Topic/Title:

Keywords/Questions:

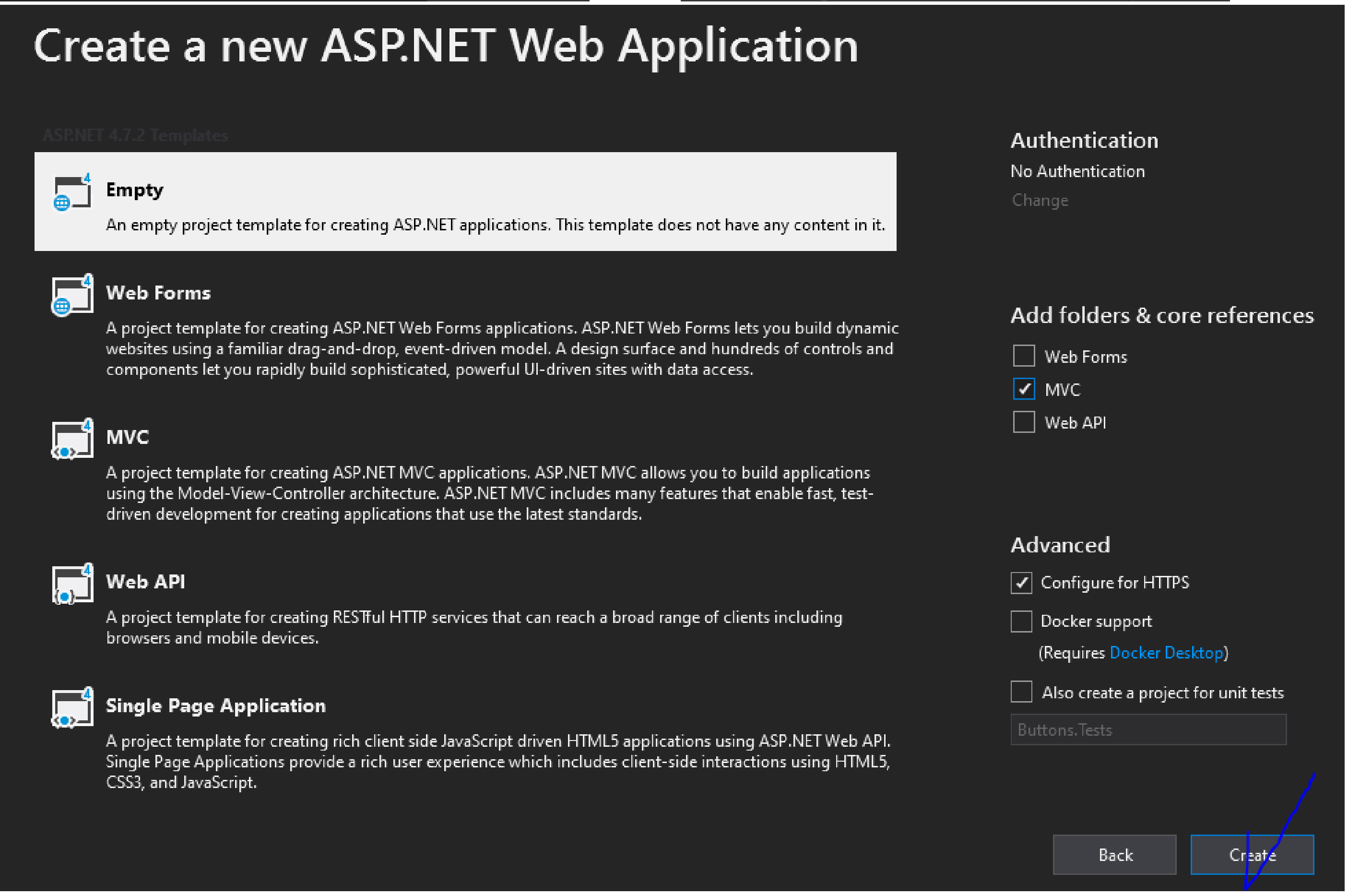
pixlr.com

Notes:

Summary:

ASP.NET Web Development Button Grid Game

What you will learn
1. Add graphics
2. Button click handler
3. Array of button objects


Create a new ASP.NET Web Application

ASP.NET 4.7.2 Templates

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication
No Authentication
Change

Add folders & core references
- [ ] Web Forms
- [x] MVC
- [ ] Web API

Advanced
- [x] Configure for HTTPS
- [ ] Docker support
  (Requires Docker Desktop)
- [ ] Also create a project for unit tests

