# ASP.Net MVC SelectList, SelectedValue, and DropDownListFor

This article will be an overview of how to use drop-down lists, setting a selected item, and issues you’ll run into on a strongly typed view. The following code applies to the Razor view engine and have been written for MVC 4.

**Sample Model**

For the sake of this article, assume we have two classes, Movie and Director. In our application we want to add new movies, and select directors from a drop-down list. The classes are structured as follows.

|  |  |
| --- | --- |
|  | public class Movie {   public int Id { get; set; }     public string Title { get; set; }     public virtual Director Director { get; set; }     public virtual int DirectorId { get; set; }}  public class Director {   public int Id { get; set; }     public string Name { get; set; } } |

**Html.DropDownList vs Html.DropDownListFor**

To add the drop-down markup to your .cshtml page you could of course simply write out a select element by hand, but then you lose out on validation. MVC provides HTML helpers for generating common HTML elements.

For an HTML select element you have two choices: Html.DropDownList and Html.DropDownListFor. The difference is the way they reference the name attribute of the resulting HTML element. DropDownList takes as it’s first argument a string that will be turned into the form field. So the call

|  |
| --- |
| @Html.DropDownList("director", directorList)  Assume directorList is an IEnumerable of SelectListItem to create options from |

Will result in an html element that looks like this.

|  |  |
| --- | --- |
|  | <select id="director" name="director">...</select> |

The problem with this approach is if you change the name of the property on your model from ‘director’ to ‘auteur’ you won’t get compile time checking and your form will no longer work with model binding. Html.DropDownListFor was introduced in MVC 2 and allows binding to strongly typed views. The first argument should be a lambda function that returns the model property you want the control to bind to. So in our case if the view-model includes a property DirectorId we can create a drop-down list with the code

|  |  |
| --- | --- |
|  | @Html.DropDownListFor(viewModel => viewModel.DirectorId, directorList) |

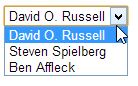
Which generates the following html

|  |  |
| --- | --- |
|  | <select id="DirectorId" name="DirectorId">  ...  </select> |

Now if we change the name of the Director property our build will break because the lambda expression will be invalid.Note that we’re using DirectorId instead of a Director object because we likely want to store the id in a foreign key.  
 **Populating Drop-Down List Options**

To populate a drop-down we need to pass the HTML helper an IEnumerable. This is easily created by making a SelectList object in your controller, and passing it in via a view-model.

|  |  |
| --- | --- |
|  | public ActionResult Index()  {      var directors = new Collection          {   new Director {Id = 1, Name = "David O. Russell"},              new Director {Id = 2, Name = "Steven Spielberg"},              new Director {Id = 3, Name = "Ben Affleck"}          };      var selectList = new SelectList(directors, "Id", "Name");      var vm = new ViewModel {DirectorSelectList = selectList};      return View(vm);  } |

A collection of directors and passed it into the SelectList constructor was created. This collection could have been queried from a database, this sample was simply for demonstration. I’ve also supplied which fields should be the drop-down value (Id) and display text (Name). If I omitted those additional parameters the ToString method would be called on the each object to generate the item.  
My drop-down list now looks like this.  
  
If you don’t like passing in a SelectList as part of your model you could pass in the IEnumerable and construct the SelectList in the view, but I prefer putting as little code in the view as possible.

**Setting a Default Value of the Drop-Down**

And here is where things start falling apart. The aforementioned StackOverflow question highlighted how hard it is to set a default value on a drop-down. The problem stems from poor documentation for DropDownListFor. There is an overload that takes a fourth parameter called selectedValue which is an object to set the value. Theoretically I should be able to have the following the default to the option with an id of 3

|  |  |
| --- | --- |
| 1 | var selectList = new SelectList(directors, "Id", "Name", new {id = 3}); |

However the first value, David O. Russell, is still selected in the view. And as good a job as he did in Silver Linings Playbook, I want to default to Ben Affleck. The problem is by using a strongly typed view, MVC is trying to bind my DirectorId field to the DirectorId property of my model. And because I didn’t populate the DirectorId field of my view-model, MVC is defaulting the drop-down to the first value.

My solution is to set the DirectorId property of the viewmodel to the value I want defaulted as follows.

|  |  |
| --- | --- |
| 1 | var vm = new ViewModel {DirectorSelectList = selectList, DirectorId = 3}; |

Then when my view is templated the correct option is selected.  
