# Model Binding

ASP.NET MVC – REVIEW

SEPT 9, 2016

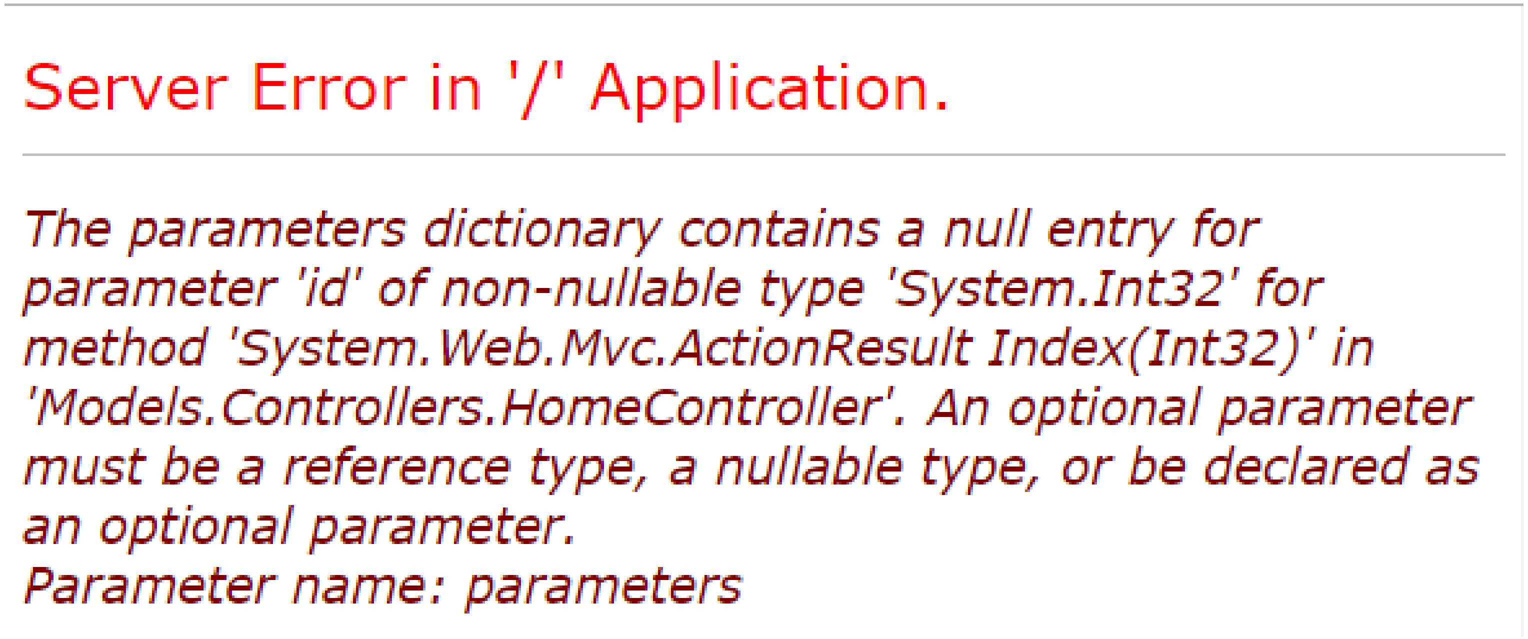
## Project set up

1. Create a new MVC application
2. Now delete it and download one from OneDrive =)

http://goo.gl/GnbeL3

## Model Binding

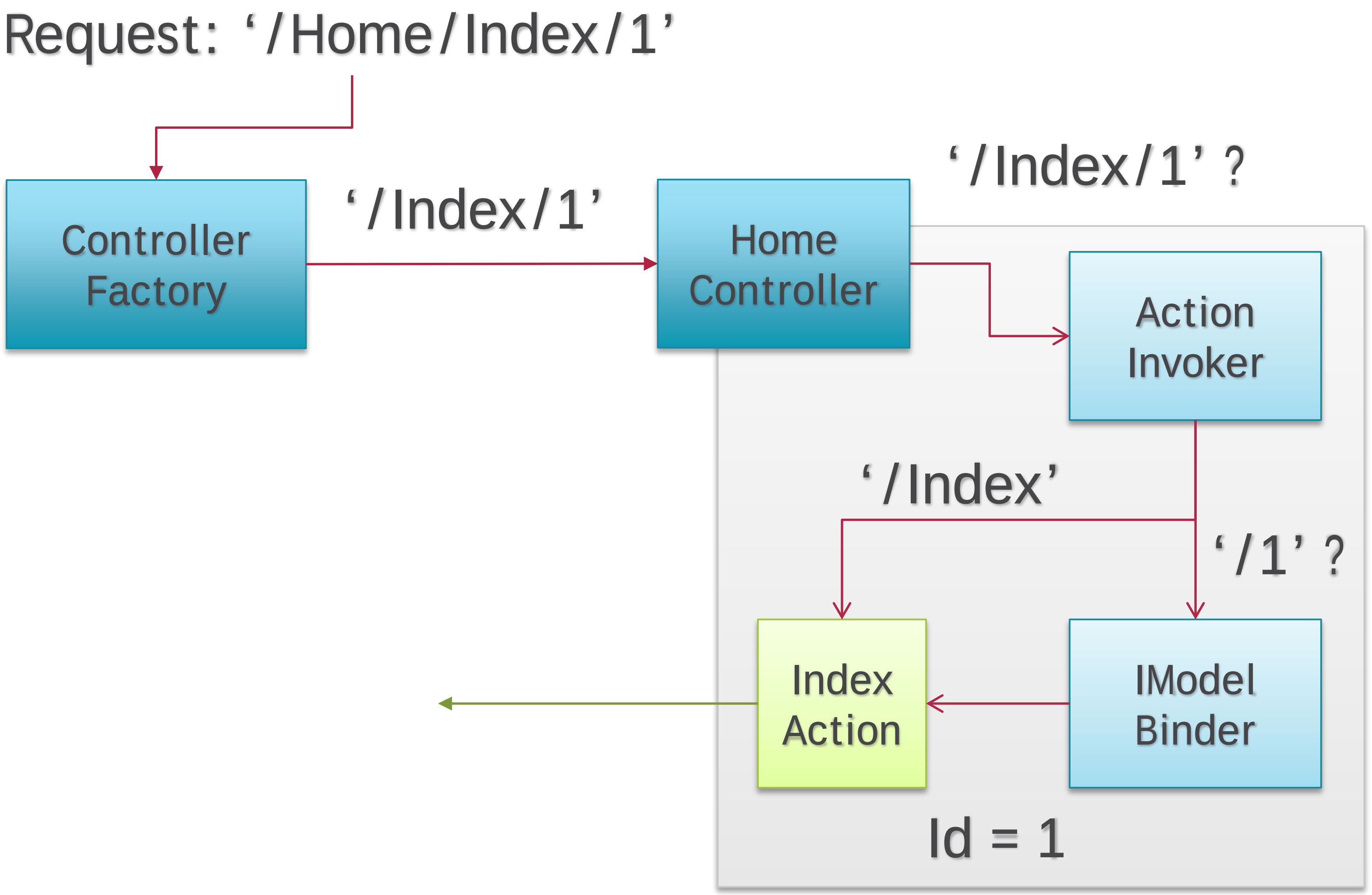
- If we navigates to ‘/Home/Index/1’ – everything works fine
- If we navigates to ‘/Home/Index/’



Server Error in '/' Application.

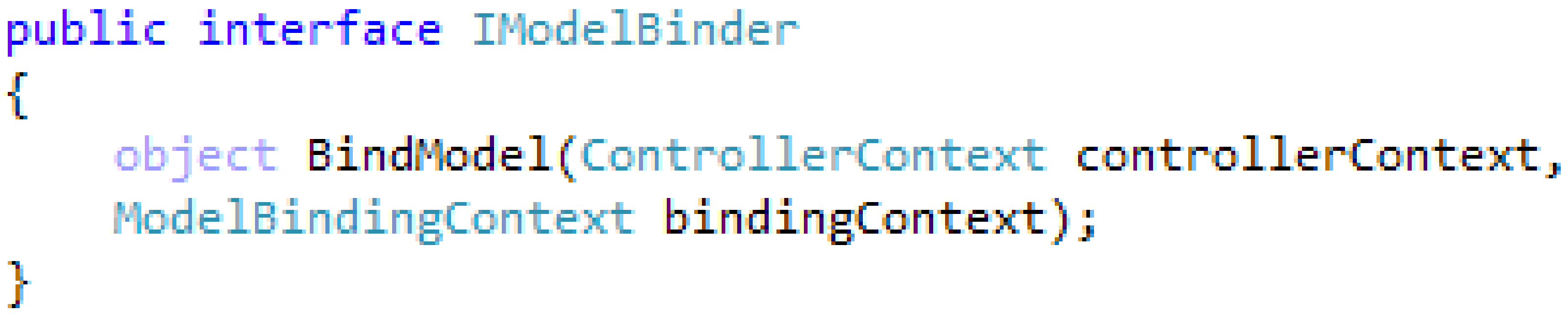
*The parameters dictionary contains a null entry for parameter 'id' of non-nullable type 'System.Int32' for method 'System.Web.Mvc.ActionResult Index(Int32)' in 'Models.Controllers.HomeController'. An optional parameter must be a reference type, a nullable type, or be declared as an optional parameter.*
*Parameter name: parameters*

# Model Binding



# Model Binding

**IModelBinder:**



```
public interface IModelBinder
{
    object BindModel(ControllerContext controllerContext,
    ModelBindingContext bindingContext);
}
```

