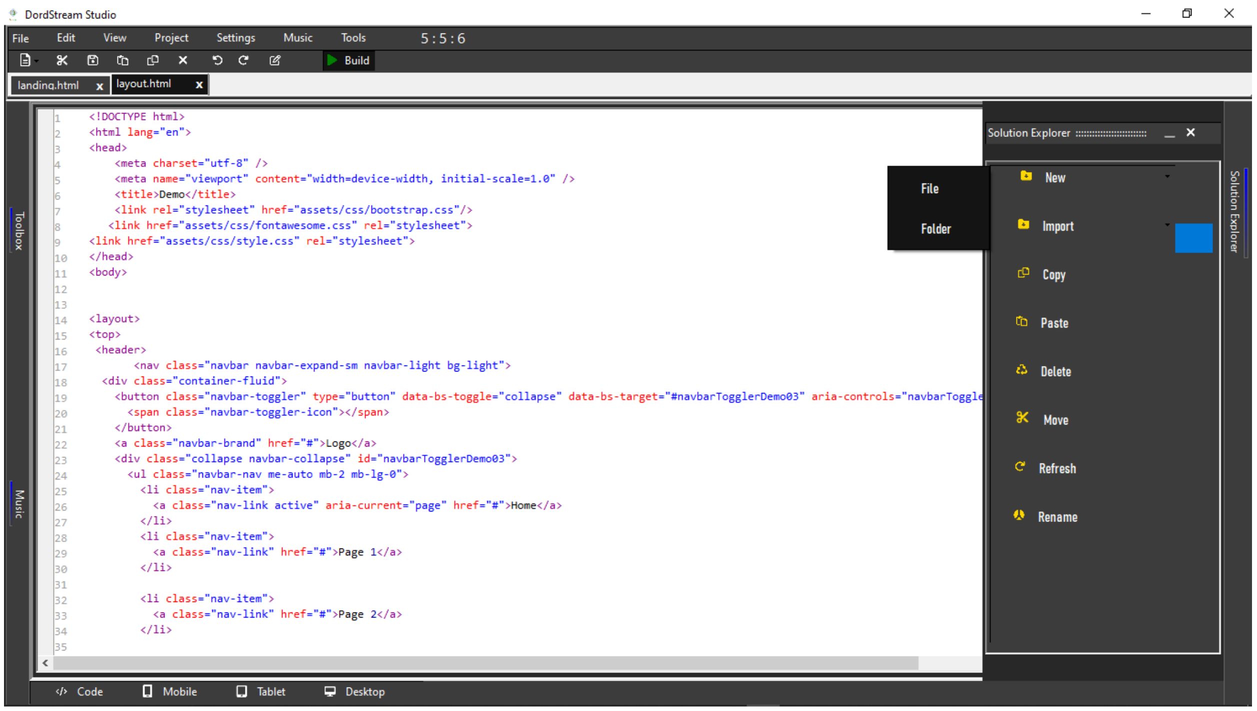
Folder	Purpose
	www.example.com/route/user or www.example.com/route/auth on browser will check for index.html and render on browser if exist or else redirect to error page.
Controllers	This is a folder that contain the file that has the application logic and file extension is .ds (DORDSTREAM). E.g. HomeController.ds.
Models	This is the folder that contain the JavaScript files which is used as data model. For transferring data from view to controller vise versa. The file extension is .js (JAVASCRIPT). E.g. HomeModel.js

Adding a Controller

In this section, you add controllers for managing the project

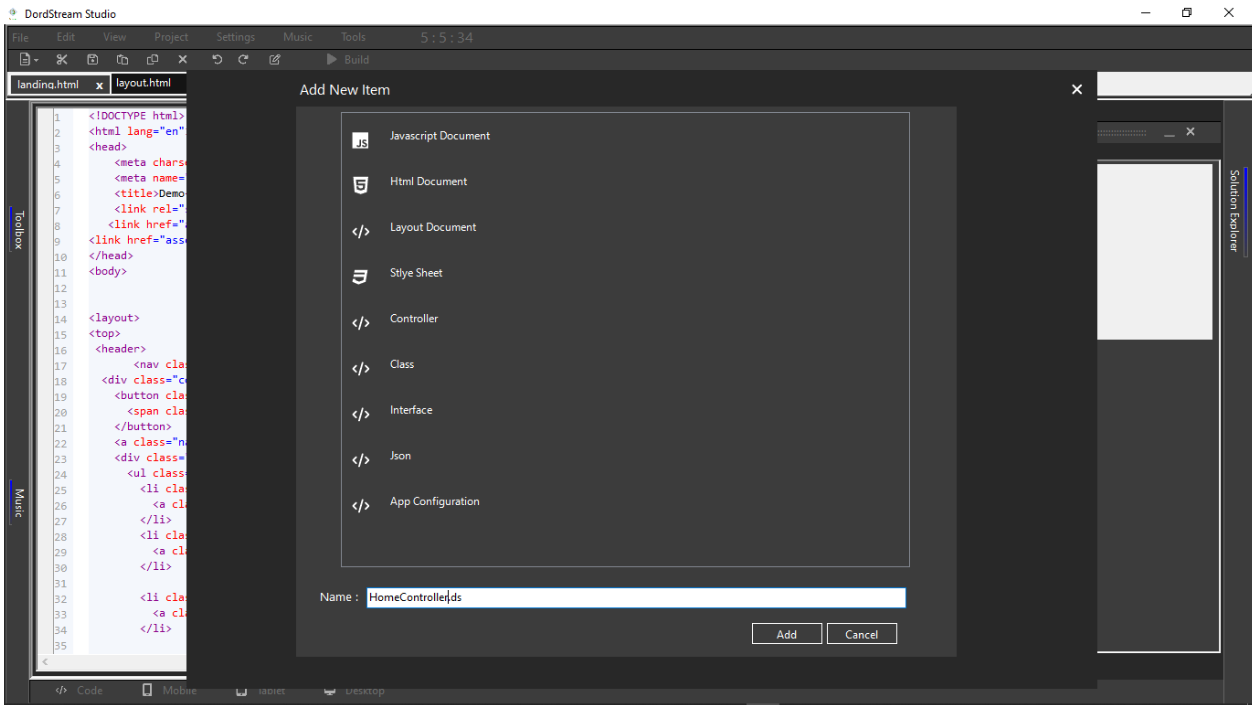


- Right click on Controllers then click **New** > **File**



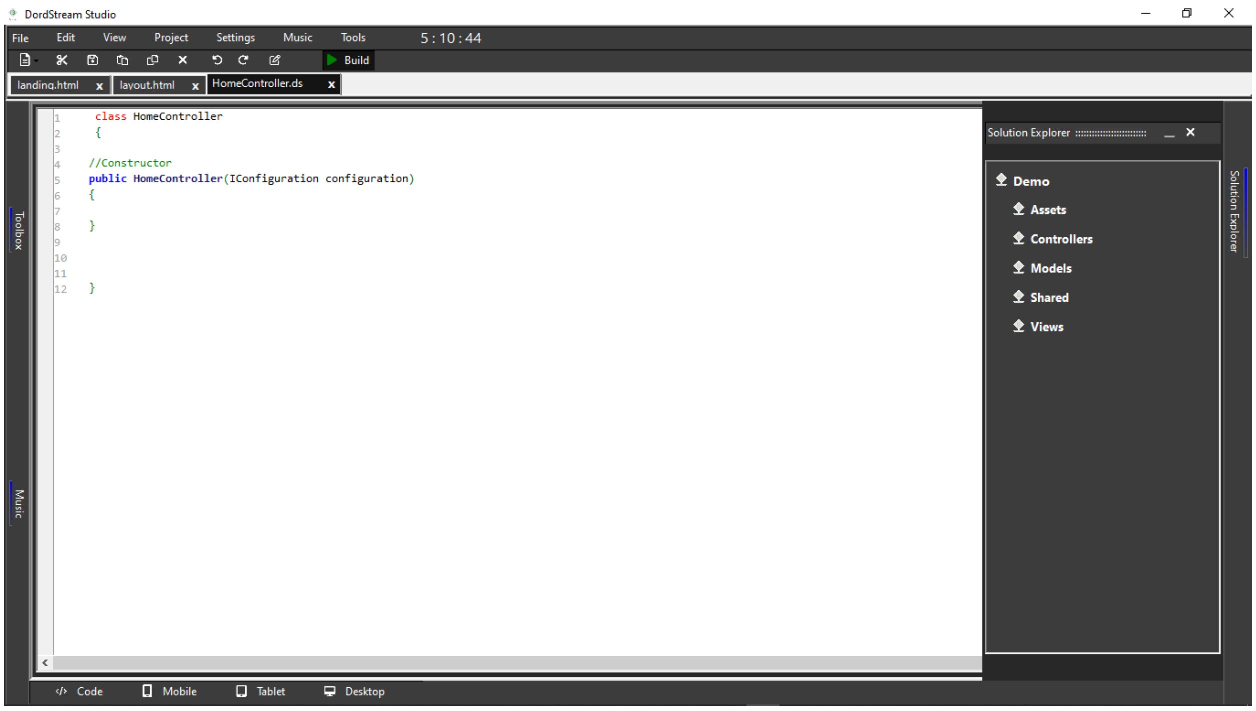
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Demo</title>
    <link rel="stylesheet" href="assets/css/bootstrap.css"/>
   <link href="assets/css/fontawesome.css" rel="stylesheet">
<link href="assets/css/style.css" rel="stylesheet">
</head>
<body>


<layout>
<top>
 <header>
      <nav class="navbar navbar-expand-sm navbar-light bg-light">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarTogglerDemo03" aria-controls="navbarToggle
      <span class="navbar-toggler-icon"></span>
    </button>
    <a class="navbar-brand" href="#">Logo</a>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo03">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Page 1</a>
        </li>

        <li class="nav-item">
          <a class="nav-link" href="#">Page 2</a>
        </li>

```

- Click on controller and specify the name. In this tutorial the controller will be named HomeController



Name : HomeController.ds

- Click on add to create HomeController.