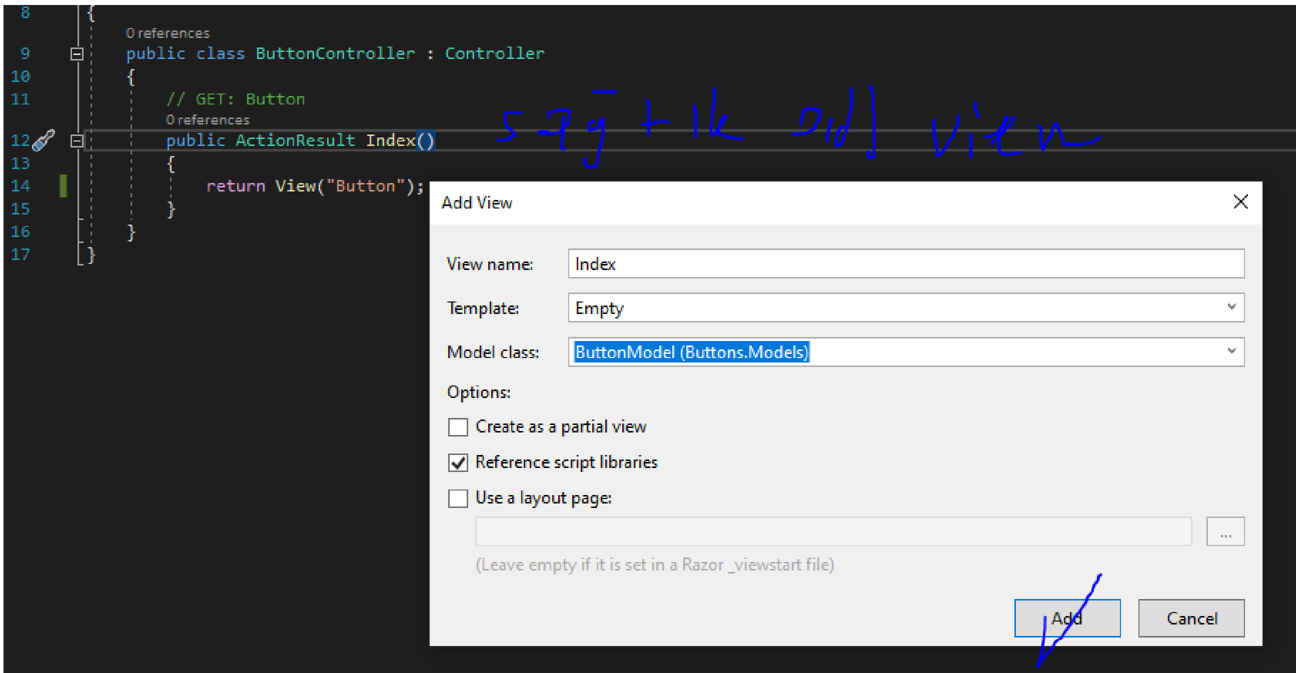
Buttons.Tests

İlk olarak Controllers klasörüne ButtonControllers adında controller ekle.
Models klasörüne ButtonModel.cs class ekle.

```
public class ButtonModel
  {
      public ButtonModel(bool state)
      {
          State = state;
      }

      public bool State { get; set; }
  }
```

ButtonController.cs ekle

```
public ActionResult Index()
        {
                    return View();

        }
```


```
{
    0 references
    public class ButtonController : Controller
    {
        // GET: Button
        0 references
        public ActionResult Index()
        {
            return View("Button");
        }
    }
}
```

sağ tık add view

Add View

View name:	Index
Template:	Empty
Model class:	ButtonModel (Buttons.Models)

Options:
- [ ] Create as a partial view
- [x] Reference script libraries
- [ ] Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Views/Button Index.cshtml

```
@model Buttons.Models.ButtonModel

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        <h1>Welcome to the buttons page.</h1>
    </div>
</body>
</html>
```

pixlr sitesiyle resim düzeltme yapabilirisin.on off button resmi bul. 50x50 size olsun.

images klasörü oluştur. buna ekle gifleri.

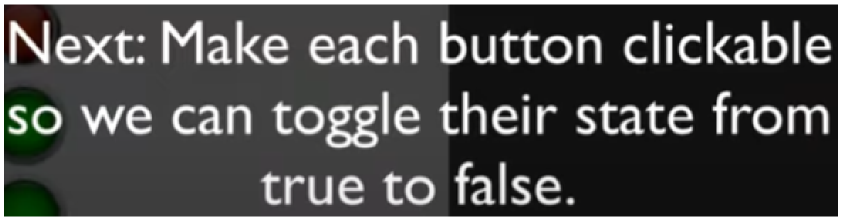
```
<h1>Welcome to the buttons page.</h1>
       @Html.Label("Playing with buttons")
       <img src="~/images/off.png"/>
       <img src="~/images/on.png" />
```

