**What is an HTML helper?**  
An HTML helper is a method that is used to render html content in a view. **HTML helpers** are implemented as **extension methods**.  
  
**For example**, to produce the HTML for a textbox with id="firstname" and name="firstname", we can type all the html in the view as shown below  
<input type="text" name="firtsname" id="firstname" />  
  
**OR**  
  
**We can use the "TextBox" html helper.**  
@Html.TextBox("firstname")  
  
  
  
There are several overloaded versions. To set a value, along with the name, use the following overloaded version.  
@Html.TextBox("firstname", "John")  
  
**The above html helper, generates the following HTML**  
<input id="firstname" name="firstname" type="text" value="John" />  
  
To set **HTML attributes**, use the following overloaded version. Notice that, we are passing HTML attributes (**style**& **title**) as an **anonymous type**.  
@Html.TextBox("firstname", "John", new { style = "background-color:Red; color:White; font-weight:bold", title="Please enter your first name" })  
Some of the **html attributes**, are reserved keywords. Examples include **class, readonly** etc. To use these attributes, use **"@"** symbol as shown below.  
@Html.TextBox("firstname", "John", new { @class = "redtextbox", @readonly="true" })  
  
**To generate a label for "First Name"**  
@Html.Label("fisrtname", "First Name")  
  
**To generate a textbox to enter password, so that the input is masked**  
@Html.Password("Password")  
  
**To generate a multi-line textbox with 5 rows and 20 columns**  
@Html.TextArea("Comments", "", 5, 20, null)  
  
**To generate a hidden textbox**  
@Html.Hidden("id")  
  
**Hidden textbox** is used to store id values. Id values are not displayed on the page to the end user, but we need them to update data when the form is posted to the server.  
  
  
**Is it mandatory to use HTML helpers?**  
No, you can type the required HTML, but using HTML helpers will greatly reduce the amount of HTML that we have to write in a view. Views should be as simple as possible. All the complicated logic to generate a control can be encapsulated into the helper, to keep views simple.

**Generating a dropdownlist control in mvc using HTML helpers**

To generate a dropdownlist, use **DropDownList**html helper. A **dropdownlist in MVC** is a collection of **SelectListItem** objects. Depending on your project requirement you may either **hard code the values in code** or **retrieve them from a database table.**

**Generating a dropdownlist using hard coded values**: We will use the following overloaded version of "DropDownList" html helper.  
DropDownList(string name, IEnumerable<SelectListItem> selectList, string optionLabel)  
  
The following code will generate a departments dropdown list. The first item in the list will be **"Select Department"**.  
@Html.DropDownList("Departments", new List<SelectListItem>  
{   
    new SelectListItem { Text = "IT", Value = "1", Selected=true},  
    new SelectListItem { Text = "HR", Value = "2"},  
    new SelectListItem { Text = "Payroll", Value = "3"}  
}, "Select Department")  
  
  
  
The downside of hard coding dropdownlist values with-in code is that, if we have to add or remove departments from the dropdownlist, the code needs to be modified.   
  
In most cases, we get values from the database table.

**o pass list of Departments from the controller, store them in "ViewBag"**  
public ActionResult Index()  
{  
    // Connect to the database  
    SampleDBContext db = new SampleDBContext();  
    // Retrieve departments, and build SelectList  
    ViewBag.Departments = new SelectList(db.Departments, "Id", "Name");  
              
    return View();  
}  
  
**Now in the "Index" view, access Departments list from "ViewBag"**  
@Html.DropDownList("Departments", "Select Department")

**how to set an item selected when an asp.net mvc dropdownlist options are loaded from a database table**. [Please watch Part 34](http://csharp-video-tutorials.blogspot.com/2013/06/part-34-generating-dropdownlist-control.html) before proceeding.  
  