```
 class HomeController
 {

//Constructor
public HomeController(IConfiguration configuration)
{

}



}
```

Generated program consists of three logical parts:
- Definition of a class **HomeController**;

- Definition of a Constructor **HomeController (IConfiguration)**;

Defining a Class

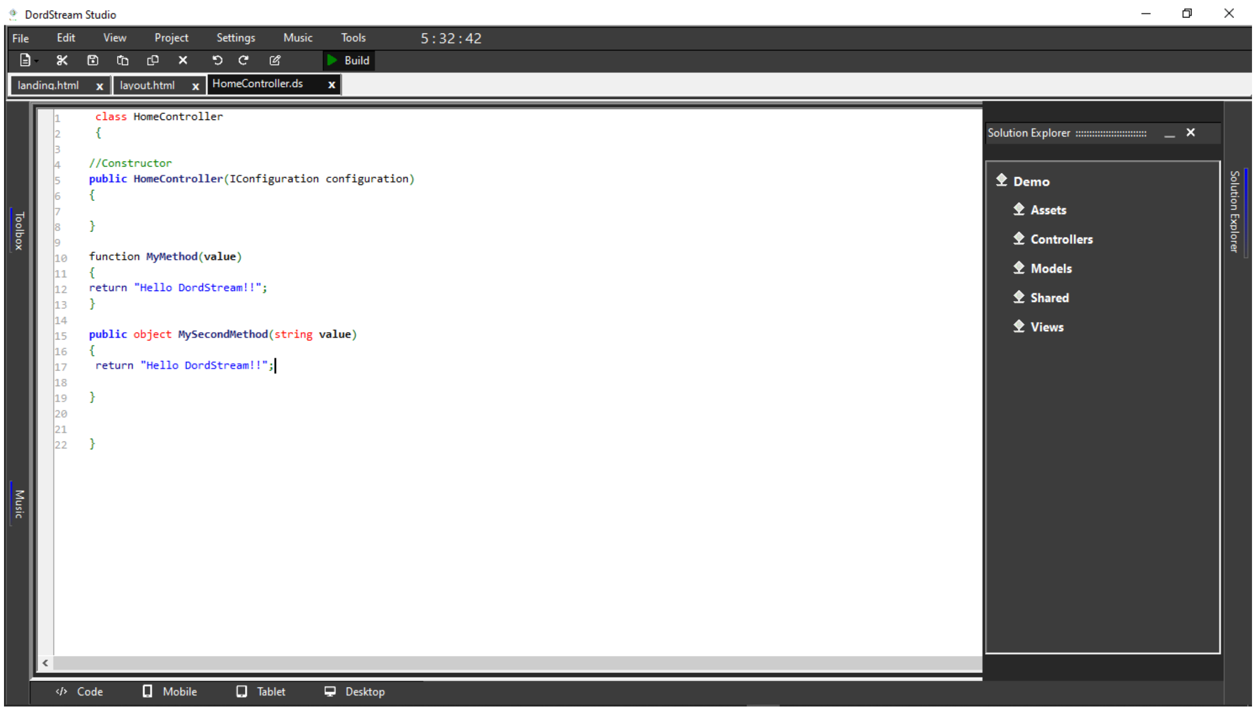
On the first line of our program we define a class called **HomeController.** The simplest definition of a class consists of the keyword **class**, followed by its name. In our case the name of the class is **HomeController**. The content of the class is located in a block of program lines, surrounded by curly brackets: **{}**.

Defining a Constructor

A constructor is a special method that is used to initialize objects. We define a constructor **HomeController** and having **IConfiguration** as a parameter or argument.

Adding a method to the generated program

let’s take a look at a simple example, illustrating how a **program a method written in C#** looks like:



```
1    class HomeController
2     {
3
4   //Constructor
5   public HomeController(IConfiguration configuration)
6   {
7
8   }
9
10  function MyMethod(value)
11  {
12  return "Hello DordStream!!";
13  }
14
15  public object MySecondMethod(string value)
16  {
17   return "Hello DordStream!!";
18
19  }
20
21
22  }
```

```
class HomeController
 {

//Constructor
public HomeController(IConfiguration configuration)
{

}

function MyMethod(value)
{
return "Hello DordStream!!";
}

public object MySecondMethod(string value)
{
 return "Hello DordStream!!";

}


}
```

the program above consists of two methods named MyMethod and MySecondMethod.

Defining the MyMethod(argument) Method

On the tenth line we define a method with the name **`MyMethod(value)`**, which is the starting point for our program. Every program written in C# starts from a **`MyMethod()`** method with the following title (signature):

```
function MyMethod(value);
```

The method must be declared as shown above, the method was declared using function keyword and has one parameter of type **value**. which has dynamic type. Using function keyword in dordstream controller will return dynamic type.

Defining the MySecondMethod(argument) Method

The method must be declared as shown above, the method was declared using object keyword and has one parameter of type **value**. which has string type.

```
public object MySecondMethod(string value)
```

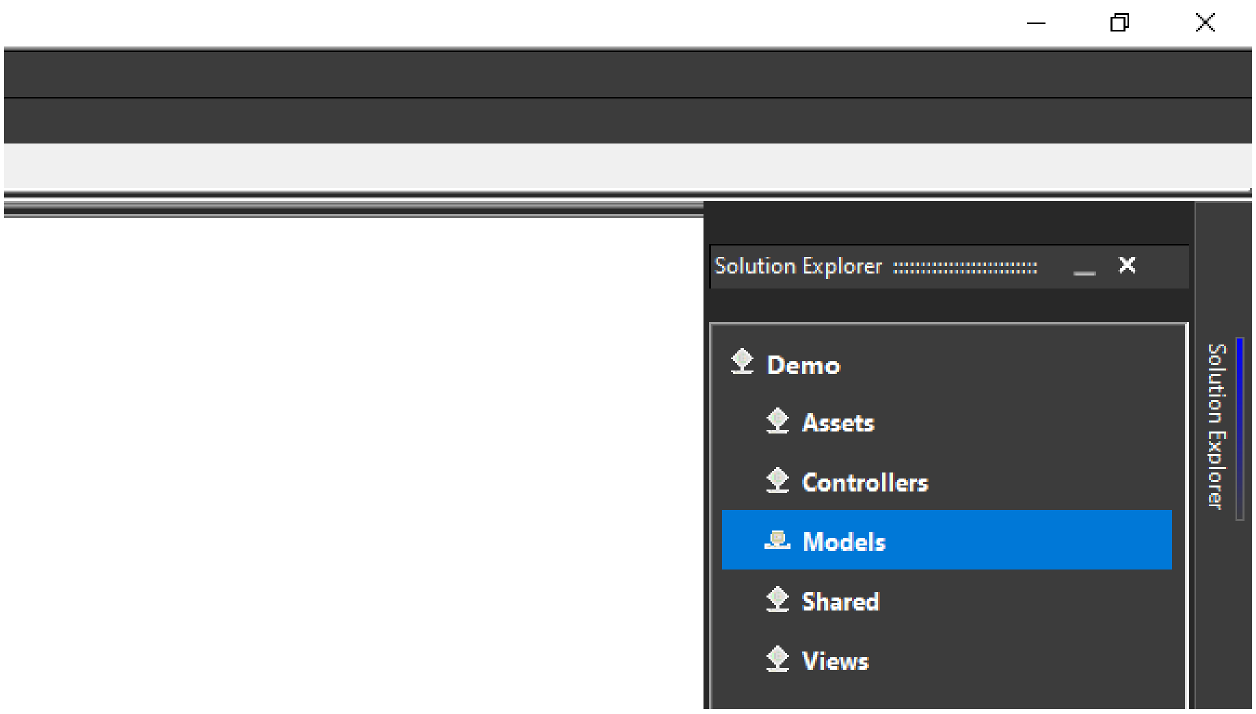
Contents of the MyMethod(argument) and MySecondMethod(argument) Method

The content of every method is found after its signature, surrounded by opening and closing curly brackets. On the next line of our sample program we use return to print a message on browser, in this case `"Hello DordStream!!”`.

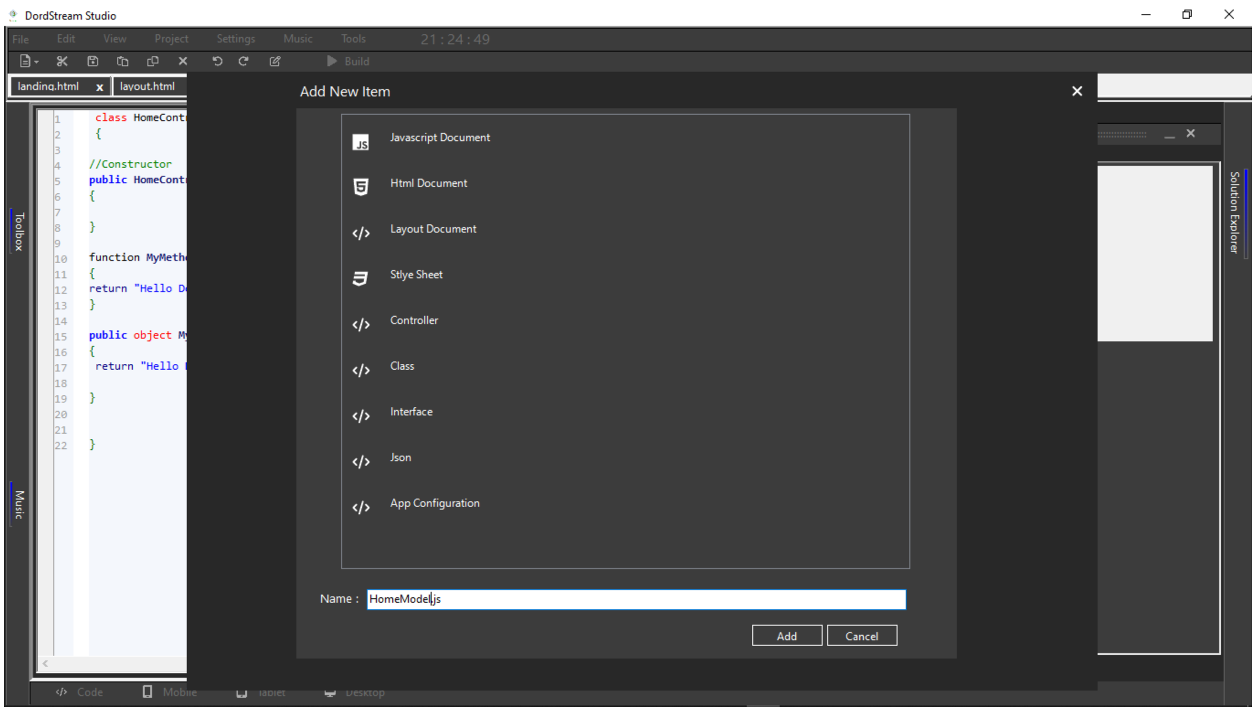
Click on Build to compile the project and debug it through browser.

Adding a Model

In this section, you add JavaScript to communicate with the controller or transfer data from the controller to the view vice versa.



- Right click on Models then click **New** > **File**

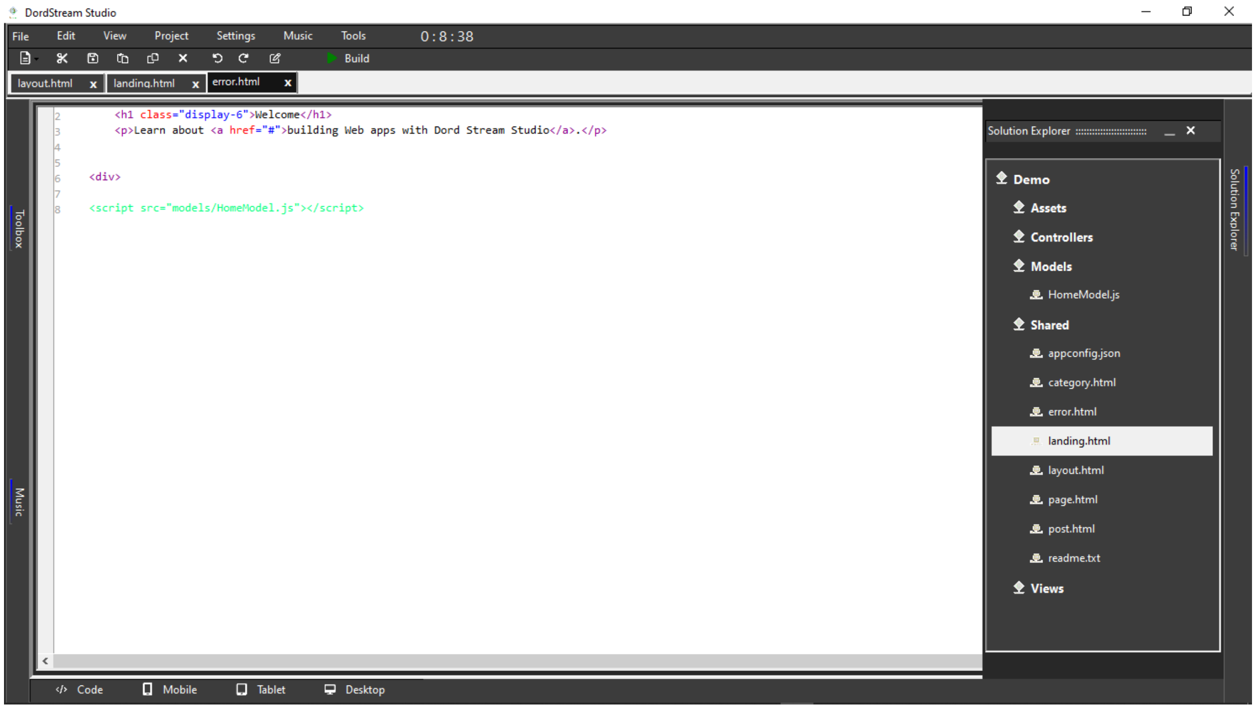


Name : HomeModel.js

