ListBox and Dropdown list in asp.net mvc

Let’s say our requirement is to build a page to add a User where we will enter the name of user and select the city from a list of available cities. So to show this list we need to render a dropdown / SELECT element to our view for the city selection part. So Let’s start with a viewmodel.

What is a ViewModel ?

Do not be scared ! Viewmodels are nothing but POCO’s(Plain old class objects). These are simple classes which we uses to transfer data. They are specifically made for views. That means your viewmodel will have properties which are absolutely required in the view.

So for our screen, we will create a viewmodel like this

```

public class AddUserVM
{
    public string Name { set;get;}
    public List<SelectListItem> Cities { set;get;}
    public int SelectedCity { set;get;}

    public AddUserVM()
    {
        Cities=new List<SelectListItem>();
    }
}

```

and in our GET Action method, We will create an object of this viewmodel, set the `Cities` property and send it to the view.

```
public ActionResult AddUser()
{
    var addUserVM = new AddUserVM();
    addUserVM.Cities = GetCities();
    return View(addUserVM);
}
```

?

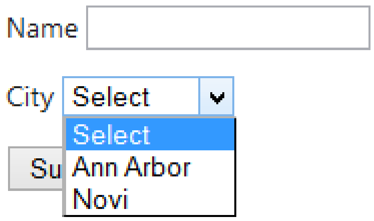
```
private List<SelectListItem> GetCities()
{
    var cityList = new List<SelectListItem>();
    //Hard coded for demo. You may get the items
    // from a datasource and add to the list
    cityList.Add(new SelectListItem { Value = "1", Text = "Ann Arbor" });
    cityList.Add(new SelectListItem { Value = "2", Text = "Novi" });
    return cityList;
}
```

You can see that, I am using the `GetCities` method to fill the data to the Cities property of our viewmodel. For demo purpose, i hard coded 2 items to the list and returning it. You may update this part to read the data from your db tables/XML/APIs and load it to the list.

And now we will make our razor view(AddUser.cshtml) a strongly typed one like below.

```
@model ReplaceYourNameSpaceHere.AddUserVM
@using (Html.BeginForm())
{
    <p>
        Name  @Html.TextBoxFor(s=>s.Name)
    </p>
    <p>
        City @Html.DropDownListFor(s=>s.SelectedCity,Model.Cities,"Select")
    </p>
    <input type="submit" />
}
```

So here we are using the `Html.DropDownListFor` html helper to render the SELECT element. When you run your page, you will get the out put like this where you have the Cities in the dropdown.



If you look at the source of the HTML page, You can see that we have a form tag which has action method set to “Home/AddUser”. So now let’s write the HttpPost action method to handle the form submit.

```
[HttpPost]
public ActionResult AddUser(AddUserVM model)
{
    if (ModelState.IsValid)
    {
        // get the selected city
        int city = model.SelectedCity;
        // to do : Save and redirect (PRG patttern) instead of this demo code below
        return View("Results", model);
    }
    //Reload the Cities collection again
    model.Cities = GetCities();
    return View(model);
}
```

You can see that the parameter for our HttpPost action method is of type `AddUserVM`, the same as what our view is strongly typed to. MVC Model binding will bind the posted form data to our method parameter. To get the selected item's value, you can check the `SelectedCity` property.Inside the method, we are checking for validation and If everything goes good, we will save the data and redirect to another get action (to follow the PRG pattern). If validation fails, we are returning the model back to the view again so that user can correct it and re submit.

Why are we reloading the Cities collection again ?

The answer is "Because HTTP is stateless". It can not keep the dropdown data between http requests like web forms do (web forms do this with the ugly viewstate which we don't have here in ASP.NET MVC). So If we are returning the model back to the view, we need to reload the data again.

Listbox / Multi selectable dropdown

If you want a listbox/multi selectable dropdown, The only change you have to make is to change the type of the selected item property to an integer array.

```
public class AddUserVM
{
    public string Name { set;get;}
    public List<SelectListItem> Cities { set;get;}
    public int[] SelectedCity { set;get;}

    public AddUserVM()
    {
        Cities=new List<SelectListItem>();
    }
}
```

Now when the user submits the form, the ids of the selected items will be available in the `SelectedCity` property which is an integer array.