To have the **"IT"** department selected, when the departments are loaded from **tblDepartment** table, use the following overloaded constructor of **"SelectList"** class. Notice that we are passing a value of **"1"** for "selectedValue" parameter.  
ViewBag.Departments = new SelectList(db.Departments, "Id", "Name", "1");  
  
  
  
If you run the application at this point, **"IT"** department will be selected, when the dropdownlist is rendered. The downside of hard-coding the **"selectedValue"** in code is that, application code needs to be modified, if we want "HR" department to be selected instead of "IT". **So every time there is a change in requirement, we need to change the application code.**  
  
  
  
Let's now discuss, the steps required to drive the selection of an item in the dropdownlist using a column in **tblDepartment**table.  
**Step 1**: Add **"IsSelected"** bit column to tblDepartment table  
ALTER TABLE tblDepartment  
ADD IsSelected BIT  
  
**Step 2:** At this point, this column will be null for all the rows in **tblDepartment**table. If we want **IT** department to be selected by default when the dropdownlist is loaded, set **"IsSelected=1"** for the **"IT"** department row.  
Update tblDepartment Set IsSelected = 1 Where Id = 2  
  
**Step 3:** Refresh ADO.NET Entity Data Model  
  
**Step 4:** Finally, make the following changes to the **"Index()"** action method in **"HomeController"** class.  
public ActionResult Index()  
{  
    SampleDBContext db = new SampleDBContext();  
    List<SelectListItem> selectListItems = new List<SelectListItem>();  
  
    foreach (Department department in db.Departments)  
    {  
        SelectListItem selectListItem = new SelectListItem  
        {  
            Text = department.Name,  
            Value = department.Id.ToString(),  
            Selected = department.IsSelected.HasValue ? department.IsSelected.Value : false  
        };  
        selectListItems.Add(selectListItem);  
    }  
  
    ViewBag.Departments = selectListItems;  
    return View();  
}  
  
Run the application and notice that, **"IT"** department is selected, when the dropdownlist is loaded.   
  
If you now want **"HR"** department to be selected, instead of **"IT"**, set **"IsSelected=1"** for **"HR"** department and **"IsSelected=0"** for **"IT"** department.  
Update tblDepartment Set IsSelected = 1 Where Id = 2  
Update tblDepartment Set IsSelected = 0 Where Id = 1

### Difference between Html.TextBox and Html.TextBoxFor

Let's understand the difference between **TextBox**and **TextBoxFor** & **DropDownList** and **DropDownListFor** HTML helpers with an example.

**Right click on the "Models" folder and add a class file with "name=Company.cs"**. Copy and paste the following code.  
public class Company  
{  
    private string \_name;  
    public Company(string name)  
    {  
        this.\_name = name;  
    }  
          
    public List<Department> Departments  
    {  
        get  
        {  
            SampleDBContext db = new SampleDBContext();  
            return db.Departments.ToList();  
        }  
    }  
  
    public string CompanyName  
    {  
        get  
        {  
            return \_name;  
        }  
        set  
        {  
            \_name = value;  
        }  
    }  
}  
  
  
  
Copy and paste the following code in **HomeController**class. Notice that we are storing the **"Departments"** and **"CompanyName"** in the **ViewBag** object.  
public ActionResult Index()  
{  
    Company company = new Company("Pragim");  
  
    ViewBag.Departments = new SelectList(company.Departments, "Id", "Name");  
    ViewBag.CompanyName = company.CompanyName;  
  
    return View();  
}  
  
Right click on the **"Index"** action method in **"HomeController"** and add a view with **"name=Index"**. Copy and paste the following code. Notice that, here **the view is not strongly typed**, and we are hard-coding the name for **TextBox** and **DropDownList** HTML helpers.  
@{  
    ViewBag.Title = "Index";  
}  
  
<h2>Index</h2>  
@Html.TextBox("CompanyName", (string)ViewBag.CompanyName)  
<br />  
@Html.DropDownList("Departments", "Select Department")  
  