```
class HomeCont
{

//Constructor
public HomeCont
{

}

function MyMeth
{
return "Hello D
}

public object M
{
 return "Hello

}


}
```

- Click on Javascript Document and specify the name. In this tutorial the controller will be named HomeModel
- Click on add to create HomeModel.

- It will create an empty file by default.
- Link the HomeModel to a view.in this tutorial we are linking HomeModel to landing.html.
- Add `<script src="models/HomeModel.js"></script>.` To landing.html



```
        <h1 class="display-6">Welcome</h1>
        <p>Learn about <a href="#">building Web apps with Dord Stream Studio</a>.</p>


    <div>

    <script src="models/HomeModel.js"></script>
```

Add the following codes to the model file HomeModel.js

```
var homeModel = new ControllerConnection("HomeController");

homeModel.start().then(function(){

homeModel.invoke("MyMethod","").then(function(result){
alert(result);
}).catch(function(error){
alert(error);
});

}).catch(function(error){
alert(error);
});
```

Explanations of the above codes

ControllerConnection is a function that connect to the controller. It takes one argument.

The argument is the Controller name, in this tutorial we created HomeController.

```
var homeModel = new ControllerConnection("HomeController");
```

homeModel has been assigned to ControllerConnection which we can invoke the method in the Controller using invoke method.

Before invoking any method in Controller, the connection must be in Connected State. i.e. start() must be called after creating the Controller instance.

```
homeModel.start();
```

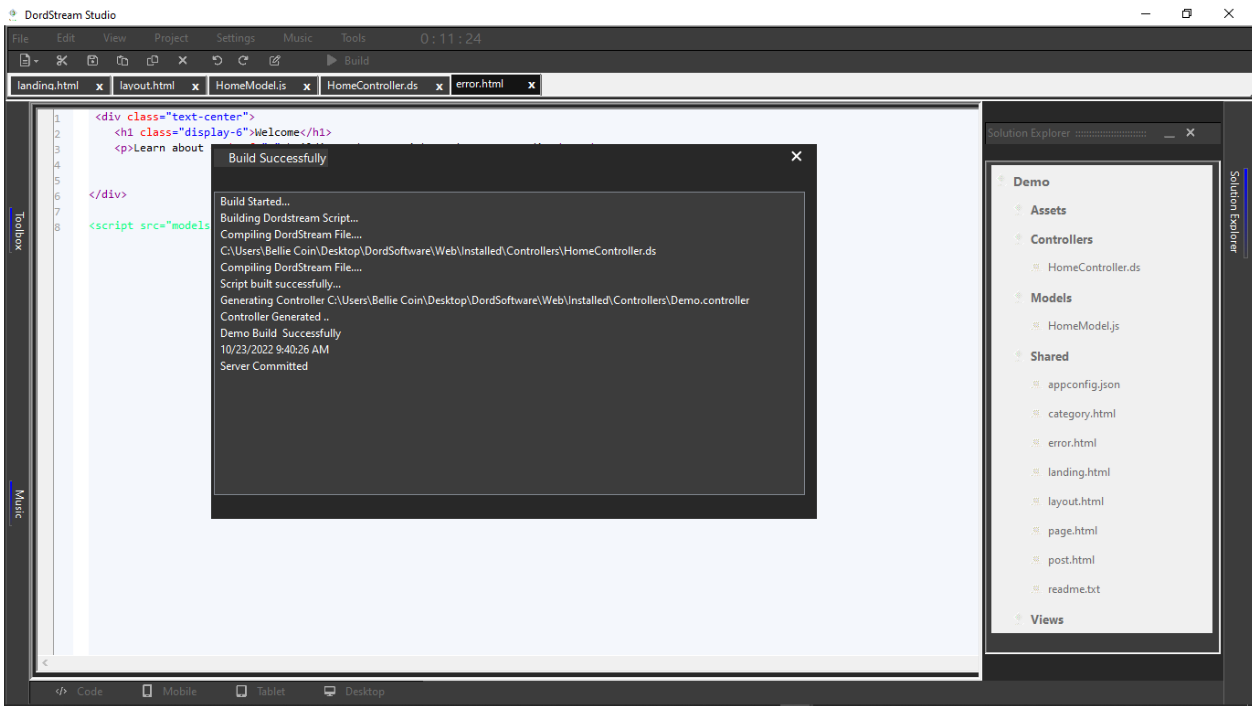
invoke method is use to controller execute method and assign parameters.

```
ControllerConnection.invoke("Method Name",argument..);
```

```
homeModel.invoke("MyMethod","");
```

MyMethod is the method name in HomeController.

To Debug it. Click on Build to compile the Code.

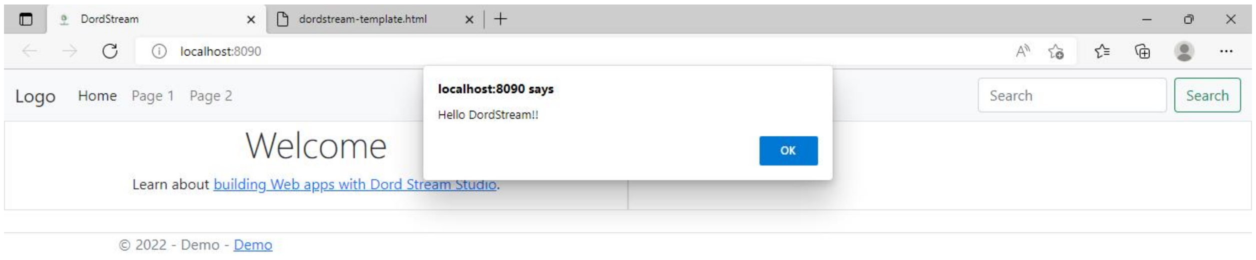


```
<div class="text-center">
   <h1 class="display-6">Welcome</h1>
   <p>Learn about