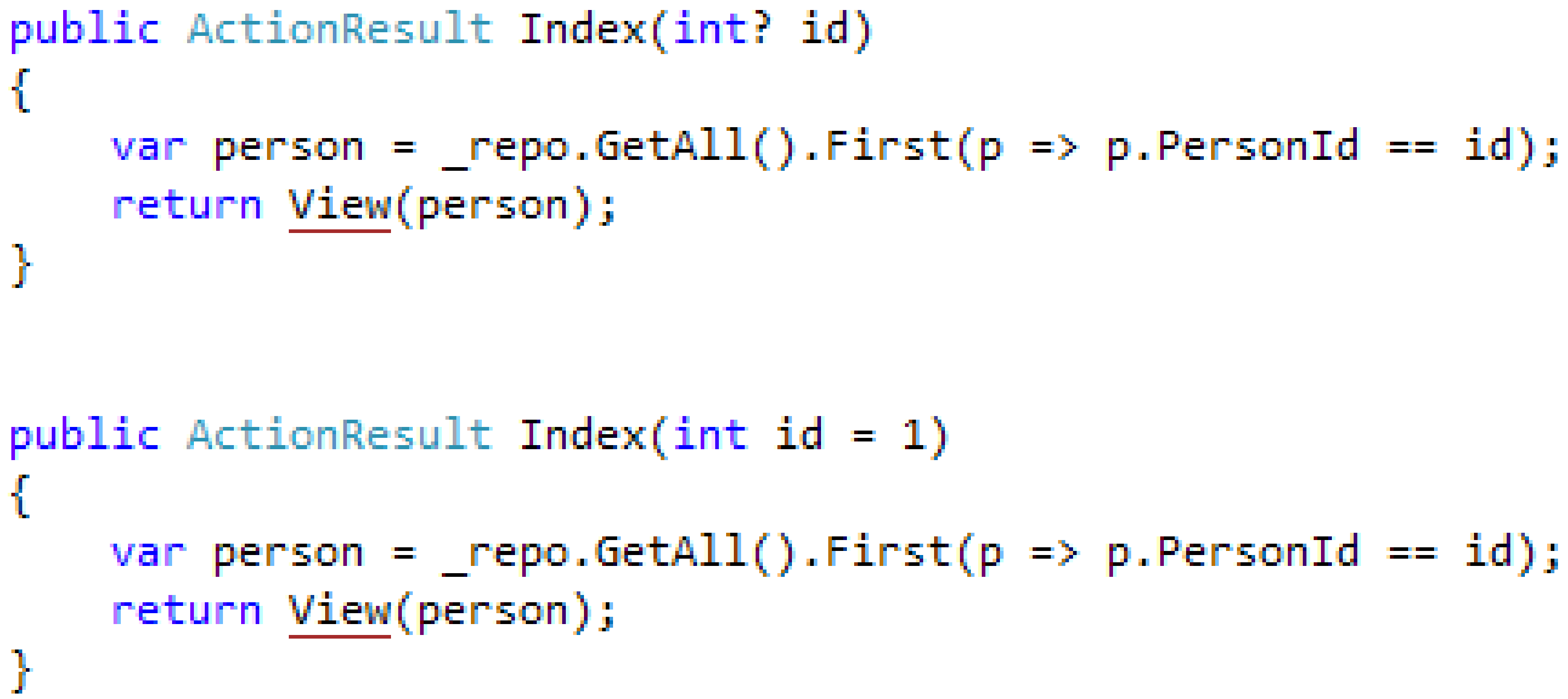
## Model Binding

### DefaultModelBinder

- Request.Form[“id”] - values from HTML form elements;
- RouteData.Values[“id”] - values from application routes;
- Request.QueryString[“id”] – values form the query string;
- Request.Files[“id”] – values from uploaded files;

**Note.** It is important that the parameters for your action method match the data property you are looking for.

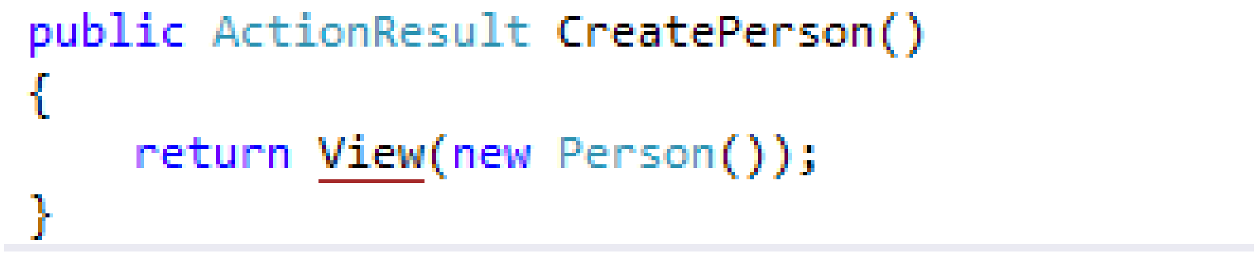
## Model Binding



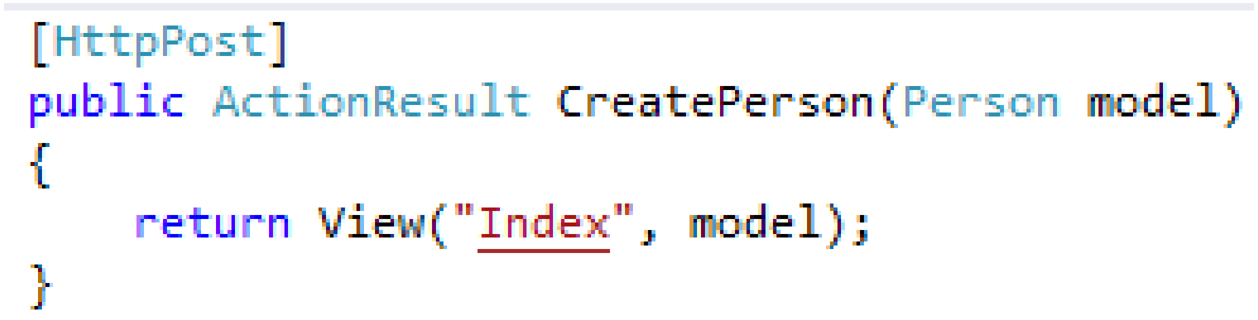
```
public ActionResult Index(int? id)
{
    var person = _repo.GetAll().First(p => p.PersonId == id);
    return View(person);
}


public ActionResult Index(int id = 1)
{
    var person = _repo.GetAll().First(p => p.PersonId == id);
    return View(person);
}
```

## Model Binding

Add 2 new actions to ‘HomeController’

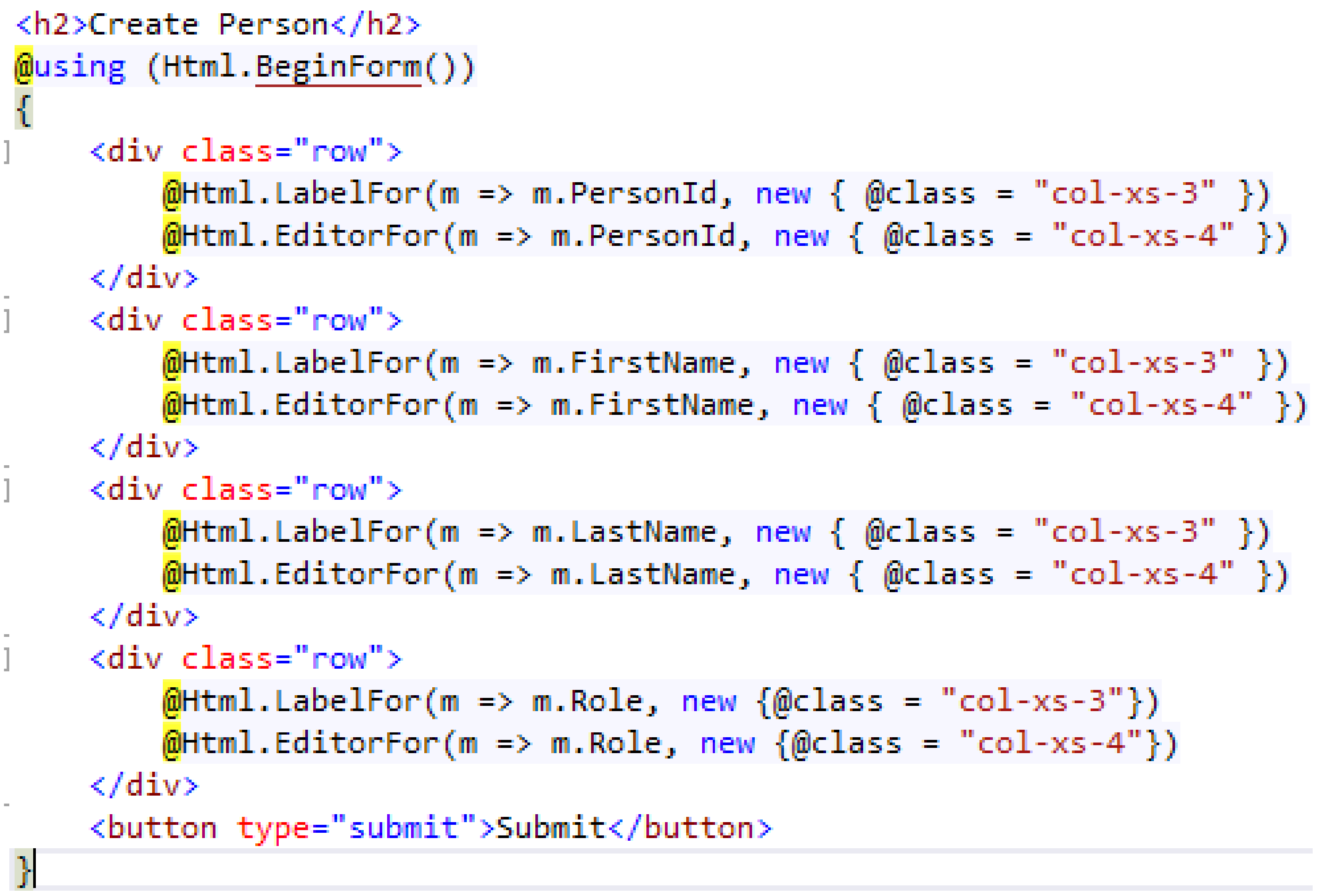


```
public ActionResult CreatePerson()
{
    return View(new Person());
}
```



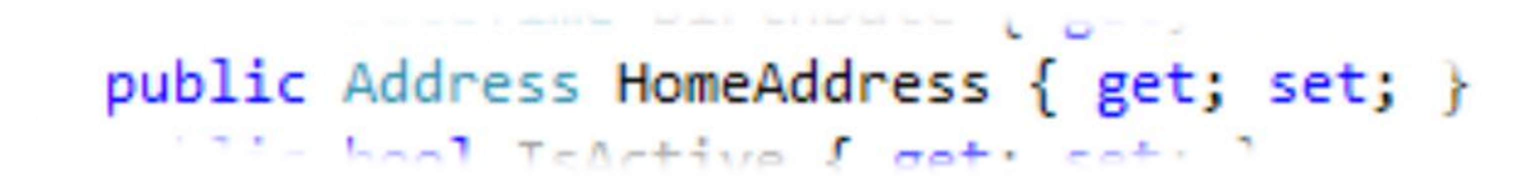
```
[HttpPost]
public ActionResult CreatePerson(Person model)
{
    return View("Index", model);
}
```

