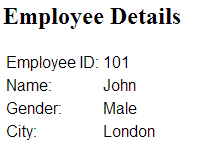
In this video we will discuss **models**in an **mvc application**.  
  
Let's understand **models** with an example. We want to retrieve an employee information from tblEmployee table and display it as shown below.  


To encapsulate Employee information, add Employee model class to the Models folder. To do this  
**1.** Right click on **"Models"** folder > Add > Class  
**2.** Name the class as Employee.cs  
**3.** Click **"Add"**  
  
**Copy and paste the following code in Employee.cs class file.**  
public class Employee  
{  
    public int EmployeeId { get; set; }  
    public string Name { get; set; }  
    public string Gender { get; set; }  
    public string City { get; set; }  
}  
  
**Now let's Add EmployeeController class to "Controllers" folder. To do this**  
**1.** Right click on "Controllers" folder > Add > Controller  
**2.** Use EmployeeController as the name  
**3.** Click "Add"  
  
We want to use **"Employee"** model class in **EmployeeController**. So copy and paste the following **"using"** statement in "EmployeeController.cs"  
using MVCDemo.Models;  
  
By default an Index() Action method is created in EmployeeController. Change the name of the function to **Details**(). Create an instance of Employee class. For now we will hard code Employee data in this class. In a later video session, we will discuss about retrieving employee information from the database table tblEmployee. At this point EmployeeController should look as shown below.  
public ActionResult Details()  
{  
    Employee employee = new Employee()  
    {  
        EmployeeId = 101,  
        Name = "John",  
        Gender = "Male",  
        City = "London"  
    };  
              
    return View();  
}  
  
Now, we need to pass the **employee**model object that we constructed in EmployeeController to a view, so the view can generate the HTML and send it to the requested client. To do this we first need to add a view. To add a view  
**1.** Right click on Details() action method and select "Add View" from the context menu  
**2.** Set  
    **a)**View Name = Details  
    **b)**View Engine = Razor  
    **c)**Select "Create strongly typed view" check box  
    **d)**From the "Model class" dropdownlist, select "Employee (MVCDemo.Models)"  
    Note: If Employee class is not visible in the dropdownlist, please build your project and then try adding the view again.  
**3.**Finally click "Add"  
  
At this point, **Details.cshtml** should be added to **"Employee"** folder. Please note that "Employee" folder is automatically created and added to "Views" folder.  
  
**Copy and paste the following code in Details.cshtml file.**  
@model MVCDemo.Models.Employee  
  
@{  
    ViewBag.Title = "Employee Details";  
}  
  
<h2>Employee Details</h2>  
  
<table style="font-family:Arial">  
    <tr>  
        <td>  
            Employee ID:  
        </td>  
        <td>  
            @Model.EmployeeId  
        </td>  
    </tr>  
    <tr>  
        <td>  
            Name:  
        </td>  
        <td>  
            @Model.Name  
        </td>  
    </tr>  
    <tr>  
        <td>  
            Gender:  
        </td>  
        <td>  
            @Model.Gender  
        </td>  
    </tr>  
    <tr>  
        <td>  
            City:  
        </td>  
        <td>  
            @Model.City  
        </td>  
    </tr>  
</table>  
  
At this point if you run the project, and if you navigate to the following URL, you get a runtime error stating - Object reference not set to an instance of an object.  
http://localhost/MVCDemo/Employee/Details  
  
To fix this error, pass **"Employee"** object to the view. The **"return"** statement in Details() action method need to be modified as shown below.  
return View(employee);  
  
That's it. Run the application and navigate to http://localhost/MVCDemo/Employee/Details. We should get the output as expected.