</div>

<script src="models
```

Build Successfully

```
Build Started...
Building Dordstream Script...
Compiling DordStream File....
C:\Users\Bellie Coin\Desktop\DordSoftware\Web\Installed\Controllers\HomeController.ds
Compiling DordStream File....
Script built successfully...
Generating Controller C:\Users\Bellie Coin\Desktop\DordSoftware\Web\Installed\Controllers\Demo.controller
Controller Generated ..
Demo Build  Successfully
10/23/2022 9:40:26 AM
Server Committed
```

Run the browser to display the result.



localhost:8090 says

Hello DordStream!!

OK

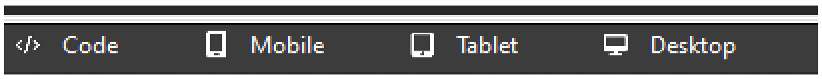
Welcome

Learn about building Web apps with Dord Stream Studio.

© 2022 - Demo - Demo

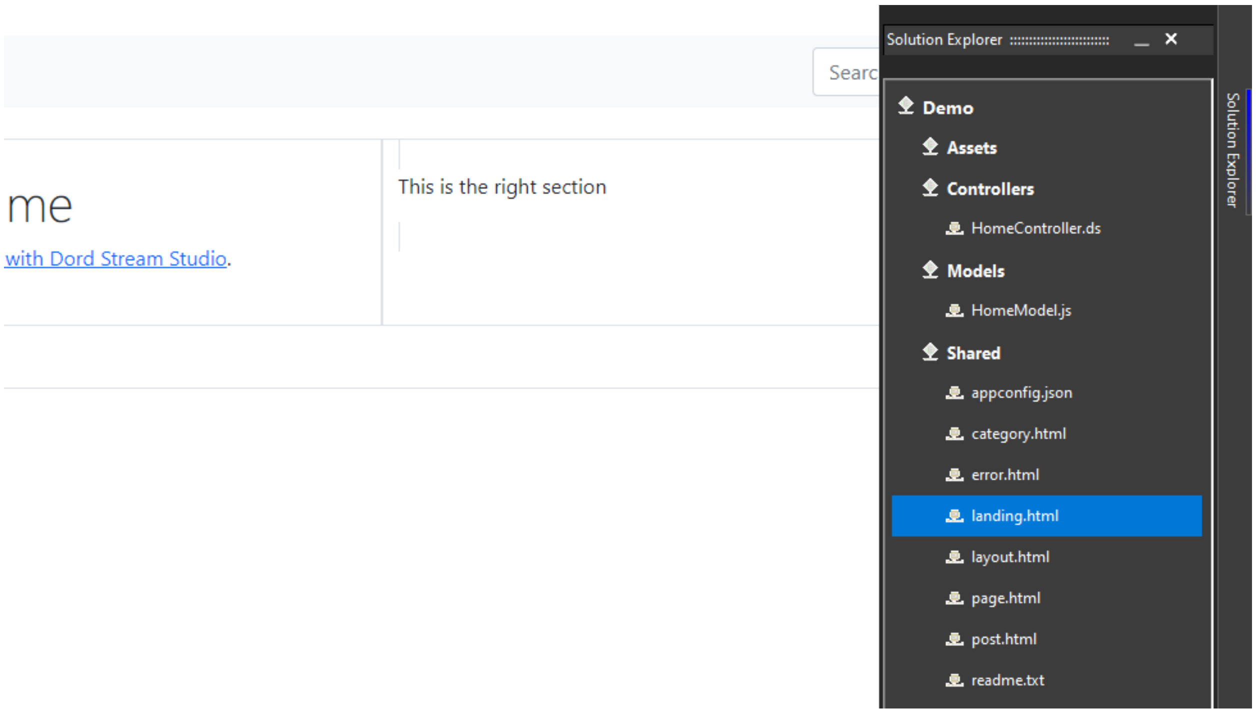
Auto Generate Controller and Model

In this section, you will learn how to auto generate controller and model

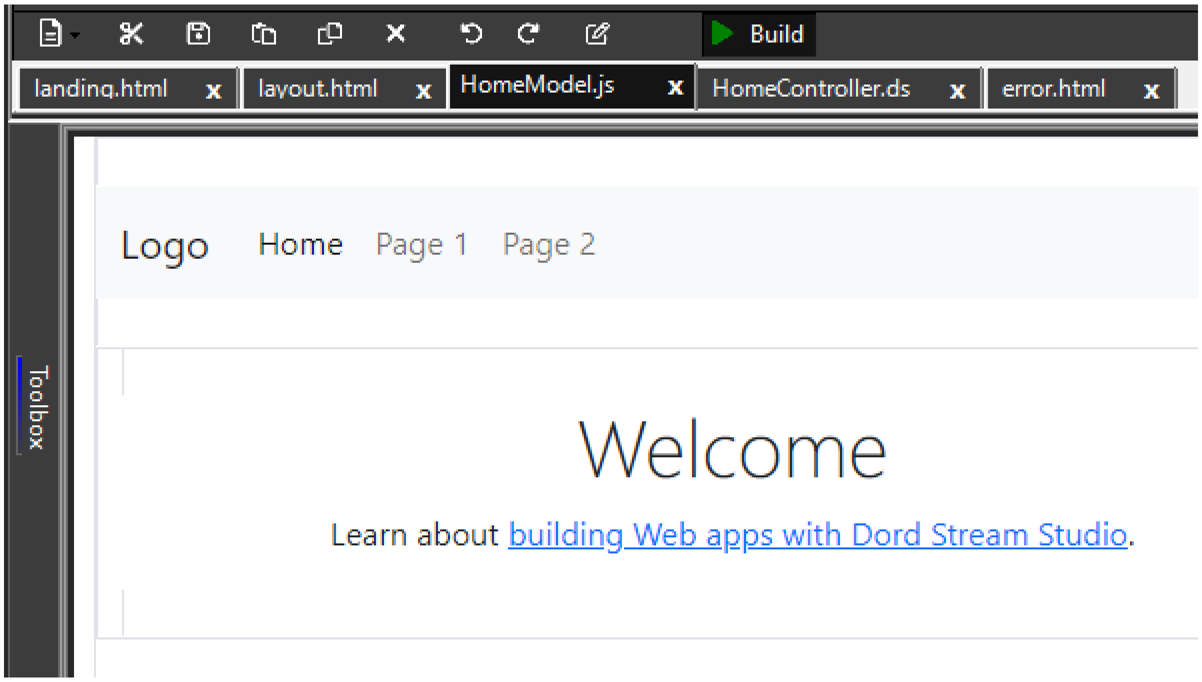


- At the bottom section of the application click on **Desktop**

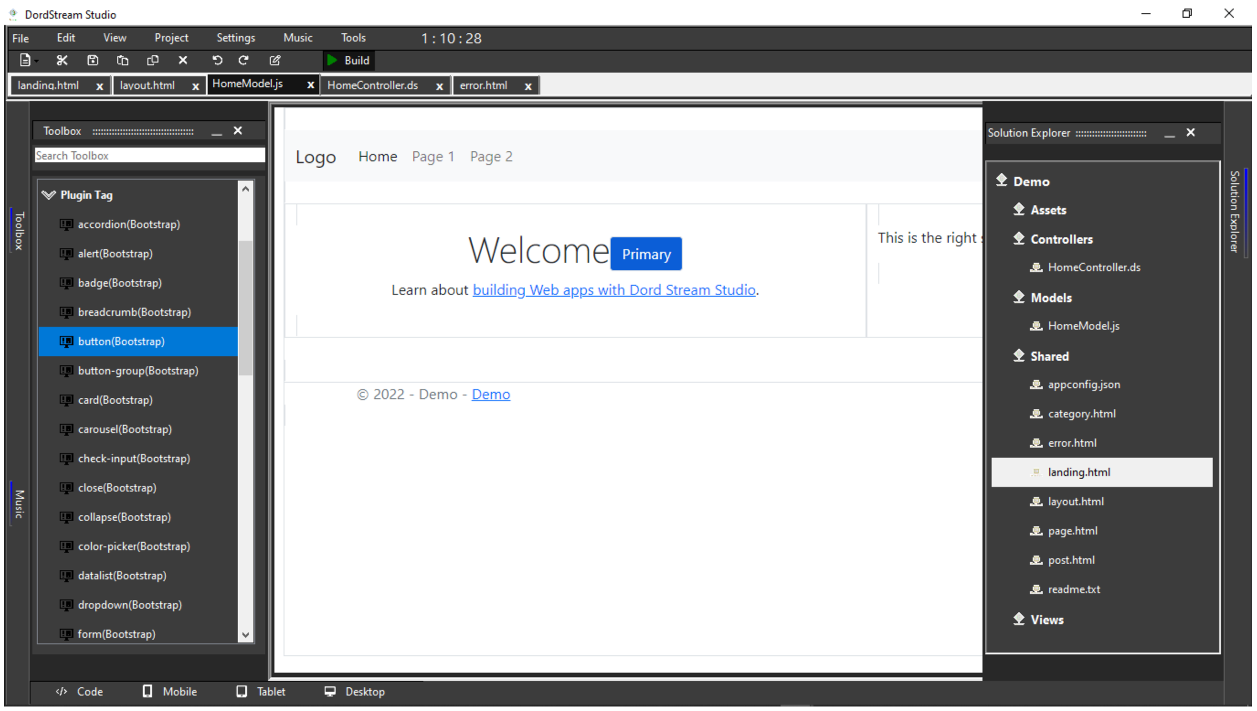
- In this tutorial we are using landing.html page.



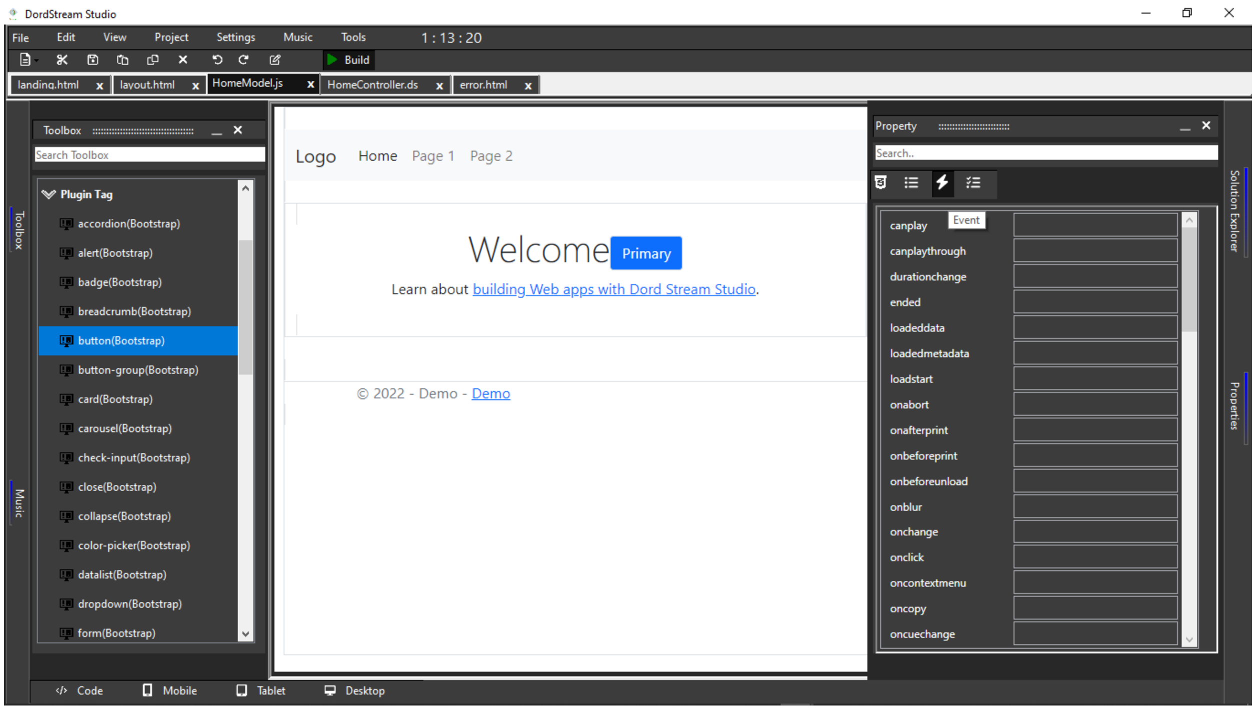
- Click on **landing.html**



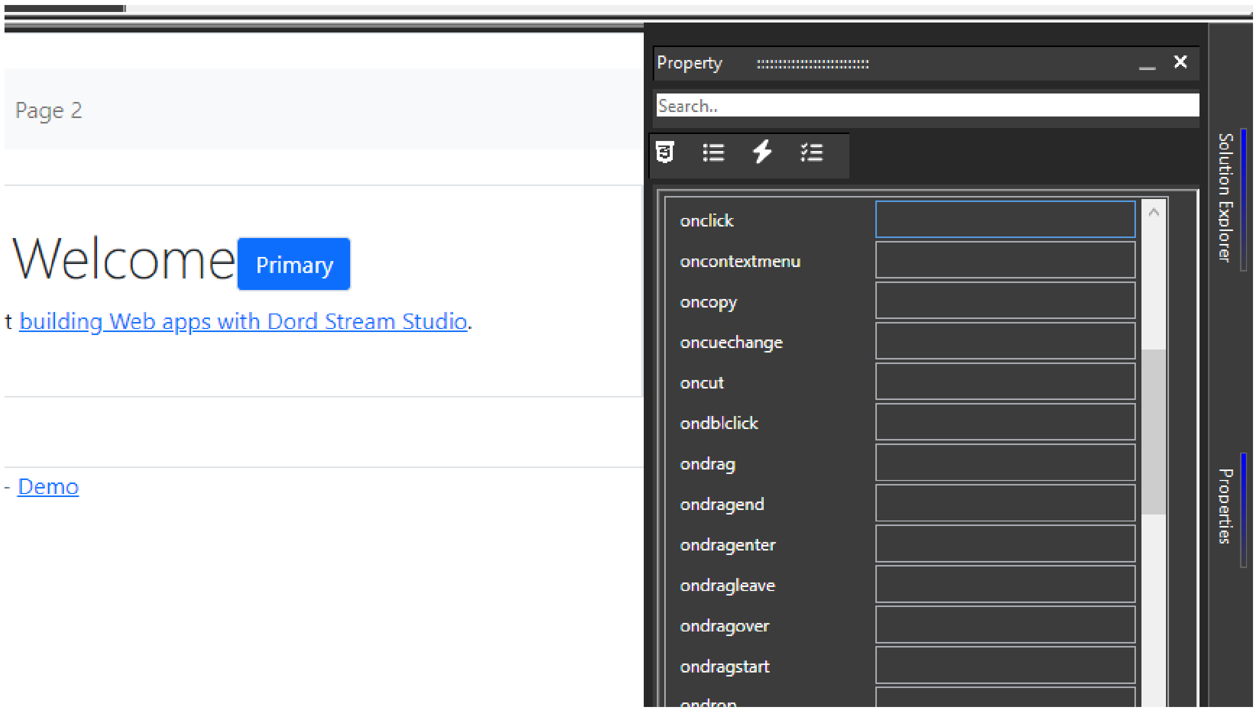
- At the left section of the application click on **Toolbox**



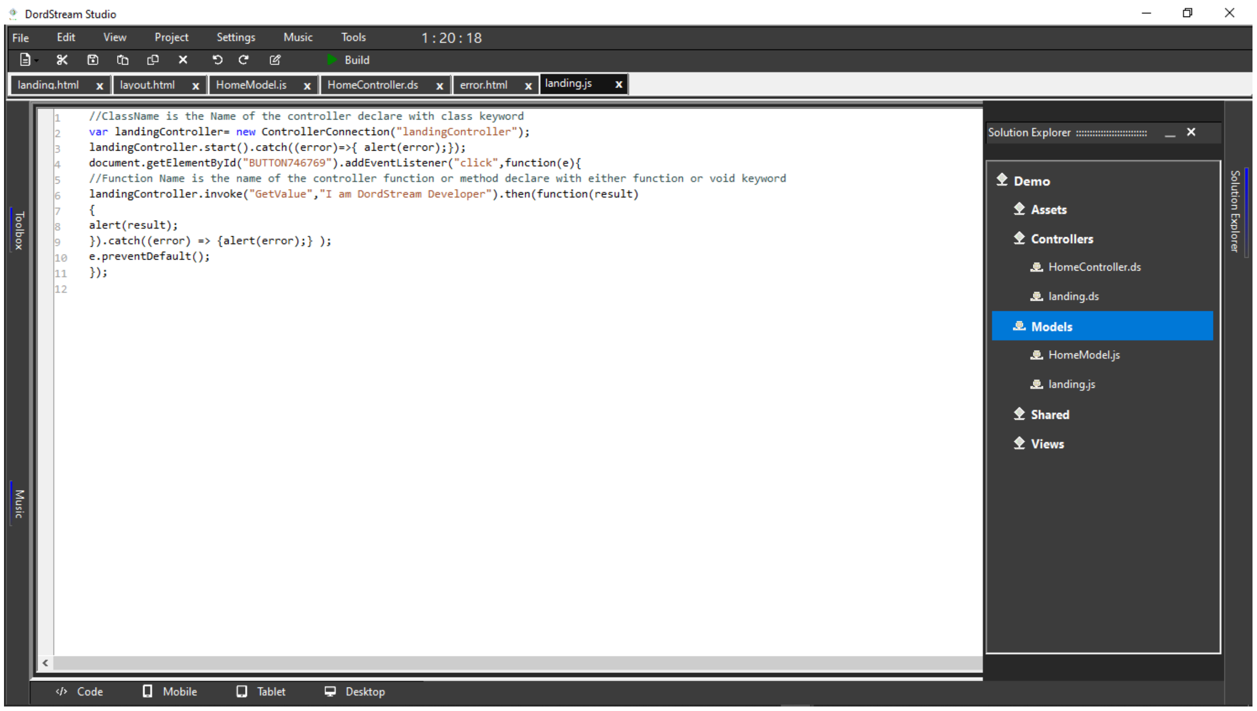
- Click on Plugin Tag to Expand the control then drag and drop **button(Bootstrap)** on browser.



- Double click on the dragged **button(Primary)** then, at property control click on event icon
- We want to an event to trigger from the controller when user clicked on the button

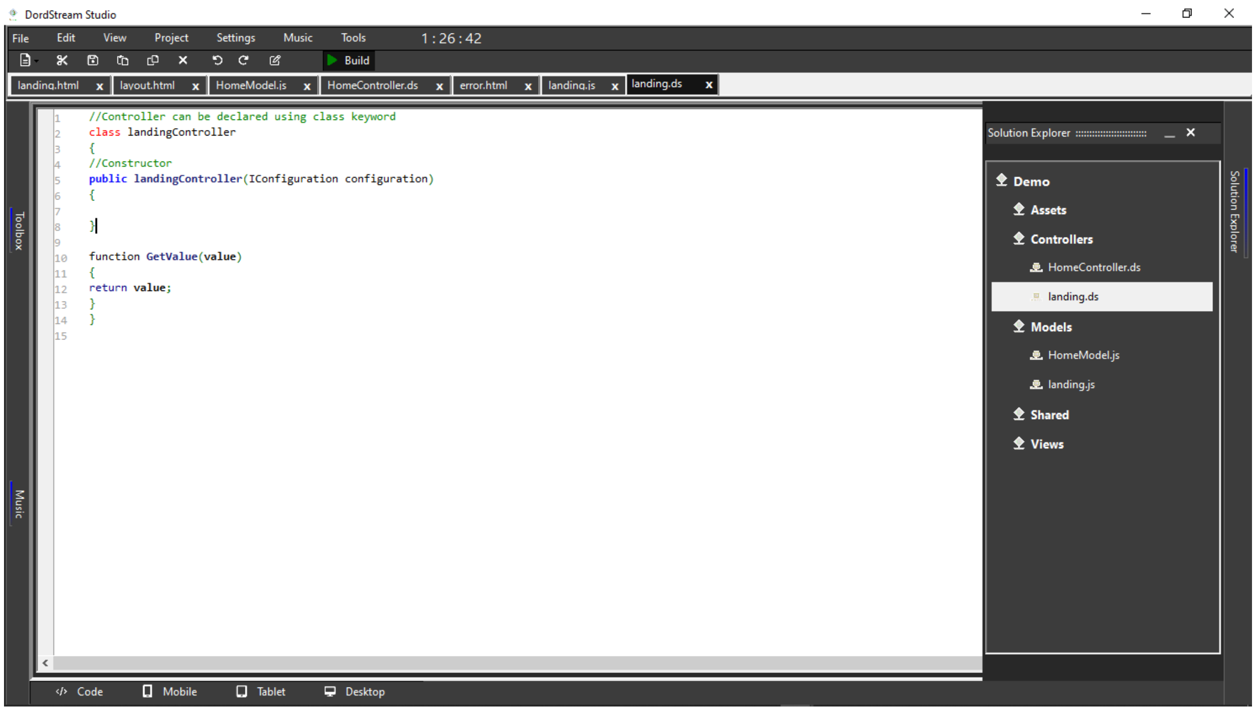


- Double click on **onclick textbox** to generate the controller and model.



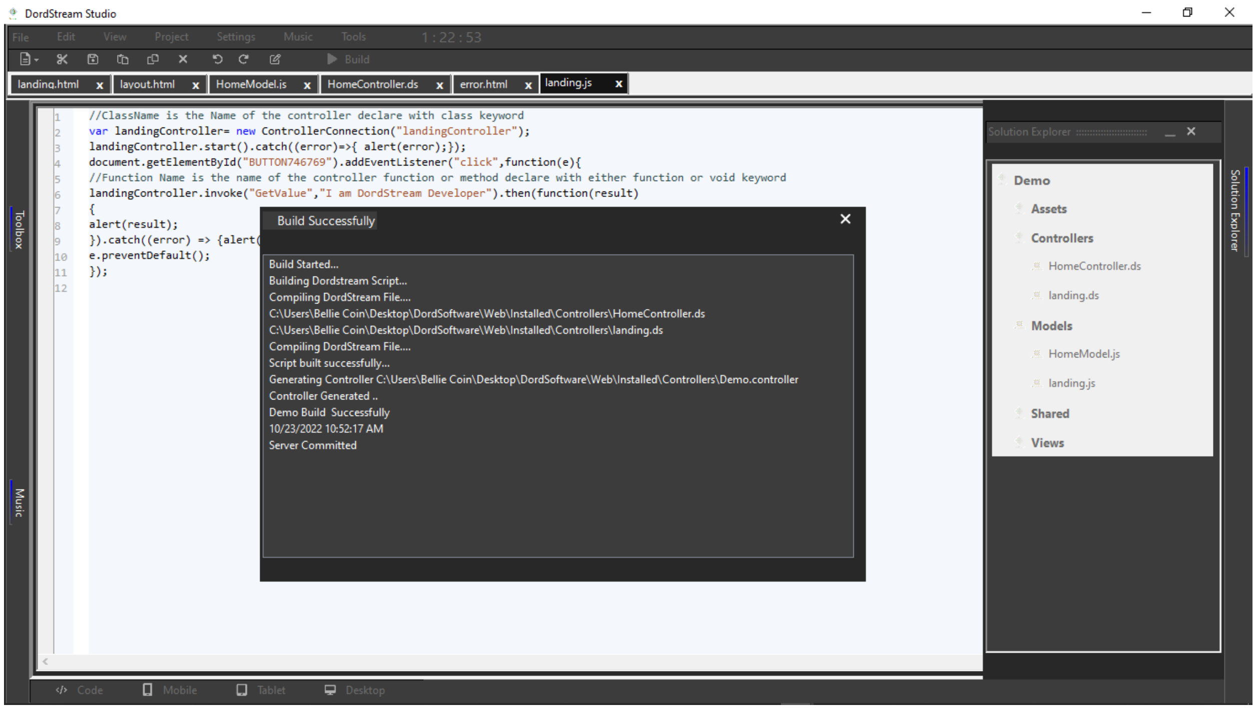
```
//ClassName is the Name of the controller declare with class keyword
var landingController= new ControllerConnection("landingController");
landingController.start().catch((error)=>{ alert(error);});
document.getElementById("BUTTON746769").addEventListener("click",function(e){
//Function Name is the name of the controller function or method declare with either function or void keyword
landingController.invoke("GetValue","I am DordStream Developer").then(function(result)
{
alert(result);
}).catch((error) => {alert(error);} );
e.preventDefault();
});
```

- Controller and Model will be generated with the file name. In the picture above a controller named landing.ds and a model named landing.js generated by default.