# Model Binding



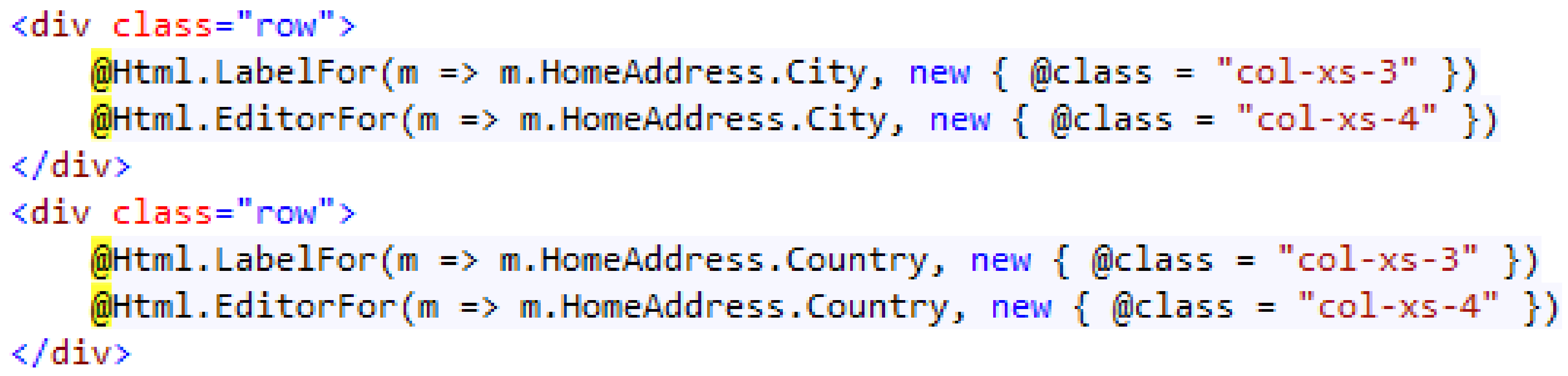
```
<h2>Create Person</h2>
@using (Html.BeginForm())
{
    <div class="row">
        @Html.LabelFor(m => m.PersonId, new { @class = "col-xs-3" })
        @Html.EditorFor(m => m.PersonId, new { @class = "col-xs-4" })
    </div>
    <div class="row">
        @Html.LabelFor(m => m.FirstName, new { @class = "col-xs-3" })
        @Html.EditorFor(m => m.FirstName, new { @class = "col-xs-4" })
    </div>
    <div class="row">
        @Html.LabelFor(m => m.LastName, new { @class = "col-xs-3" })
        @Html.EditorFor(m => m.LastName, new { @class = "col-xs-4" })
    </div>
    <div class="row">
        @Html.LabelFor(m => m.Role, new {@class = "col-xs-3"})
        @Html.EditorFor(m => m.Role, new {@class = "col-xs-4"})
    </div>
    <button type="submit">Submit</button>
}
```

## Model Binding



```
public Address HomeAddress { get; set; }
```

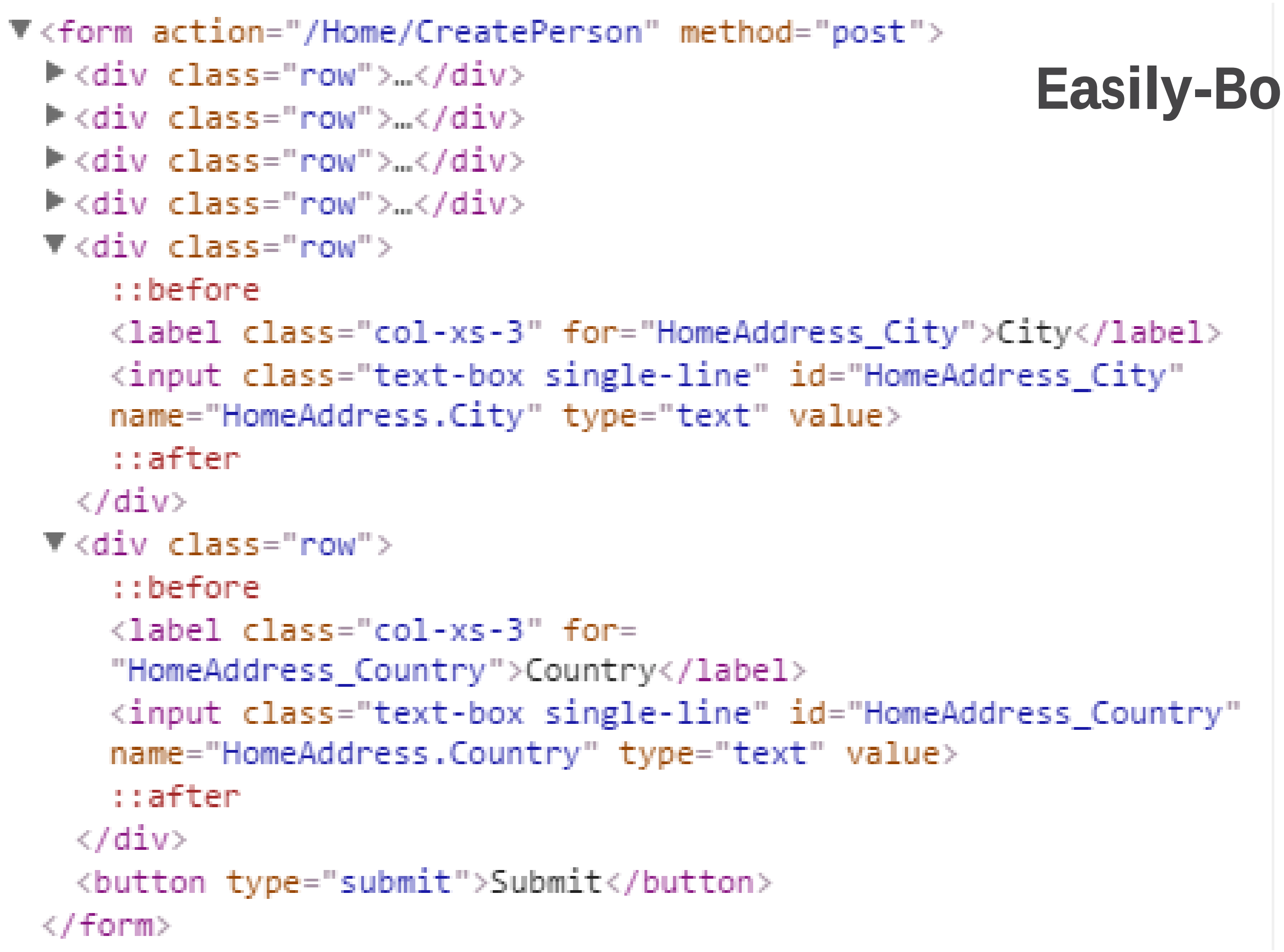
### Easily-Bound HTML



```
<div class="row">
    @Html.LabelFor(m => m.HomeAddress.City, new { @class = "col-xs-3" })
    @Html.EditorFor(m => m.HomeAddress.City, new { @class = "col-xs-4" })
</div>
<div class="row">
    @Html.LabelFor(m => m.HomeAddress.Country, new { @class = "col-xs-3" })
    @Html.EditorFor(m => m.HomeAddress.Country, new { @class = "col-xs-4" })
</div>
```

## Model Binding

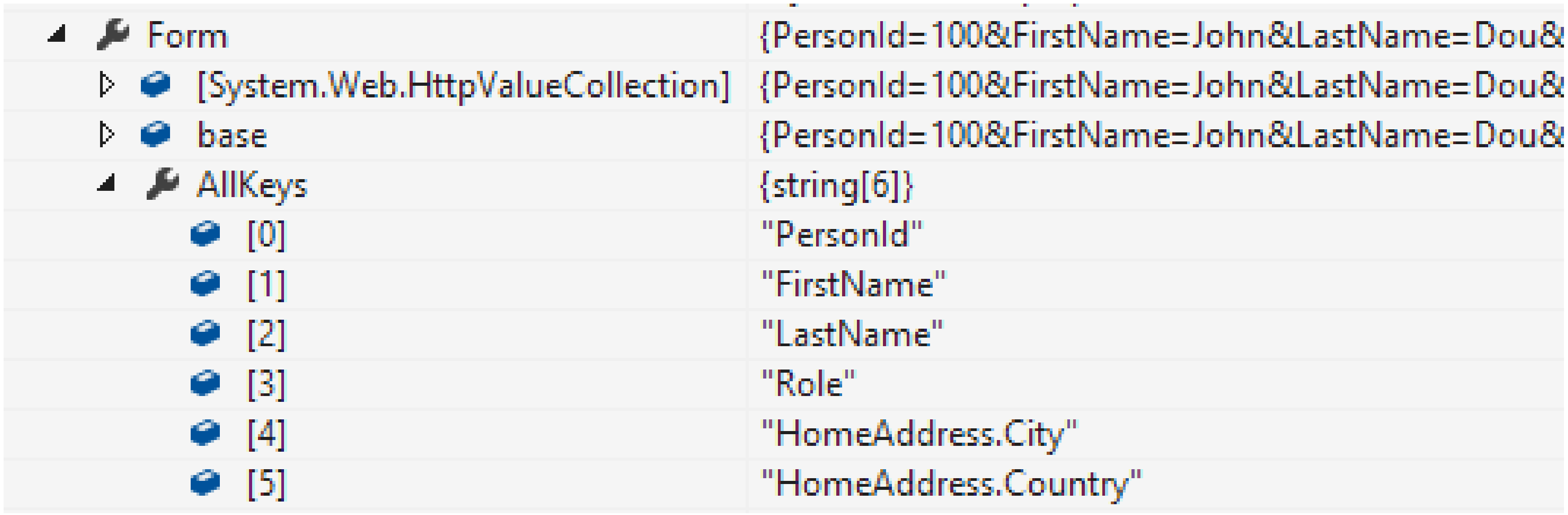
**Easily-Bound HTML**



```
▼<form action="/Home/CreatePerson" method="post">
  ▶<div class="row">…</div>
  ▶<div class="row">…</div>
  ▶<div class="row">…</div>
  ▶<div class="row">…</div>
  ▼<div class="row">
      ::before
      <label class="col-xs-3" for="HomeAddress_City">City</label>
      <input class="text-box single-line" id="HomeAddress_City"
      name="HomeAddress.City" type="text" value>
      ::after
    </div>
  ▼<div class="row">
      ::before
      <label class="col-xs-3" for=
      "HomeAddress_Country">Country</label>
      <input class="text-box single-line" id="HomeAddress_Country"
      name="HomeAddress.Country" type="text" value>
      ::after
    </div>
    <button type="submit">Submit</button>
  </form>
```

