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**. In this video, we will discuss both the approaches.

**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. For this example, let's use entity framework to retrieve data. Add **ADO.NET** entity data model. We discussed working with entity framework in [Part 8](http://csharp-video-tutorials.blogspot.com/2013/05/part-8-data-access-in-mvc-using-entity.html) & [Part 25](http://csharp-video-tutorials.blogspot.com/2013/05/part-25-insert-update-delete-in-mvc.html).  
  
**To 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")