```
//Controller can be declared using class keyword
class landingController
{
//Constructor
public landingController(IConfiguration configuration)
{

}

function GetValue(value)
{
return value;
}
}
```

- Click Build button to compile the codes.

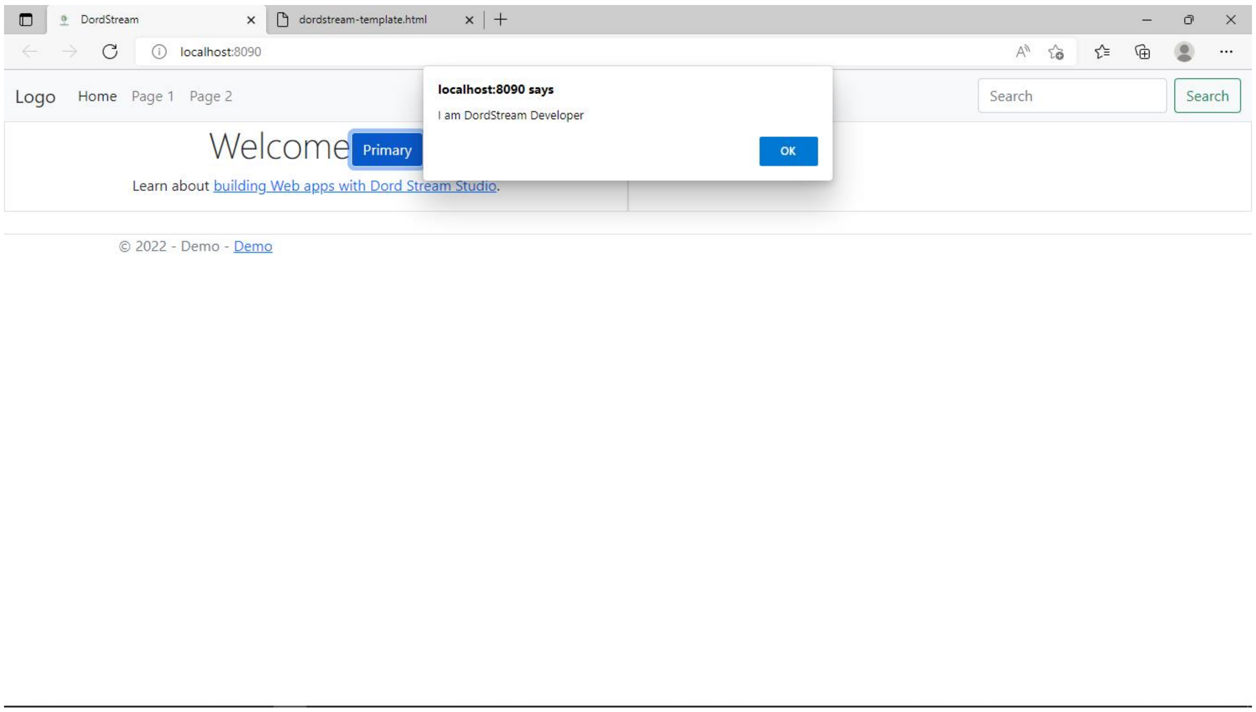


