**Templated helpers are introduced in mvc 2**. These built in templated helpers can be broadly classified into 2 categories.  
**1.** Display Templates  
**2.** Editor Templates

**There are 3 DISPLAY templated helpers**  
**@Html.Display("EmployeeData")** - Used with a view that is not strongly typed. For example, if you have stored data in ViewData, then we can use this templated helper using the key that was used to store data in ViewData.  
**@Html.DisplayFor(model => model)** - Used with strongly typed views. If your model has properties that return complex objects, then this templated helper is very useful.  
**@Html.DisplayForModel()** - Used with strongly typed views. Walks thru each property, in the model to display the object.

**Along the same lines, there are 3 EDIT templated helpers**  
@Html.Editor("EmployeeData")  
@Html.EditorFor(model => model)  
@Html.EditorForModel()  
  
To associate metadata with model class properties, we use attributes. In the previous sessions of this video series, we have discussed about using various data annotations attributes. These templated helpers use metadata associated with the model to render the user interface.  
  
The built-in display and edit templated helpers can be very easily customised. We will discuss this in a later video session.  
  
We will use the following Employee class that we have been working with in the previous sessions.  
[MetadataType(typeof(EmployeeMetadata))]  
public partial class Employee  
{     
}  
public class EmployeeMetadata  
{  
    [HiddenInput(DisplayValue = false)]  
    public int Id { get; set; }  
  
    [ReadOnly(true)]  
    [DataType(DataType.EmailAddress)]  
    public string EmailAddress { get; set; }  
  
    [ScaffoldColumn(true)]  
    [DataType(DataType.Currency)]  
    public int? Salary { get; set; }  
  
    [DataType(DataType.Url)]  
    [UIHint("OpenInNewWindow")]  
    public string PersonalWebSite { get; set; }  
  
    [DisplayAttribute(Name = "Full Name")]  
    public string FullName { get; set; }  
  
    [DisplayFormat(DataFormatString = "{0:d}")]  
    public DateTime? HireDate { get; set; }  
  
    [DisplayFormat(NullDisplayText = "Gender not specified")]  
    public string Gender { get; set; }  
}  
  
Copy and paste the following **Details**action method in **HomeController**. Notice that, the employee object is stored in ViewData using **"EmployeeData"** key.  
public ActionResult Details(int id)  
{  
    SampleDBContext db = new SampleDBContext();  
    Employee employee = db.Employees.Single(x => x.Id == id);  
    ViewData["EmployeeData"] = employee;  
    return View();  
}  
  
Copy and paste the following code in **Details.cshtml** view. Since our employee object is in ViewData, we are using @Html.Display("EmployeeData") templated helper. At the moment **"Details.cshtml"** view does not have a Model associated with it. So it is not a strongly typed view.  
  
@{  
    ViewBag.Title = "Details";  
}  
  
<h2>Details</h2>  
  
<fieldset>  
    <legend>Employee</legend>  
    @Html.Display("EmployeeData")  
</fieldset>  
  
At this point, if you run the application, you should be able to view Employee details, as expected.  
  
Now, change the implementation of **"Details"**action method with in home controller as shown below. Notice that, instead of storing the **"Employee"** object in ViewData, we are passing it to the View.  
public ActionResult Details(int id)  
{  
    SampleDBContext db = new SampleDBContext();  
    Employee employee = db.Employees.Single(x => x.Id == id);  
    return View(employee);  
}  
  
Change **Details.cshtml** view as below. We have specified **"Employee"** as the model object. So, here we are working with a strongly typed view, and hence we are using @Html.DisplayFor(model => model) templated helper. Since, none of the properties of Employee class return a complex object, the ideal choice here would be, to use @Html.DisplayForModel() templated helper. In either cases, in this scenario you will get the same output.  
  
@model MVCDemo.Models.Employee  
@{  
    ViewBag.Title = "Details";  
}  
  
<h2>Details</h2>  
  
<fieldset>  
    <legend>Employee</legend>  
    @Html.DisplayFor(model => model)  
</fieldset>  
  
You work with **Editor templates** in the same way. In HomeController, implement **Edit**action method as shown below.  
public ActionResult Edit(int id)  
{  
    SampleDBContext db = new SampleDBContext();  
    Employee employee = db.Employees.Single(x => x.Id == id);  
    return View(employee);  
}  
  
**and in Edit.cshtml view**  
@model MVCDemo.Models.Employee  
@{  
    ViewBag.Title = "Edit";  
}  
  
<h2>Edit</h2>  
  
@using (@Html.BeginForm())  
{      
    @Html.EditorForModel()  
}