# Model Binding

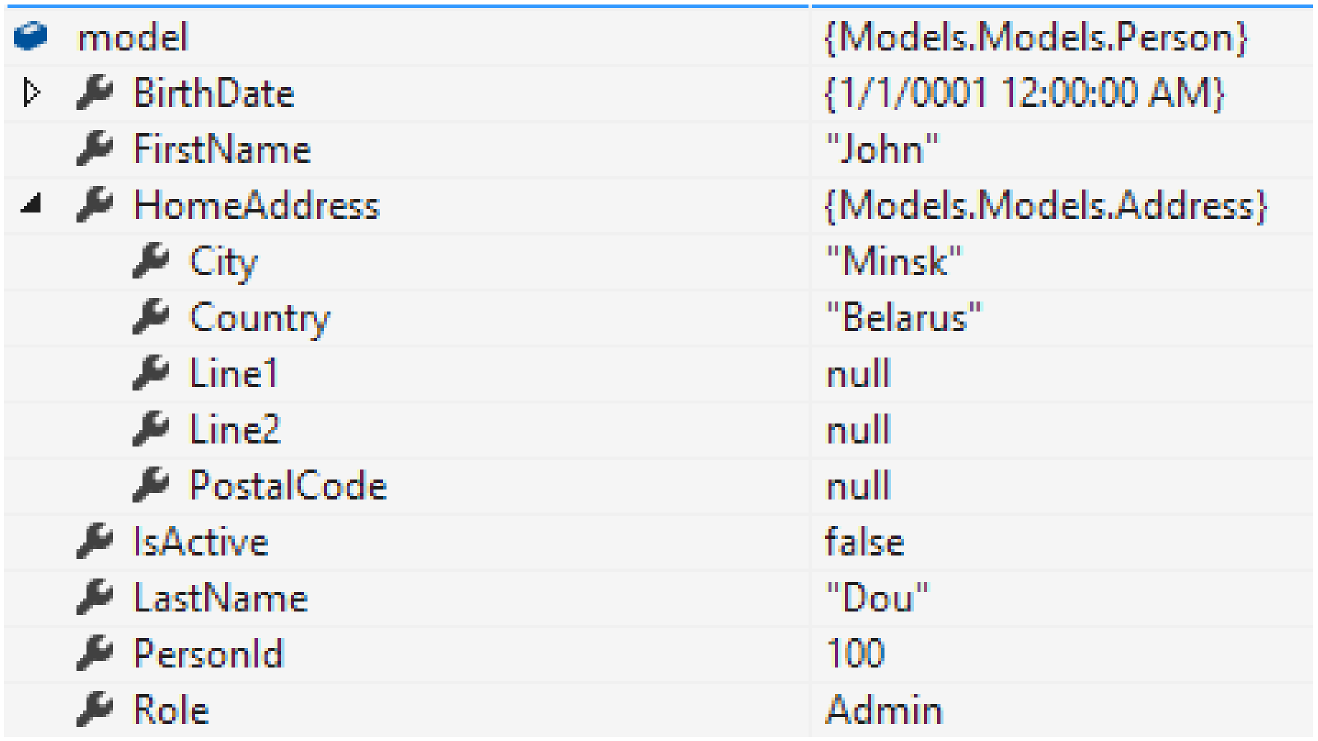
## Easily-Bound HTML



```
◢ Form                                    {PersonId=100&FirstName=John&LastName=Dou&
    ▷ [System.Web.HttpValueCollection]    {PersonId=100&FirstName=John&LastName=Dou&
    ▷ base                                {PersonId=100&FirstName=John&LastName=Dou&
    ◢ AllKeys                             {string[6]}
          [0]                             "PersonId"
          [1]                             "FirstName"
          [2]                             "LastName"
          [3]                             "Role"
          [4]                             "HomeAddress.City"
          [5]                             "HomeAddress.Country"
```

## Model Binding

### Easily-Bound HTML



| | |
|---|---|
| model | {Models.Models.Person} |
| ▷ BirthDate | {1/1/0001 12:00:00 AM} |
| FirstName | "John" |
| ◢ HomeAddress | {Models.Models.Address} |
| City | "Minsk" |
| Country | "Belarus" |
| Line1 | null |
| Line2 | null |
| PostalCode | null |
| IsActive | false |
| LastName | "Dou" |
| PersonId | 100 |
| Role | Admin |

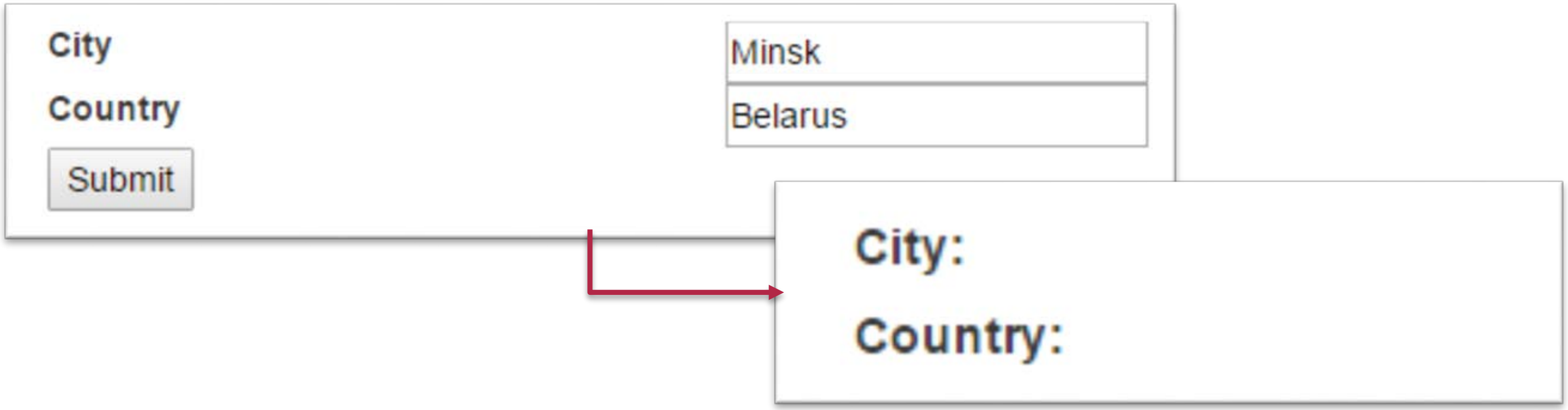
## Model Binding

Add new action method:

```
public ActionResult DisplaySummary(Address summary)
{
    return View(summary);
}
```

Add view and update existing form:

```
@using (Html.BeginForm("DisplaySummary", "Home"))
```



| | |
|---|---|
| City | Minsk |
| Country | Belarus |

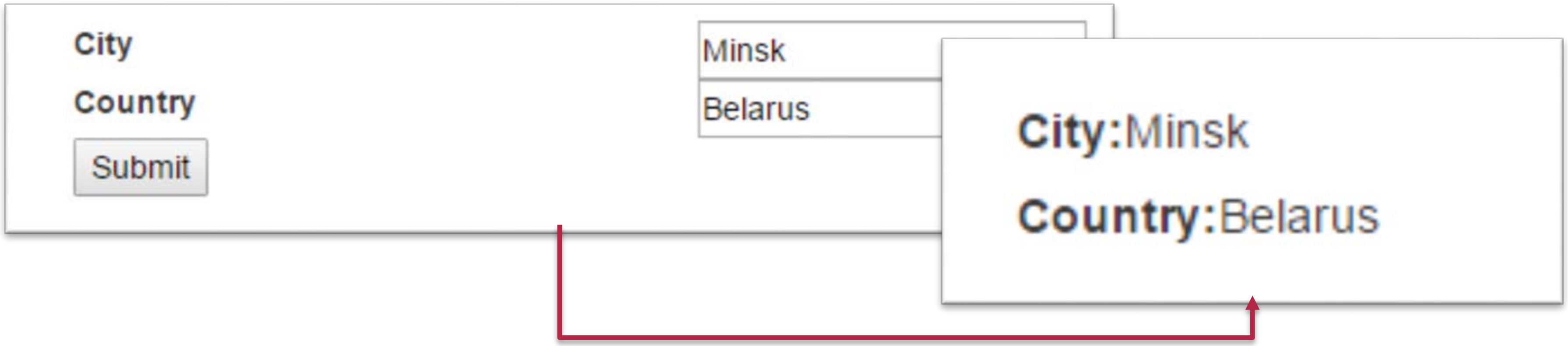
City:

Country:

## Model Binding

To successfully map data to new model we should define a local mapper prefix:

```
public ActionResult DisplaySummary(
    [Bind(Prefix = "HomeAddress")] Address summary)
{
    return View(summary);
}
```



| | |
|---|---|
| City | Minsk |
| Country | Belarus |

Submit

**City:**Minsk

**Country:**Belarus

## Model Binding

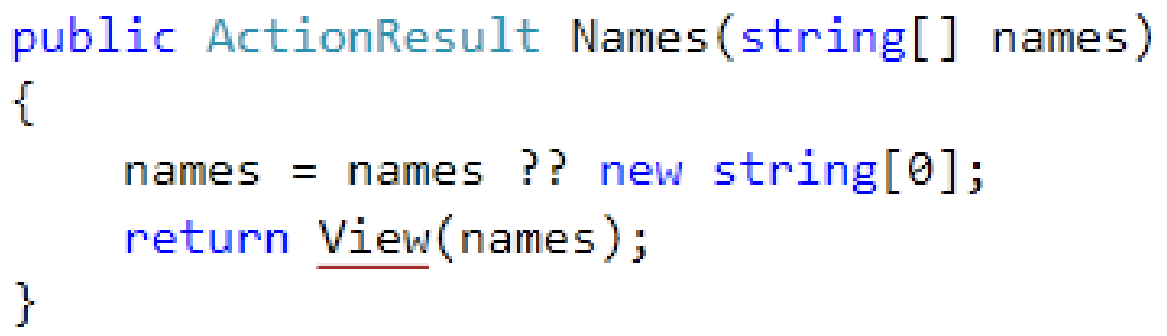
### Excluding elements from mapping

```
public ActionResult DisplaySummary(
    [Bind(Prefix = "HomeAddress", Exclude="Country")] Address summary)
```

```
[Bind(Include = "City")]
public class Address
{
    public string Line1 { get; set; }
    public string Line2 { get; set; }
    public string City { get; set; }
    public string PostalCode { get; set; }
    public string Country { get; set; }
}
```

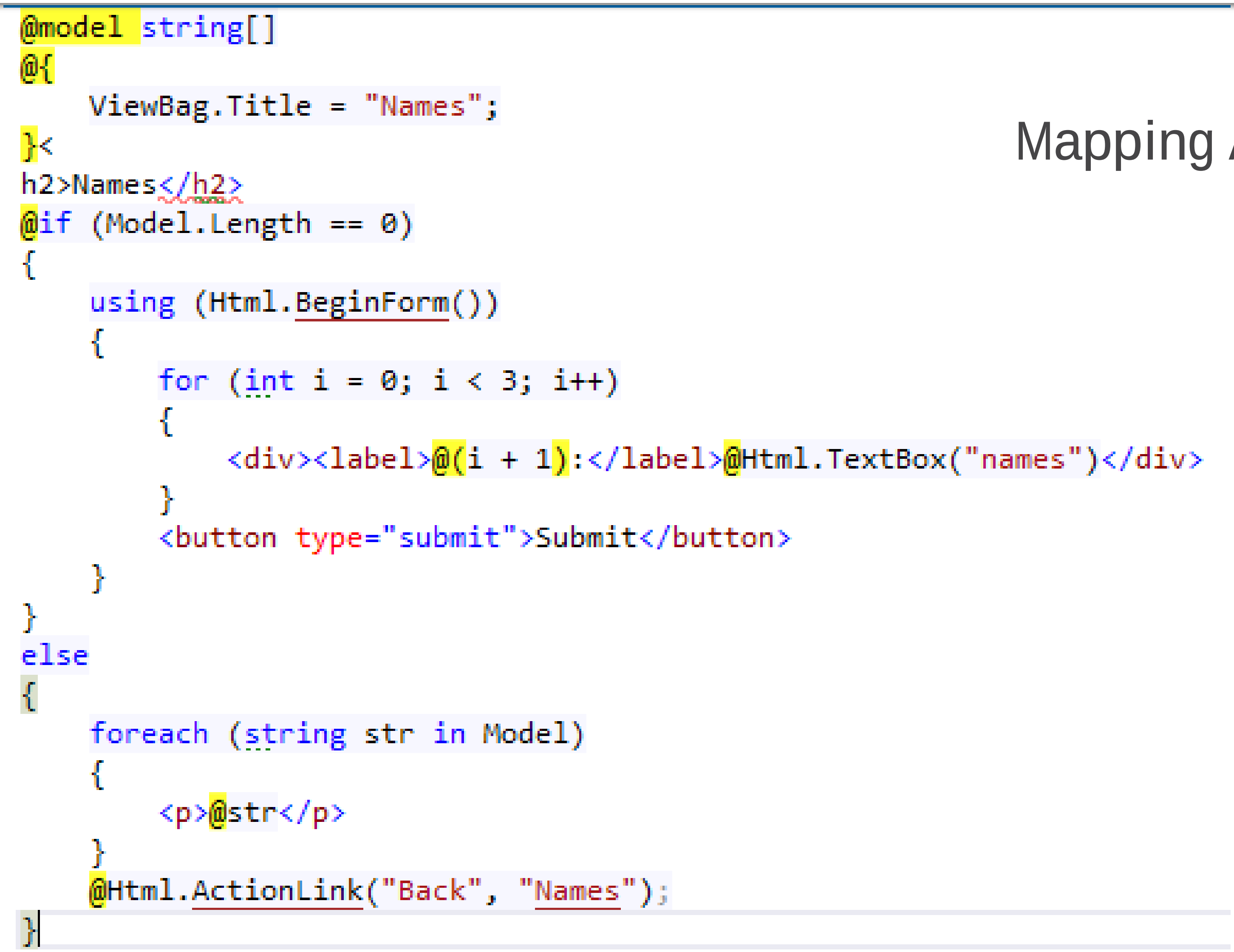
# Model Binding

## Mapping Arrays



```
public ActionResult Names(string[] names)
{
    names = names ?? new string[0];
    return View(names);
}
```