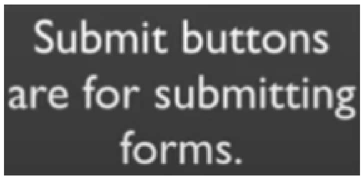
Models klasöründeki ButtonController.cs classına ekle

```
public class ButtonController : Controller
   {
       // GET: Button
       List<ButtonModel> buttons = new List<ButtonModel>();

       Random r = new Random();
       public ActionResult Index()
       {
           for (int i = 0; i < 25; i++)
           {
               if (r.Next(10) % 2 == 0)
               {
                   buttons.Add(new ButtonModel(true));
               }
               else
               {
                   buttons.Add(new ButtonModel(false));
               }

           }
           return View("Index",buttons);
       }
   }
```

View/Button/Index ekle

```
@model List<Buttons.Models.ButtonModel>
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        <h1>Welcome to the buttons page.</h1>
        @Html.Label("Playing with buttons")

        @for (int i = 0; i < Model.Count; i++)
        {

            if (i % 5 == 0)
            {
                <br />
            }
            if (Model[i].State == true)
            {
                <img src="~/images/off.png" />
            }
            else
            {
                <img src="~/images/on.png" />
            }
        }


    </div>
</body>
</html>
```


Next: Make each button clickable so we can toggle their state from true to false.

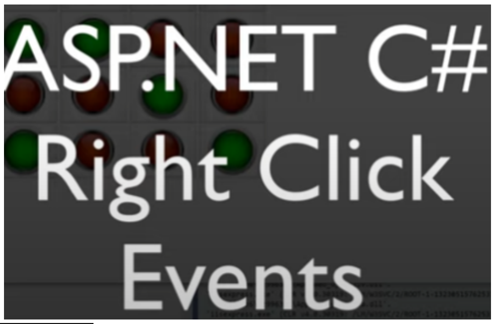

Submit buttons are for submitting forms.

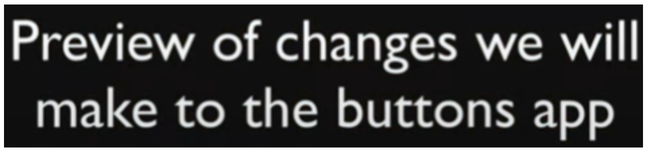
ButtonController.cs classına ekle

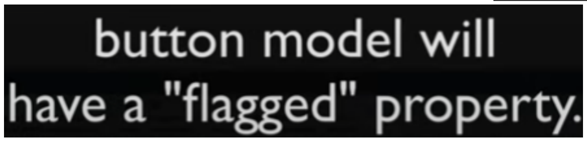
```
  public ActionResult HandleButtonClick(string mine)  //We are expecting a value to come back called
mine so that is the string that will be submitted from the form.
        {
            int number = int.Parse(mine);
            buttons[number].State = !buttons[number].State; //Change the state of the button clicked.
            return View("Index", buttons);
        }
```

index.cshtml ekle

```
<body>
    <div>
        <h1>Welcome to the buttons page.</h1>
        @Html.Label("Playing with buttons")
        @using (Html.BeginForm("HandleButtonClick", "Button"))
        {
            for (int i = 0; i < Model.Count; i++)
            {

                if (i % 5 == 0)
                {
                    <br />
                }
    <button type="submit" name="mine" value="@i">   @* submit send a message to server. value is number of
the button, button needs to have a form, submit                                                 buttons
are for submitting forms  *@
   @if (Model[i].Flagged == true)
        {
            <img src="~/images/flag.png" />
        }
else if
        {
        <img src="~/images/off.png" />
        }
        else
        {
        <img src="~/images/on.png" />
        }
    </button>
            }
        }
    </div>
</body>
```


ASP.NET C#
Right Click
Events


Preview of changes we will
make to the buttons app


button model will
have a "flagged" property.

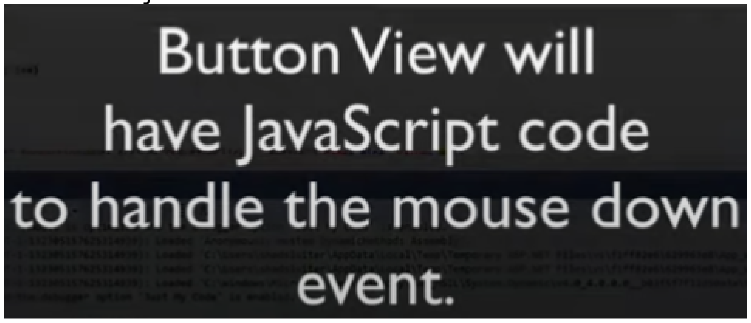
Models/ButtonModel.cs classa flagged property,constructor ekle

```
 public ButtonModel(bool state, bool flagged)
        {
            State = state;
            Flagged = flagged;
        }

        public bool State { get; set; }
        public bool Flagged { get; set; }
```

controller will have
a "OnRightClick" method

Controllers/ButtonController.cs classına OnRightClick methodunu ekle.

```
 public ActionResult OnButtonRightClick(string mine)
        {
            int mineNumber = Int32.Parse(mine);
            buttons[mineNumber].Flagged = !buttons[mineNumber].Flagged;
            return View("Button", buttons);
        }
```



Button View will have JavaScript code to handle the mouse down event.