```
//ClassName is the Name of the controller declare with class keyword
var landingController= new ControllerConnection("landingController");
landingController.start().catch((error)=>{ alert(error);});
document.getElementById("BUTTON746769").addEventListener("click",function(e){
//Function Name is the name of the controller function or method declare with either function or void keyword
landingController.invoke("GetValue","I am DordStream Developer").then(function(result)
{
alert(result);
}).catch((error) => {alert(
e.preventDefault();
});
```

Build Successfully

```
Build Started...
Building Dordstream Script...
Compiling DordStream File....
C:\Users\Bellie Coin\Desktop\DordSoftware\Web\Installed\Controllers\HomeController.ds
C:\Users\Bellie Coin\Desktop\DordSoftware\Web\Installed\Controllers\landing.ds
Compiling DordStream File....
Script built successfully...
Generating Controller C:\Users\Bellie Coin\Desktop\DordSoftware\Web\Installed\Controllers\Demo.controller
Controller Generated ..
Demo Build  Successfully
10/23/2022 10:52:17 AM
Server Committed
```

- Debug it through Browser or Refresh your browser
- Click on the **button(Primary).**



localhost:8090 says

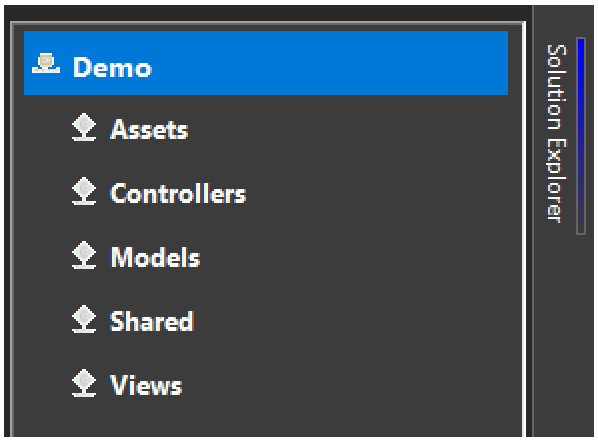
I am DordStream Developer

Learn about building Web apps with Dord Stream Studio.

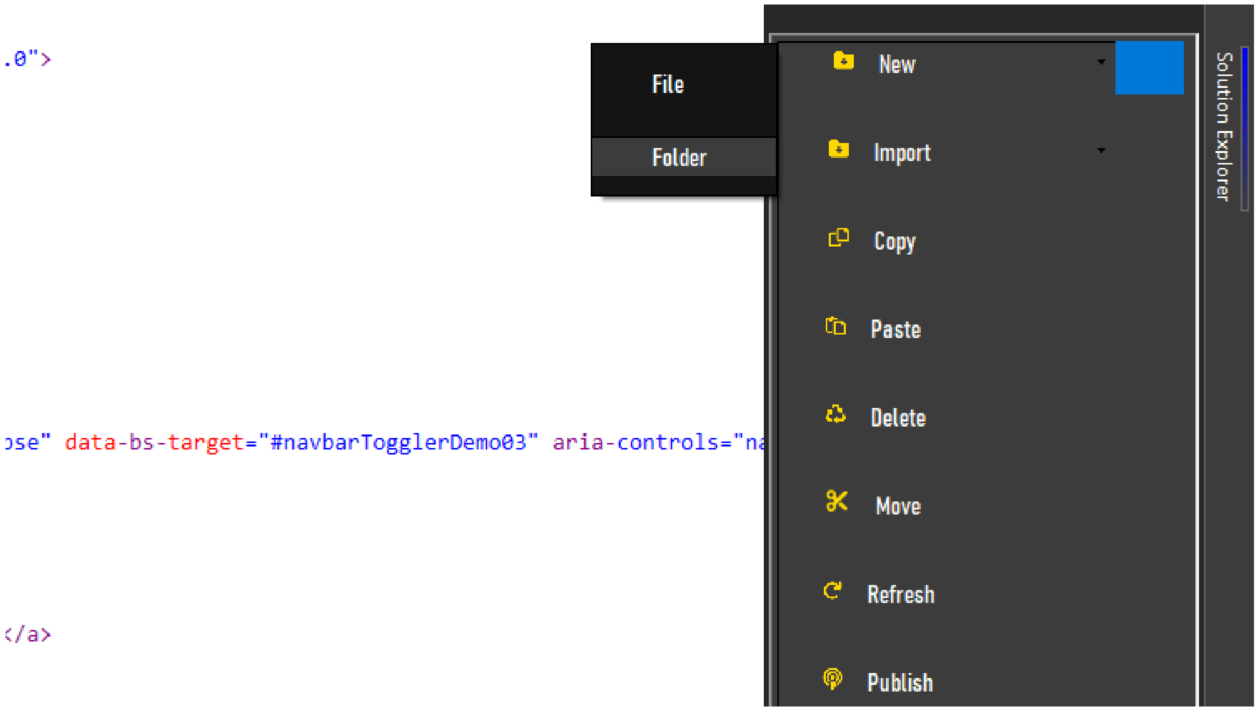
© 2022 - Demo - Demo

Adding a Class

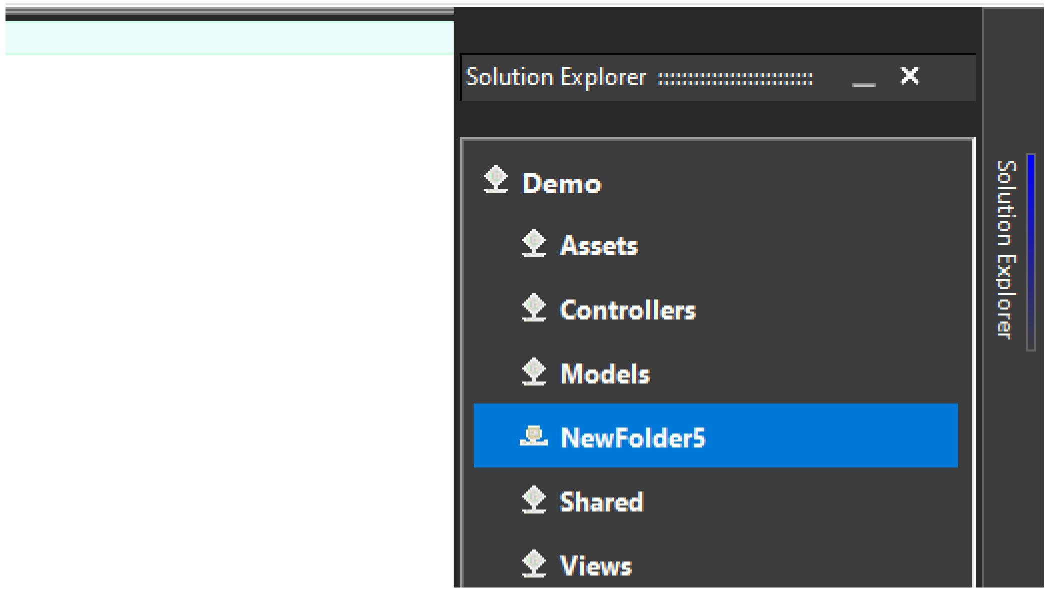
In this section, you add and reuse classes for managing the project



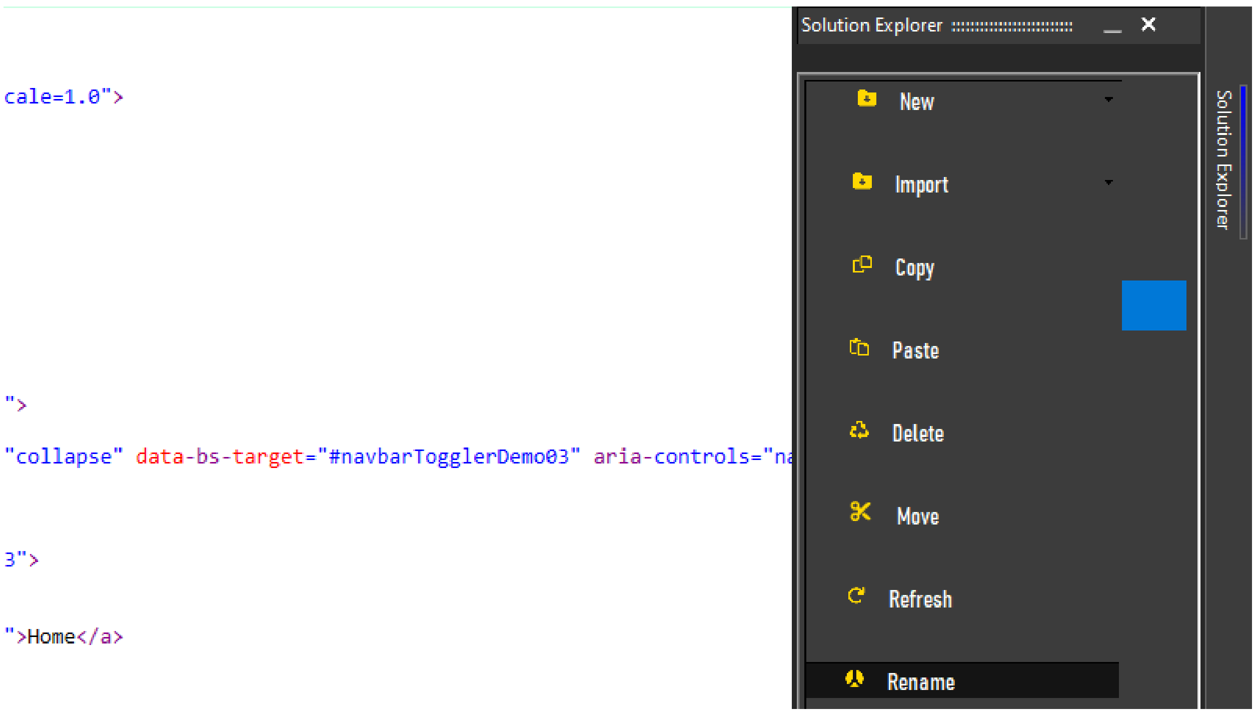
- Create a folder that will contain the class file
- Right click on the project name then click **New** > **Folder**



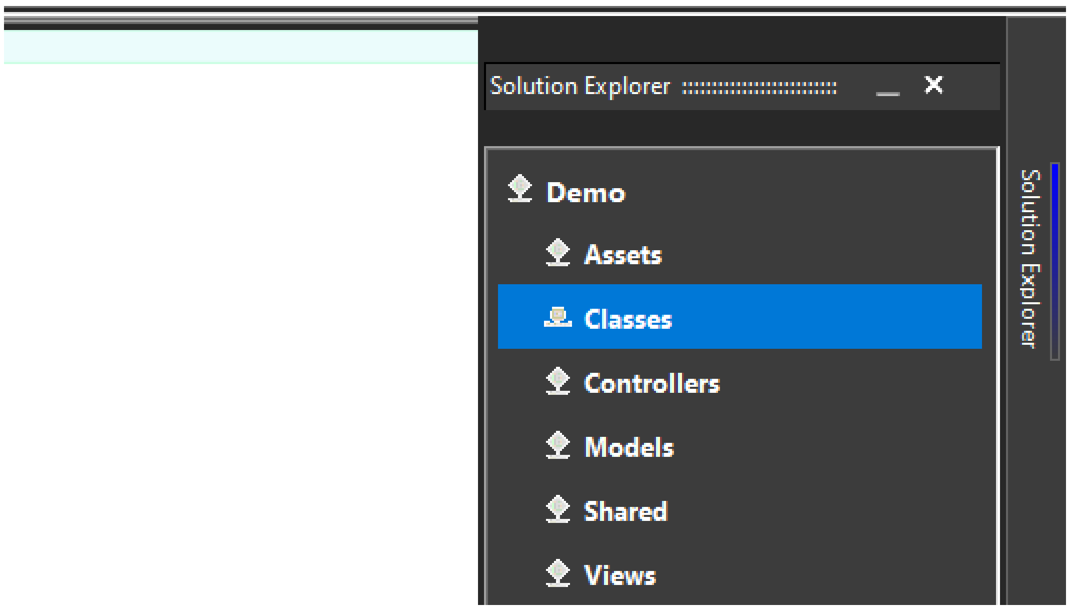
```
.0">
ose" data-bs-target="#navbarTogglerDemo03" aria-controls="na
</a>
```



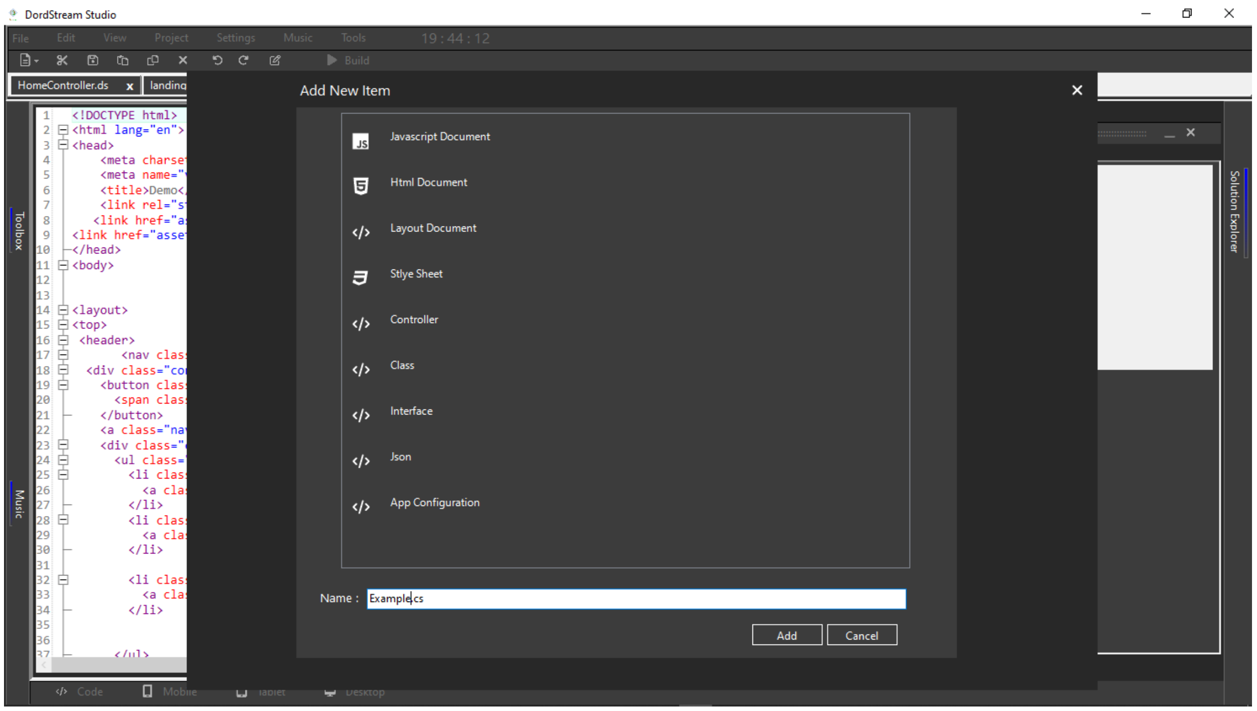
- A new folder with random number will be created by default.
- Rename the folder to the desired name. In this tutorial we will rename the folder as **Classes**.