# Model Binding

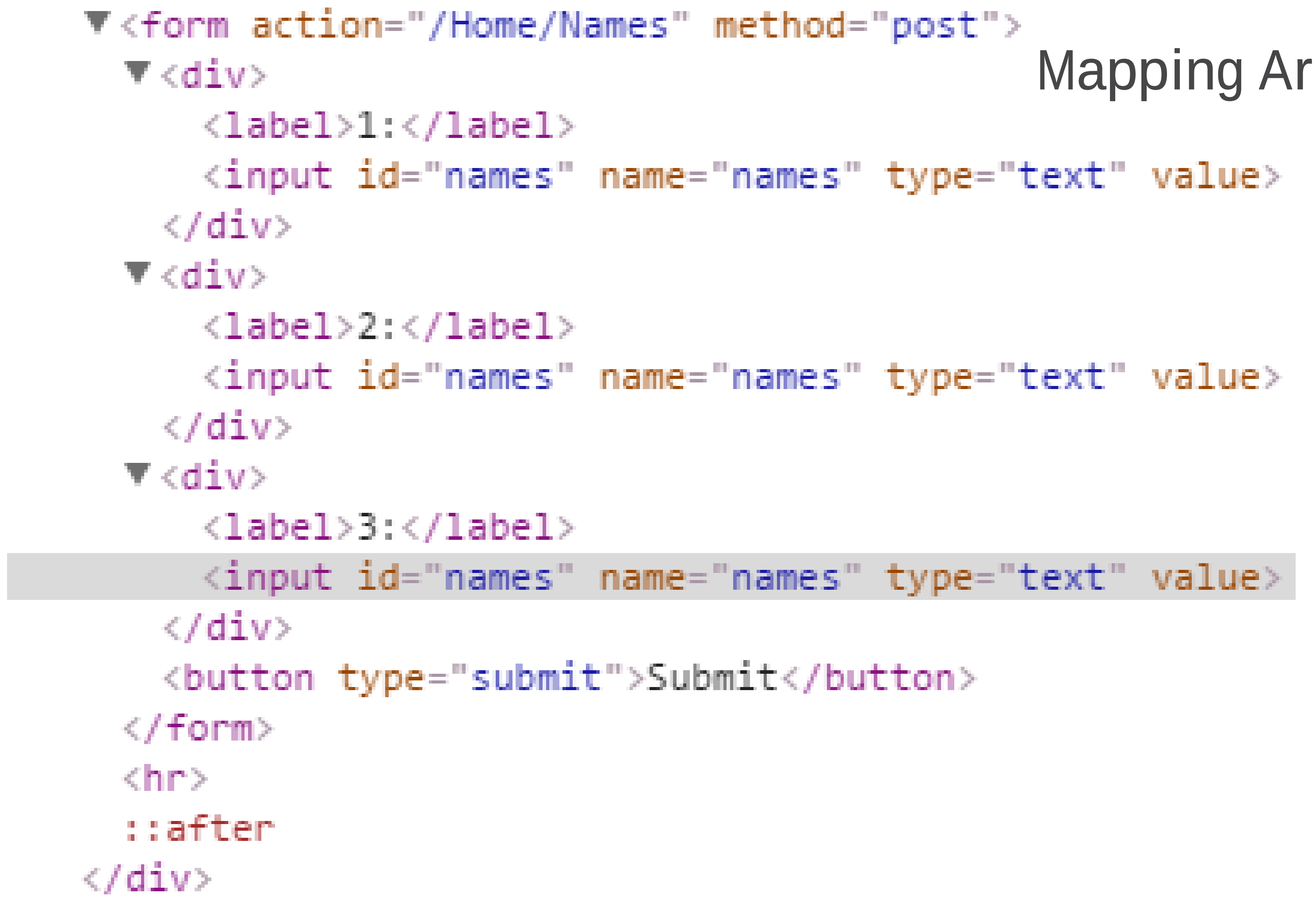


```
@model string[]
@{
    ViewBag.Title = "Names";
}<
h2>Names</h2>
@if (Model.Length == 0)
{
    using (Html.BeginForm())
    {
        for (int i = 0; i < 3; i++)
        {
            <div><label>@(i + 1):</label>@Html.TextBox("names")</div>
        }
        <button type="submit">Submit</button>
    }
}
else
{
    foreach (string str in Model)
    {
        <p>@str</p>
    }
    @Html.ActionLink("Back", "Names");
}
```

Mapping Arrays

# Model Binding

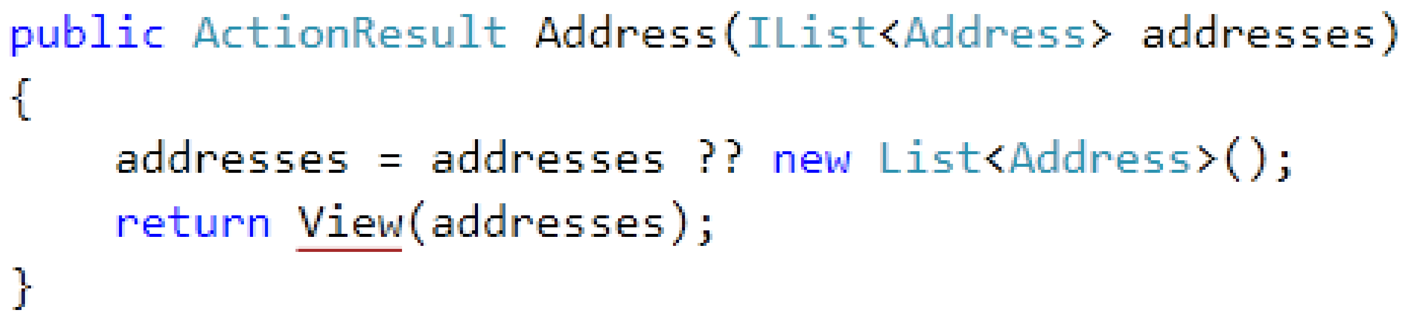
Mapping Arrays



```
▼<form action="/Home/Names" method="post">
  ▼<div>
      <label>1:</label>
      <input id="names" name="names" type="text" value>
    </div>
  ▼<div>
      <label>2:</label>
      <input id="names" name="names" type="text" value>
    </div>
  ▼<div>
      <label>3:</label>
      <input id="names" name="names" type="text" value>
    </div>
    <button type="submit">Submit</button>
  </form>
  <hr>
  ::after
</div>
```

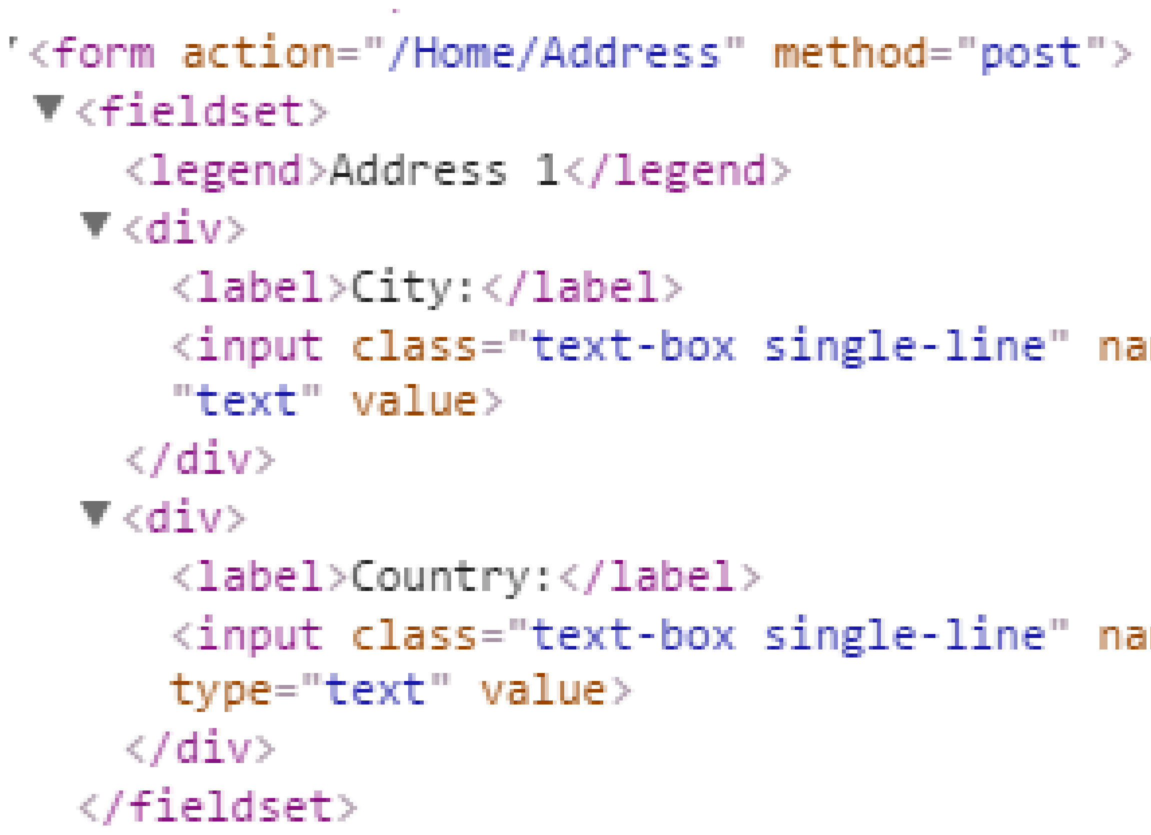
# Model Binding

## Mapping Collections



```
public ActionResult Address(IList<Address> addresses)
{
    addresses = addresses ?? new List<Address>();
    return View(addresses);
}
```