ButtonController.cs ekle

```
 static List<ButtonModel> buttons = new List<ButtonModel>();      //static'i kaldır değişimi gör

        Random r = new Random();
        public ButtonController()
        {
            if (buttons.Count == 0)
            {
                for (int i = 0; i < 25; i++)
                {
                    if (r.Next(10) % 2 == 0)
                    {
                        buttons.Add(new ButtonModel(true,false));   //Açılışta flag olmayacak bu yüzden
false.
                    }
                    else
                    {
                        buttons.Add(new ButtonModel(false,false));
                    }

                }
            }
        }
        public ActionResult Index()
        {

            return View("Index",buttons);
        }
```

Index.cshtml ekle

Jquery and testing code

```
 <script src="~/Scripts/jquery-3.3.1.min.js"></script>
</head>
<script>
    $(document).ready(alert());
</script>
```

```
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
    <script src="~/Scripts/jquery-3.3.1.min.js"></script>
</head>
<script>

    $(document).ready(function () {
        $(document).contextmenu(function () {
            return false;
        });
    $(".game-button").mousedown(function (event) {

        if (event.which == 3) {
             console.log("event 3.right click");
            console.log(this.getAttribute("value"));
        }
    });
        });
</script>
<body>
    <div>
        <h1>Welcome to the buttons page.</h1>
        @Html.Label("Playing with buttons")
        @using (Html.BeginForm("HandleButtonClick", "Button"))
        {
            for (int i = 0; i < Model.Count; i++)
            {

                if (i % 5 == 0)
                {
                    <br />
                }
    <button class="game-button" type="submit" name="mine" value="@i">
        @* submit send a message to server. value is number of the button, button needs to have a form,
submit                                                  buttons are for submitting forms  *@
        @if (Model[i].Flagged == true)
        {
            <img src="~/images/flag.png" />
        }
        else if (Model[i].State == true)
        {
            <img src="~/images/off.png" />
        }
        else
        {
            <img src="~/images/on.png" />
        }
    </button>
            }
        }




    </div>
</body>
</html>
```

index.cshtml

```
<script>
    $(document).ready(function () {
        $(document).contextmenu(function () {
            return false;
        });
    $(".game-button").mousedown(function (event) {

        if (event.which == 3) {
            console.log("event 3.right click");
            console.log(this.getAttribute("value"));

            //send this click to ann event in the button controller

            $.post("@Url.Action("OnRightButtonClick","Button")", { mine: this.getAttribute("value") },
function (data) {


                //URL, Data and a callback
                console.log(data);
                //refresh the entire page
                $("body").html(data)
            });
        }
    });
        });
</script>
```

buttonController.cs class

```
  static List<ButtonModel> buttons = new List<ButtonModel>();      //static'i kaldır değişimi gör

          Random r = new Random();
          public ButtonController()
          {
              if (buttons.Count == 0)
              {
                  for (int i = 0; i < 25; i++)
                  {
                      if (r.Next(10) % 2 == 0)
                      {
                          buttons.Add(new ButtonModel(true,false));   //Açılışta flag olmayacak bu yüzden
  false.
                      }
                      else
                      {
                          buttons.Add(new ButtonModel(false,false));
                      }
                      //buttons[0].Flagged=true;
                  }
              }
          }
          public ActionResult Index()
          {

              return View("Index",buttons);
          }
          public ActionResult HandleButtonClick(string mine)  //We are expecting a value to come back
  called mine so that is the string that will be submitted from the form.
          {
              int number = int.Parse(mine);
              if (!buttons[number].Flagged)
              {
                  buttons[number].State = !buttons[number].State; //Only change the state of the button
  clicked if the flagged proprty is false
              }

              return View("Index", buttons);
          }
```

```
 public ActionResult HandleButtonClick(string mine)  //We are expecting a value to come back called
mine so that is the string that will be submitted from the form.
        {
            int number = int.Parse(mine);
            if (!buttons[number].Flagged)
            {
                buttons[number].State = !buttons[number].State; //Only change the state of the button
clicked if the flagged proprty is false
            }

            return View("Index", buttons);
        }
        public ActionResult OnRightButtonClick(string mine)
        {
            int mineNumber = Int32.Parse(mine);
            buttons[mineNumber].Flagged = !buttons[mineNumber].Flagged;
            return View("Index", buttons);
        }
```