```
cale=1.0">

">
"collapse" data-bs-target="#navbarTogglerDemo03" aria-controls="na

3">

">Home</a>
```

- Right click on the folder and click on **Rename**

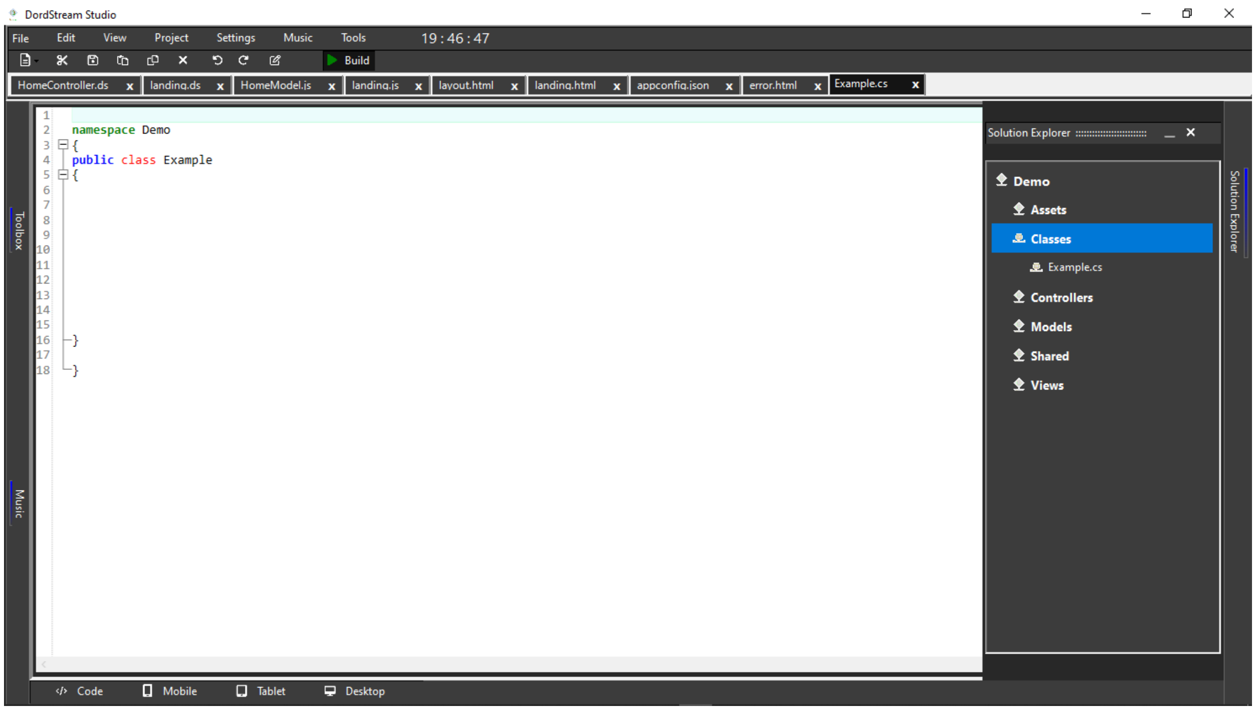


- After renaming it press **Enter key**
- Create a class file into the **Classes** folder
- Right click on the Classes folder then click **New** > **File**



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charse
    <meta name="
    <title>Demo<
    <link rel="s
   <link href="a
<link href="asse
</head>
<body>


<layout>
<top>
 <header>
       <nav clas
  <div class="co
    <button clas
      <span clas
    </button>
    <a class="na
    <div class="
      <ul class=
        <li clas
          <a cla
        </li>
        <li clas
          <a cla
        </li>

        <li clas
          <a cla
        </li>


      </ul>
```

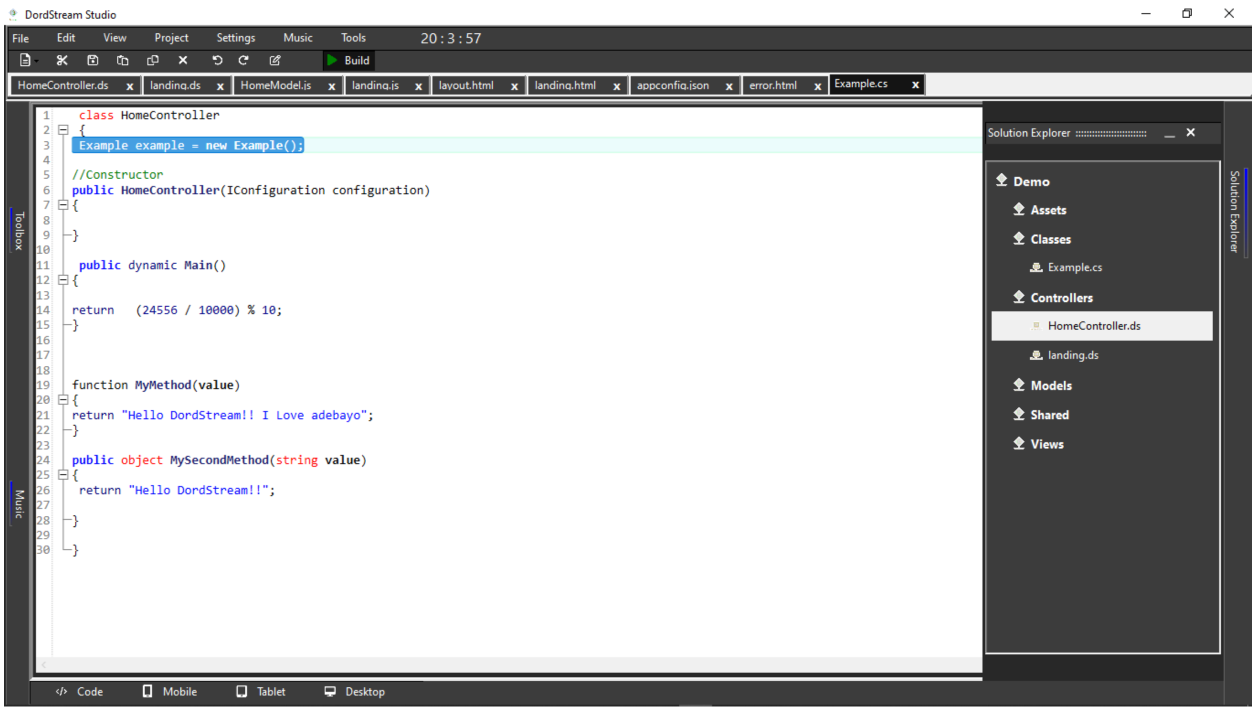
- Click on class and give it a name. In this tutorial we are naming it Example.cs
- Click on Add to create the Class.



```

namespace Demo
{
public class Example
{










}

}
```