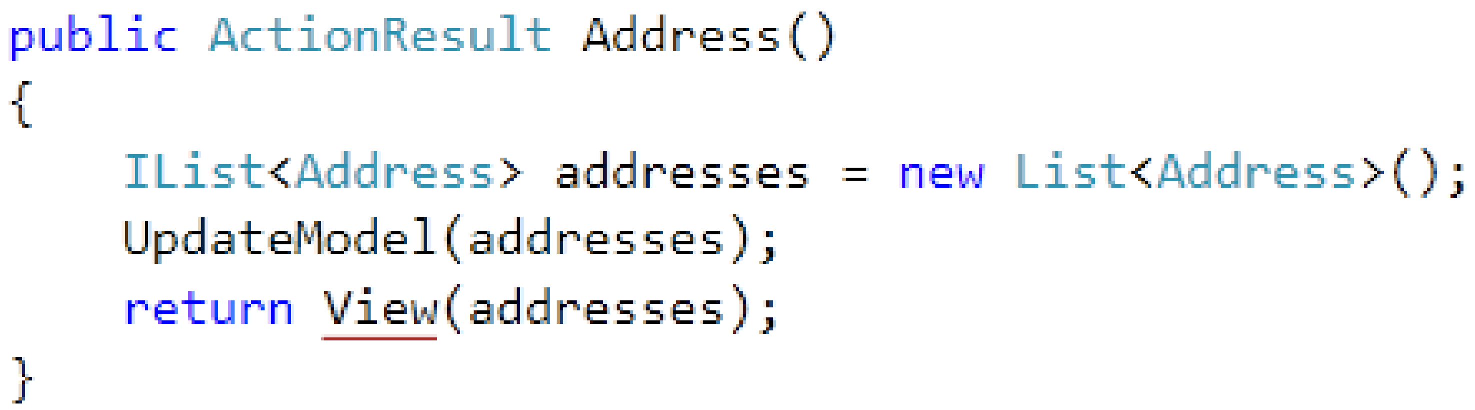
# Model Binding



```
<form action="/Home/Address" method="post">
 ▼<fieldset>
    <legend>Address 1</legend>
  ▼<div>
      <label>City:</label>
      <input class="text-box single-line" name="[0].City" type=
      "text" value>
    </div>
  ▼<div>
      <label>Country:</label>
      <input class="text-box single-line" name="[0].Country"
      type="text" value>
    </div>
  </fieldset>
```

Mapping Collections

# Invoking Model Binding



```
public ActionResult Address()
{
    IList<Address> addresses = new List<Address>();
    UpdateModel(addresses);
    return View(addresses);
}
```

The UpdateModel method takes a model object that I was previously defining as a parameter and tries to obtain values for its public properties using the standard binding process.

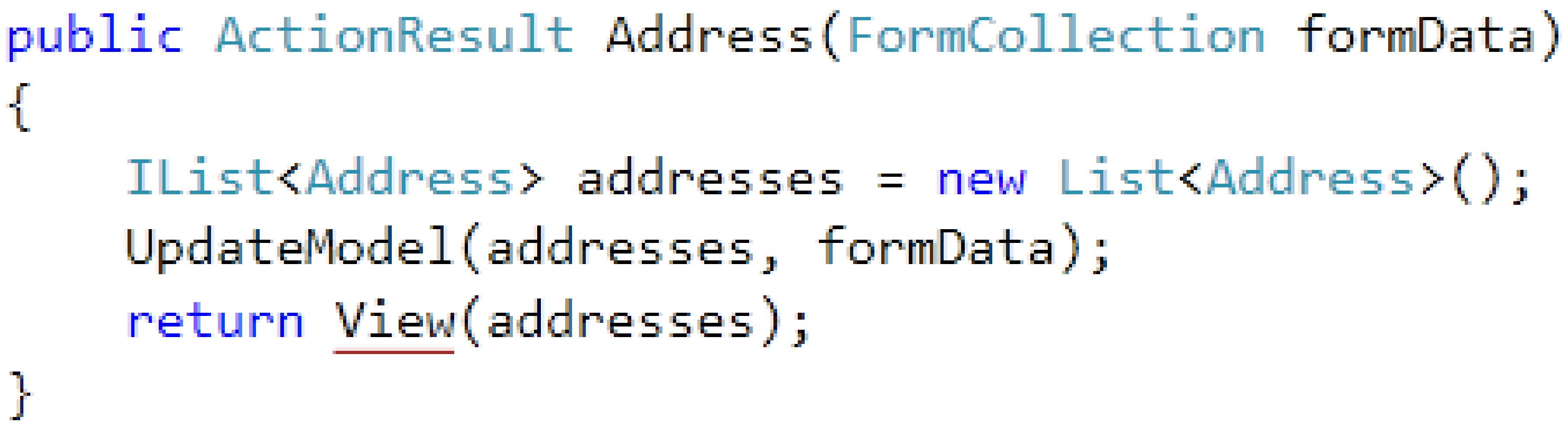
## Invoking Model Binding

```
UpdateModel(addresses,
    new FormValueProvider(ControllerContext));
```

| | |
|---|---|
| • Request.Form | FormValueProvider |
| • RouteData.Values | RouteDataValueProvider |
| • Request.QueryString | QueryStringValueProvider |
| • Request.Files | HttpFileCollectionValueProvider |

# Invoking Model Binding

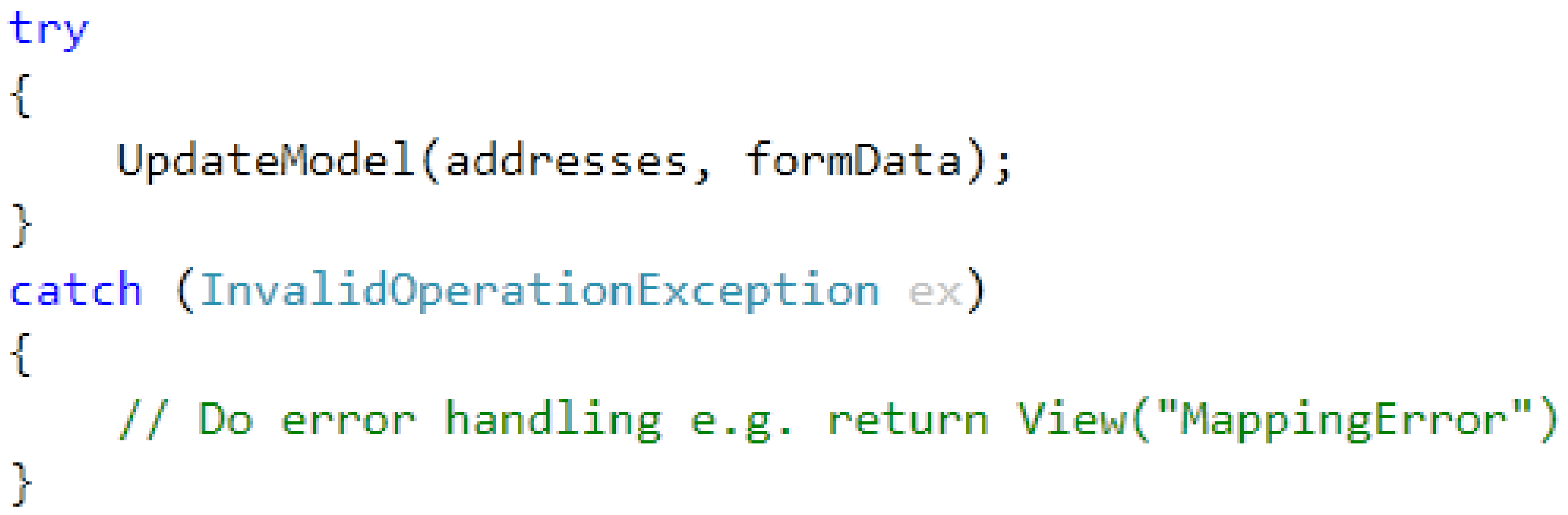
| | |
|---|---|
| • Request.Form | FormValueProvider |
| • RouteData.Values | RouteDataValueProvider |
| • Request.QueryString | QueryStringValueProvider |
| • Request.Files | HttpFileCollectionValueProvider |



```
public ActionResult Address(FormCollection formData)
{
    IList<Address> addresses = new List<Address>();
    UpdateModel(addresses, formData);
    return View(addresses);
}
```

## Binding Errors

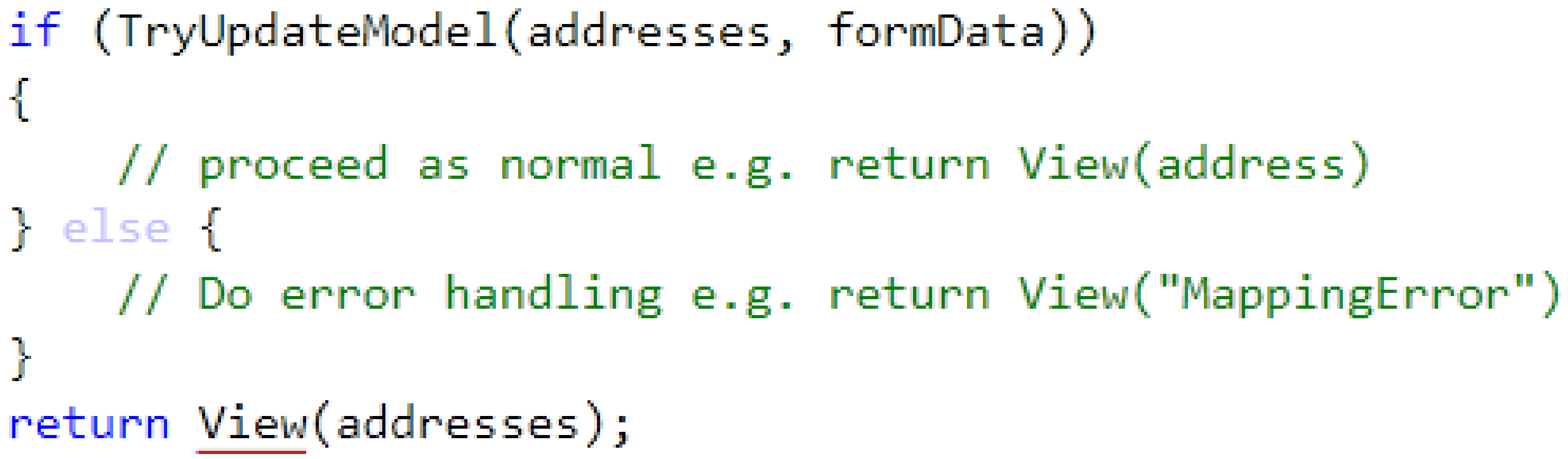
As we are calling binding manually we should aslo do an error handling by ourselves.



```
try
{
    UpdateModel(addresses, formData);
}
catch (InvalidOperationException ex)
{
    // Do error handling e.g. return View("MappingError")
}
```