Add the following **"Index1"** action method to **"HomeController"**. Notice that we are passing **"Company"** object to the View, and hence the **view is strongly typed**. Since the view is strongly typed, we can use **TextBoxFor** and **DropDownListFor** HTML helpers.  
public ActionResult Index1()  
{  
    Company company = new Company("Pragim");  
    return View(company);  
}  
  
Right click on the **"Index1"** action method in **"HomeController"** and add a view with **"name=Index1"**. Copy and paste the following code.  
@model MVCDemo.Models.Company  
  
@{  
    ViewBag.Title = "Index1";  
}  
  
<h2>Index1</h2>  
@Html.TextBoxFor(m => m.CompanyName)  
<br />  
@Html.DropDownListFor(m => m.Departments, new SelectList(Model.Departments, "Id", "Name"), "Select Department")  
  
At this point, run the application and navigate to **"http://localhost/MVCDemo/home/index"**. A textbox and a dropdownlist will be rendered. Right click on the page and view it's source. The generated HTML is as shown below.  
<h2>Index</h2>  
<input id="CompanyName" name="CompanyName" type="text" value="Test" />  
<br />  
<select id="Departments" name="Departments"><option value="">Select Department</option>  
<option value="1">IT</option>  
<option value="2">HR</option>  
<option value="3">Payroll</option>  
</select>  
  
Now navigate to **"http://localhost/MVCDemo/home/index1"** and view page source. The HTML will be exactly the same as above.  
  
**So, in short, here are the differences**  
Html.TextBox amd Html.DropDownList are not strongly typed and hence they doesn't require a strongly typed view. This means that we can hardcode whatever name we want. On the other hand, Html.TextBoxFor and Html.DropDownListFor are strongly typed and requires a strongly typed view, and the name is inferred from the lambda expression.  
  
Strongly typed HTML helpers also provide compile time checking.  
  
Since, in real time, we mostly use strongly typed views, prefer to use Html.TextBoxFor and Html.DropDownListFor over their counterparts.   
  
Whether, we use Html.TextBox & Html.DropDownList OR Html.TextBoxFor & Html.DropDownListFor, the end result is the same, that is they produce the same HTML.  
  
Strongly typed HTML helpers are added in MVC2.

### Generating a radiobuttonlist control in mvc using HTML helpers

**Right click on the "Models" folder and add a class file with "name=Company.cs"**. Copy and paste the following code.  
public class Company  
{  
    public string SelectedDepartment { get; set; }  
    public List<Department> Departments  
    {  
        get  
        {  
            SampleDBContext db = new SampleDBContext();  
            return db.Departments.ToList();  
        }  
    }  
}  
  
  
  
Copy and paste the following 2 **"Index"** action methods in **HomeController** class.  
[HttpGet]  
public ActionResult Index()  
{  
    Company company = new Company();  
    return View(company);  
}  
  
[HttpPost]  
public string Index(Company company)  
{  
    if (string.IsNullOrEmpty(company.SelectedDepartment))  
    {  
        return "You did not select any department";  
    }  
    else  
    {  
        return "You selected department with ID = " + company.SelectedDepartment;  
    }  
}  
  
Right click on the **"Index"** action method in **"HomeController"** and add a view with **"name=Index"**. Copy and paste the following code.   
@model MVCDemo.Models.Company  
@{  
    ViewBag.Title = "Index";  
}  
  
<h2>Index</h2>  
  
@using (Html.BeginForm())  
{  
    foreach (var department in Model.Departments)  
    {  
        @Html.RadioButtonFor(m => m.SelectedDepartment, department.Id) @department.Name  
    }  
    <br />  
    <br />  
    <input type="submit" value="Submit" />  
}  
  
Run the application and click on **"Submit"** without selecting any department. Notice that, you get a message stating you have not selected any department. On the other hand, select a department and click **"Submit"**. The selected department ID must be displayed.