- A default class will be generated. [To Learn How to create a class in C#](#)

- The class can be in a controller by creating an instance of the class

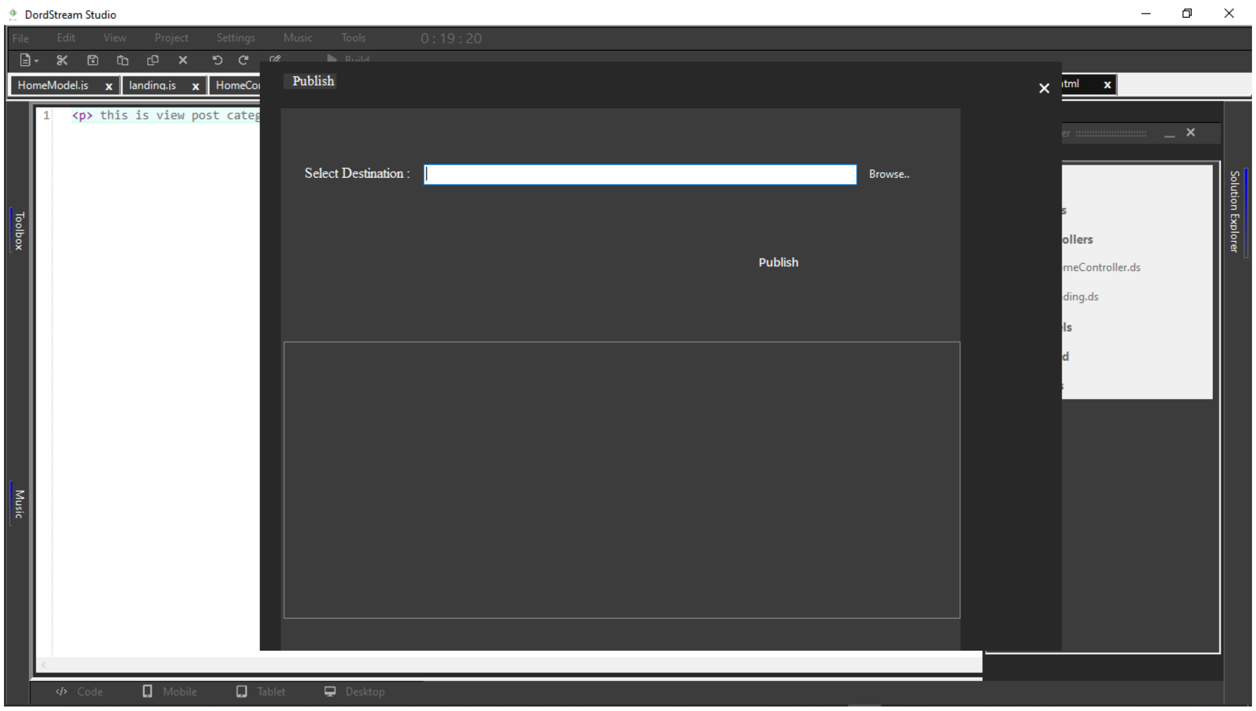


```
 class HomeController
 {
Example example = new Example();

//Constructor
public HomeController(IConfiguration configuration)
{

}

 public dynamic Main()
{

return   (24556 / 10000) % 10;
}



function MyMethod(value)
{
return "Hello DordStream!! I Love adebayo";
}

public object MySecondMethod(string value)
{
 return "Hello DordStream!!";

}

}
```

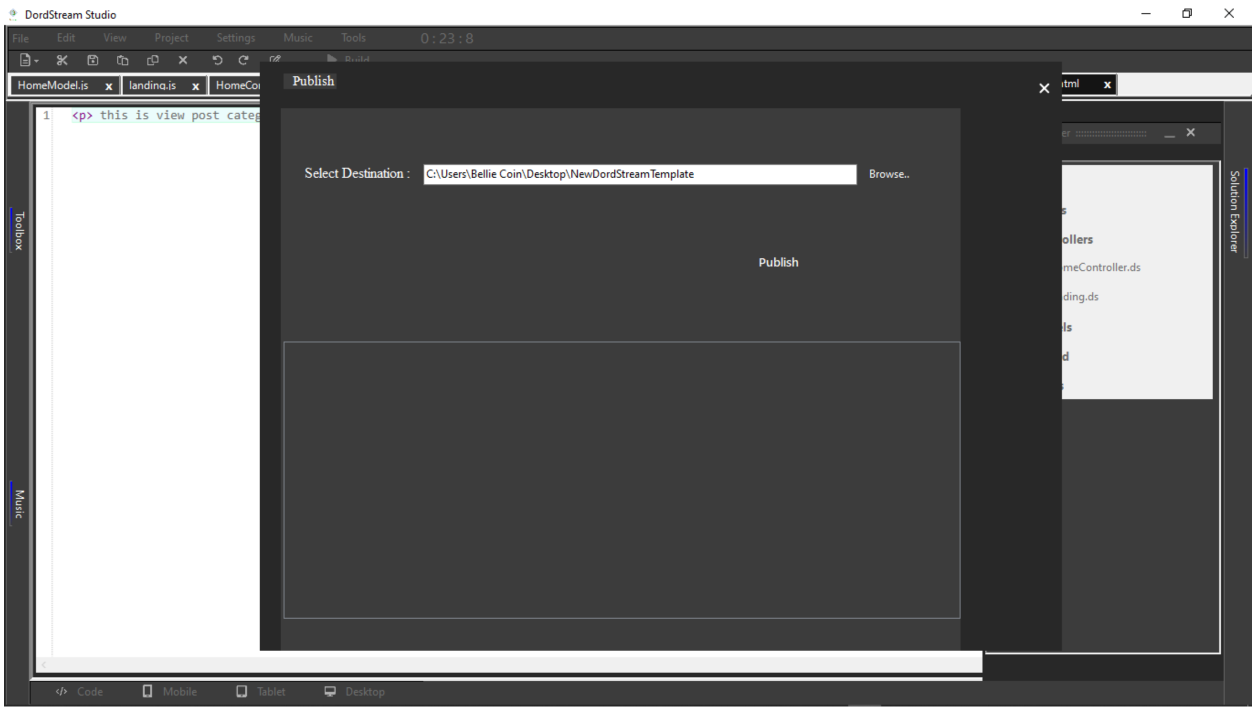
Publish Project

This is the final section where your project is ready for production.

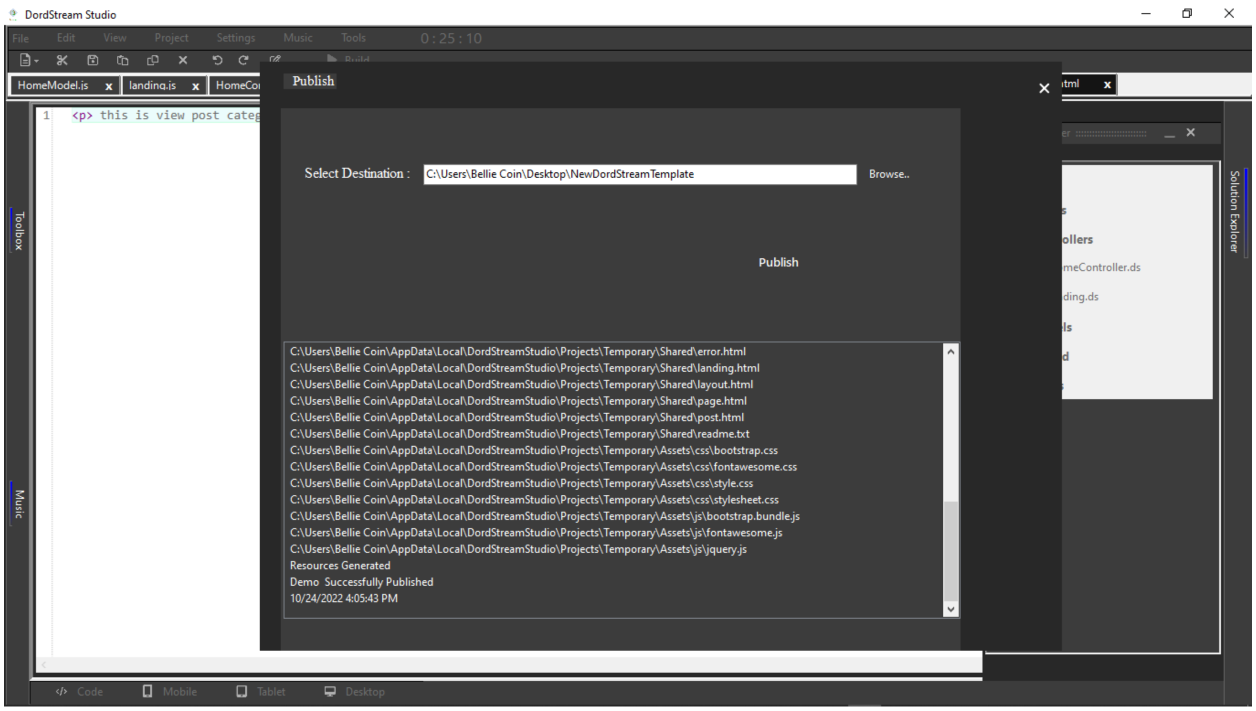
From the DordStream Studio click **Project** menu.


DordStream Studio
File Edit View Project Settings Music Tools 0 : 19 : 20
HomeModel.js landing.js HomeCo
Publish
Select Destination :
Browse..
Publish
Toolbox
Music
Solution Explorer
Code Mobile Tablet Desktop

- Click on **Browse** to select the project destination i.e. the location to save the generated file.


DordStream Studio
File Edit View Project Settings Music Tools 0 : 23 : 8
HomeModel.js landing.js HomeCo
Publish
Select Destination :
C:\Users\Bellie Coin\Desktop\NewDordStreamTemplate
Browse..
Publish
Toolbox
Music
Solution Explorer
Code Mobile Tablet Desktop

- Click on **Publish** to compile and build the project i.e. ready for production.



Select Destination : C:\Users\Bellie Coin\Desktop\NewDordStreamTemplate

```
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Shared\error.html
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Shared\landing.html
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Shared\layout.html
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Shared\page.html
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Shared\post.html
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Shared\readme.txt
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\css\bootstrap.css
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\css\fontawesome.css
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\css\style.css
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\css\stylesheet.css
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\js\bootstrap.bundle.js
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\js\fontawesome.js
C:\Users\Bellie Coin\AppData\Local\DordStreamStudio\Projects\Temporary\Assets\js\jquery.js
Resources Generated
Demo  Successfully Published
10/24/2022 4:05:43 PM
```

- After publishing, then you can now upload it to the Themes section of the CMS software.