## Binding Errors

As we are calling binding manually we should aslo do an error handling by ourselves.



```
if (TryUpdateModel(addresses, formData))
{
    // proceed as normal e.g. return View(address)
} else {
    // Do error handling e.g. return View("MappingError")
}
return View(addresses);
```

## Custom Binding Mechanics

To create a custom binding mechanics we need to implement IValueProvider interface:

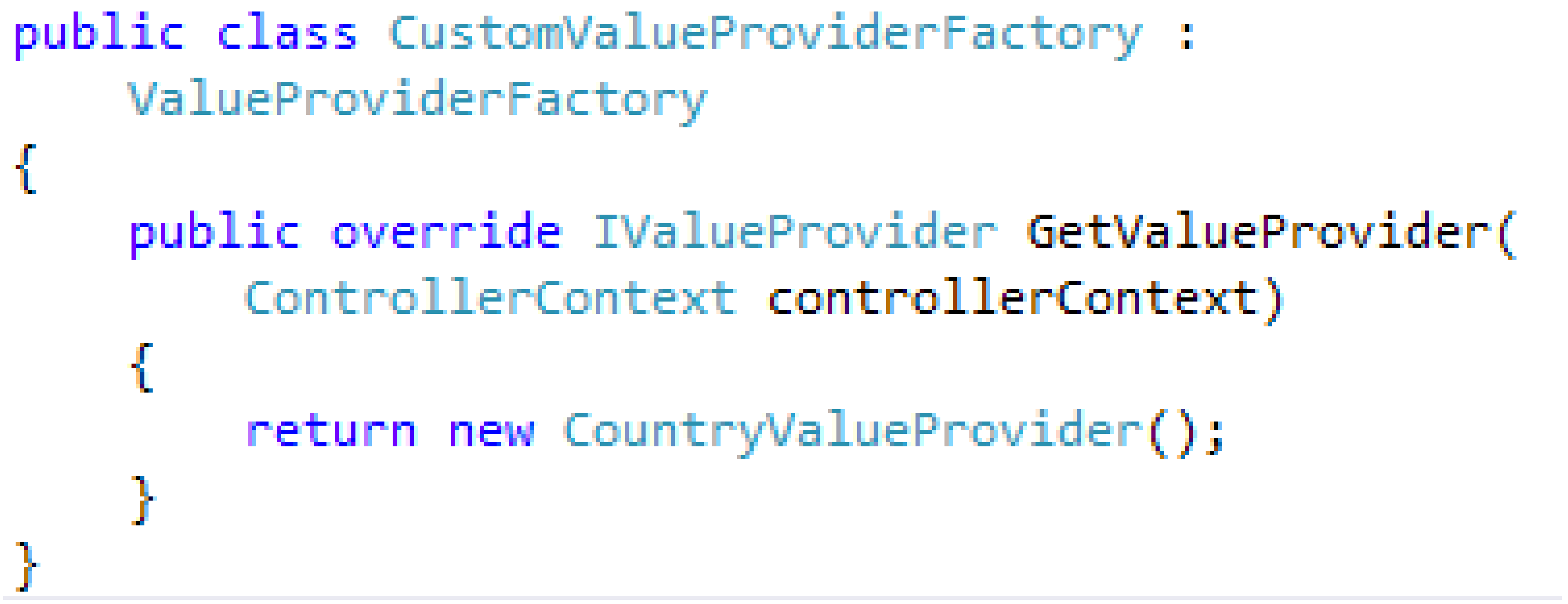
- ContainsPrefix – checks if the value provider can resolve the data for a given prefix.
- GetValue – returns a value for a given data key, or null.

## Custom Binding Mechanics

```
public class CountryValueProvider : IValueProvider
{
    public bool ContainsPrefix(string prefix)
    {
        return prefix.ToLower()
            .IndexOf("country", StringComparison.Ordinal) > -1;
    }

    public ValueProviderResult GetValue(string key)
    {
        if (ContainsPrefix(key))
        {
            return new ValueProviderResult("Belarus", "Belarus",
                CultureInfo.InvariantCulture);
        }

        return null;
    }
}
```

## Custom Binding Mechanics

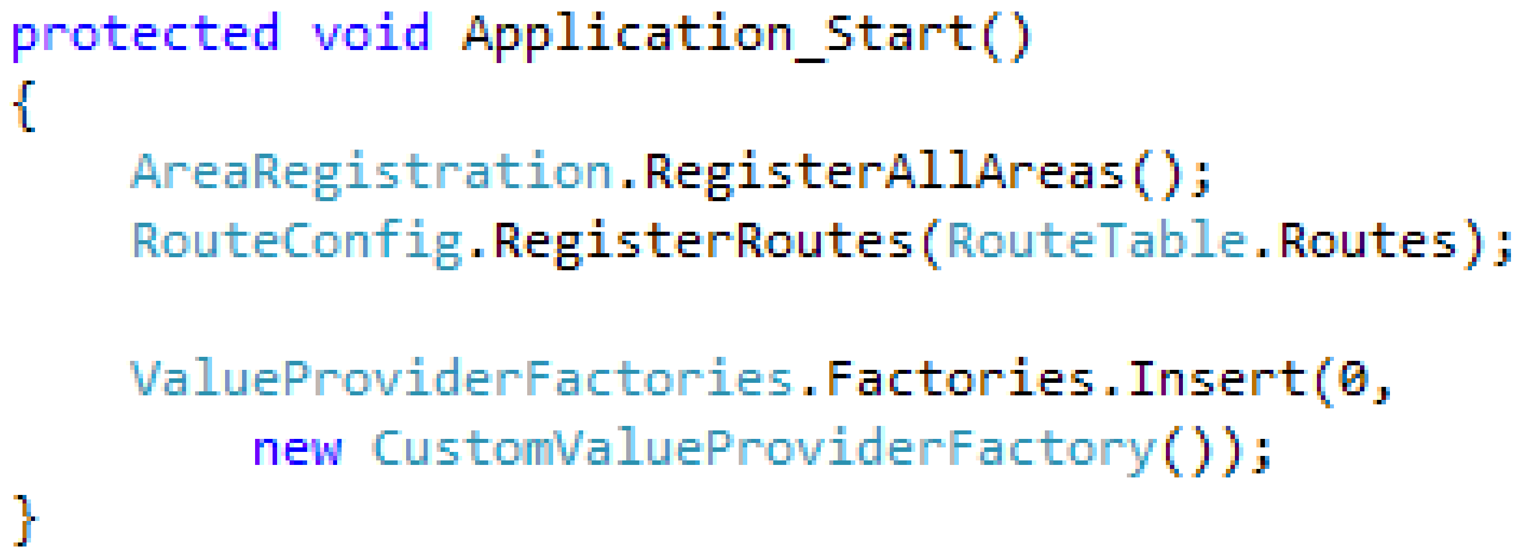
Drive new factory from base abstract ValueProviderFactory class:



```
public class CustomValueProviderFactory :
    ValueProviderFactory
{
    public override IValueProvider GetValueProvider(
        ControllerContext controllerContext)
    {
        return new CountryValueProvider();
    }
}
```

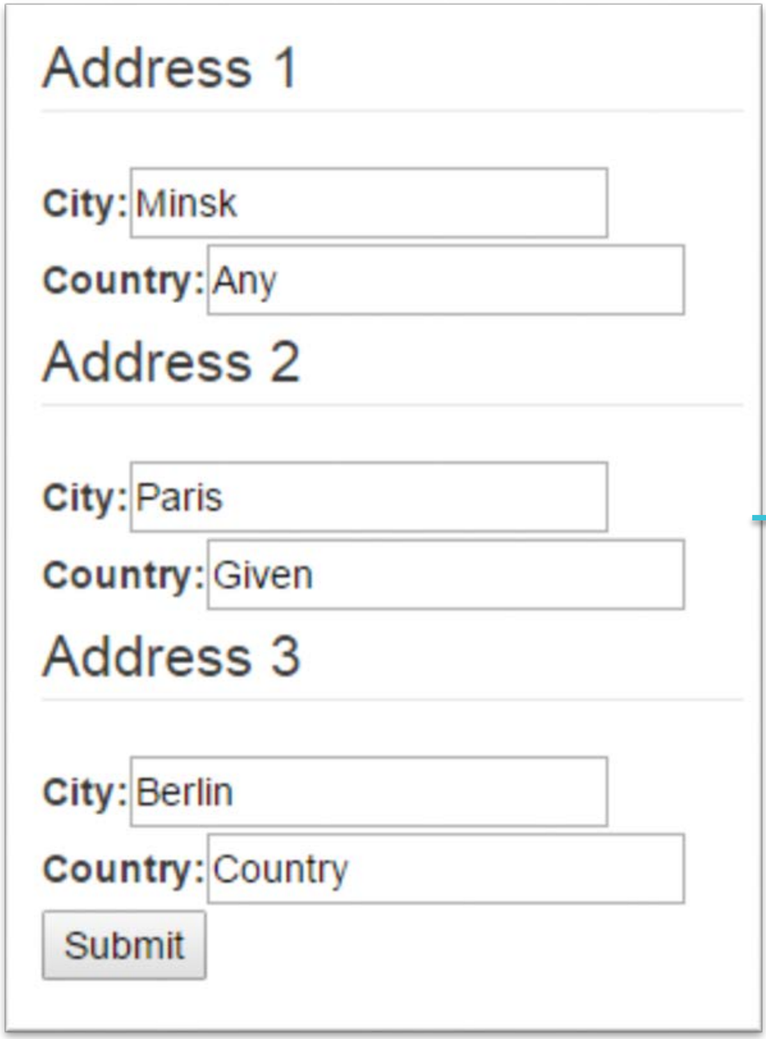
## Custom Binding Mechanics

Add new factory in Global.asax:



```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    RouteConfig.RegisterRoutes(RouteTable.Routes);

    ValueProviderFactories.Factories.Insert(0,
        new CustomValueProviderFactory());
}
```

## Custom Binding Mechanics


Address 1

City: Minsk

Country: Any

Address 2

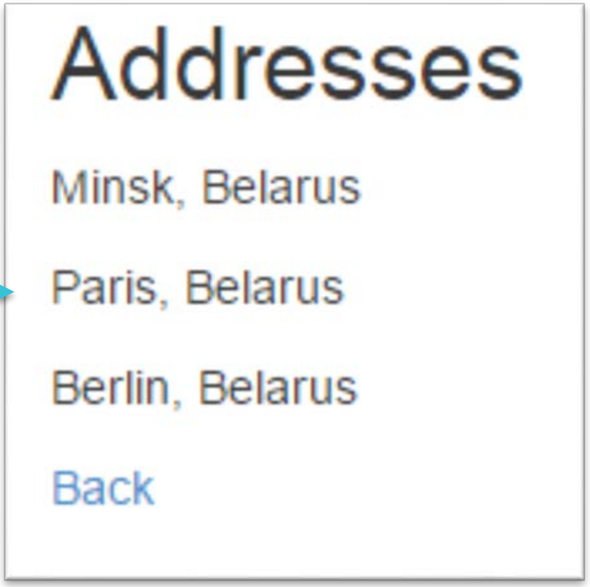
City: Paris

Country: Given

Address 3

City: Berlin

Country: Country


Addresses

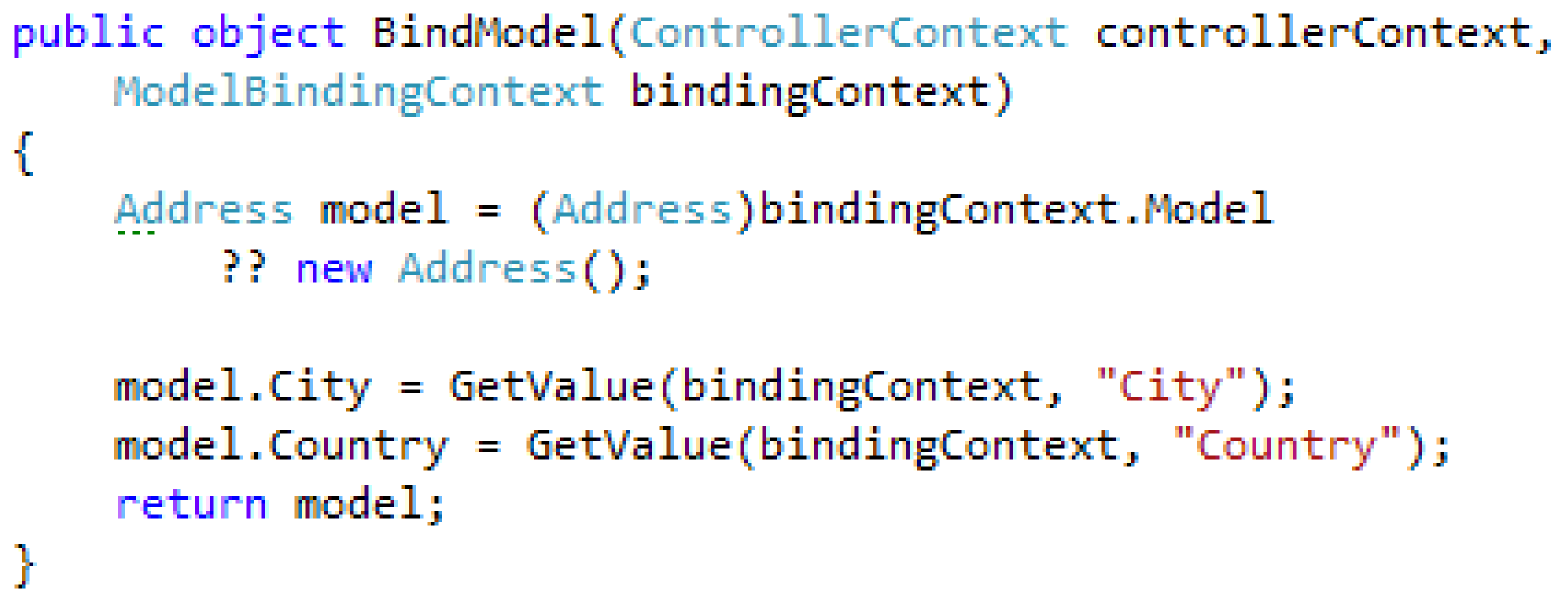
Minsk, Belarus

Paris, Belarus

Berlin, Belarus

## Custom Model Binder

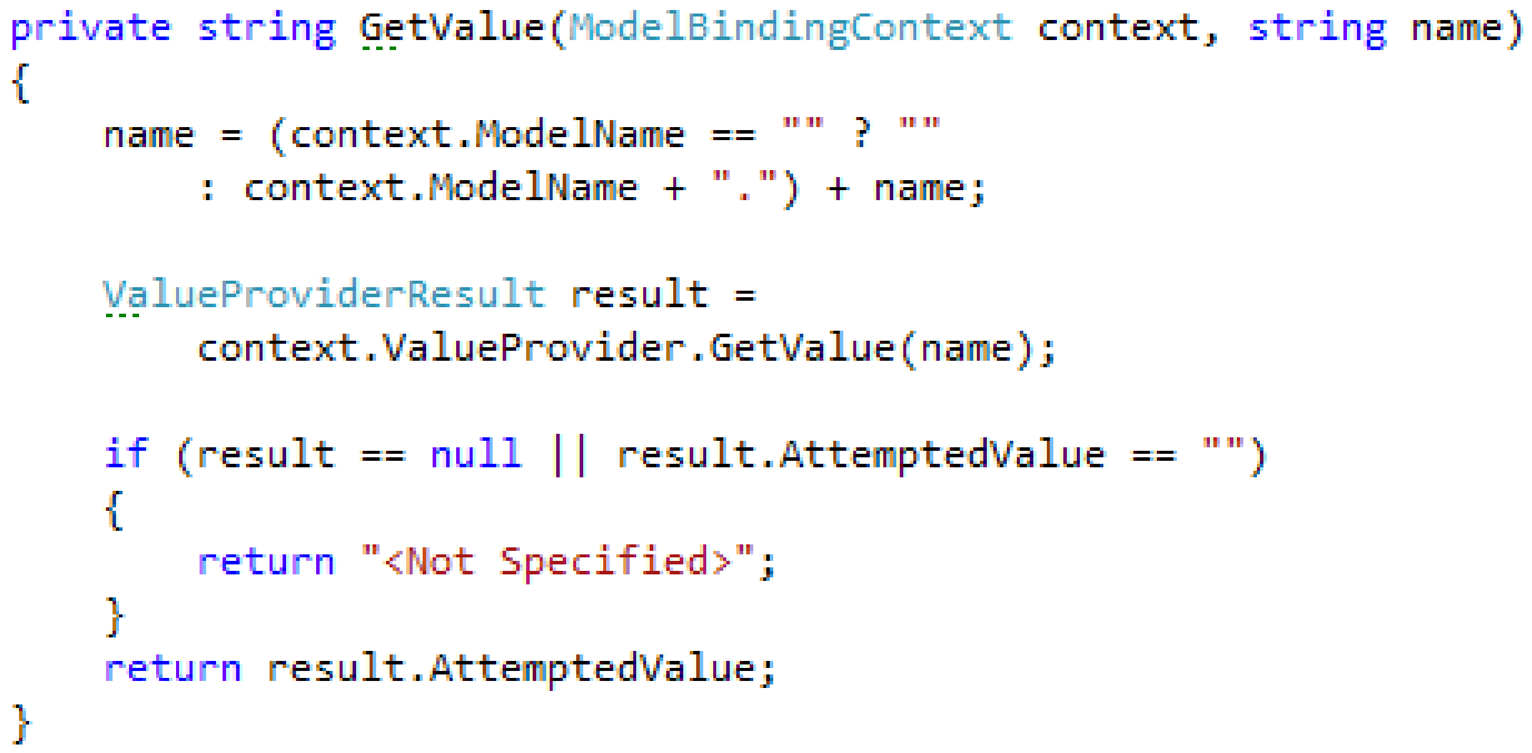
To create custom model binder we need to implement IModelBinder interface:



```
public object BindModel(ControllerContext controllerContext,
    ModelBindingContext bindingContext)
{
    Address model = (Address)bindingContext.Model
        ?? new Address();

    model.City = GetValue(bindingContext, "City");
    model.Country = GetValue(bindingContext, "Country");
    return model;
}
```

# Custom Model Binder

To create custom model binder we need to implement IModelBinder interface:



```
private string GetValue(ModelBindingContext context, string name)
{
    name = (context.ModelName == "" ? ""
        : context.ModelName + ".") + name;

    ValueProviderResult result =
        context.ValueProvider.GetValue(name);

    if (result == null || result.AttemptedValue == "")
    {
        return "<Not Specified>";
    }
    return result.AttemptedValue;
}
```

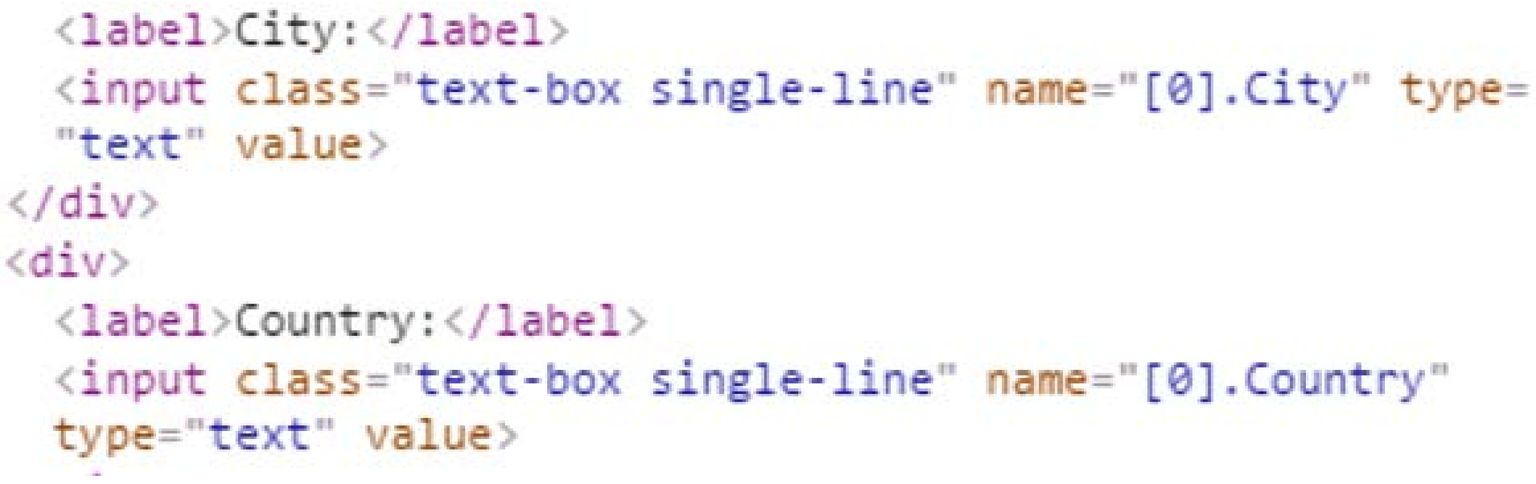
## Custom Model Binder

MVC Framework will call the BindModel method when it wants an instance of the model type that the binder supports. The AddressBinder class will only be used to create instances of the Address class.

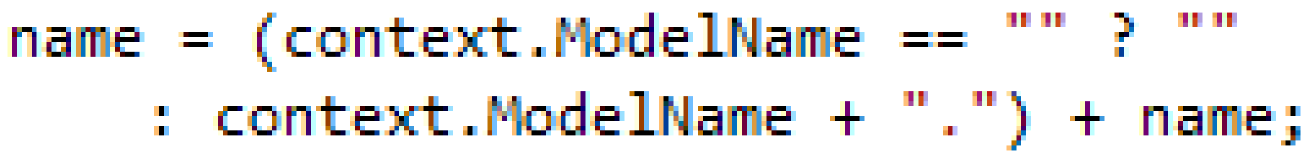
## Custom Model Binder

1. Check to see if the Model property of the ModelBindingContext object has been set.
2. Trying to retrieve City and Country properties by calling the GetValue method.
3. GetValue method use the IValueProvider implementation obtained from theModelBindingContext.ValueProvider property to get values.

# Custom Model Binder



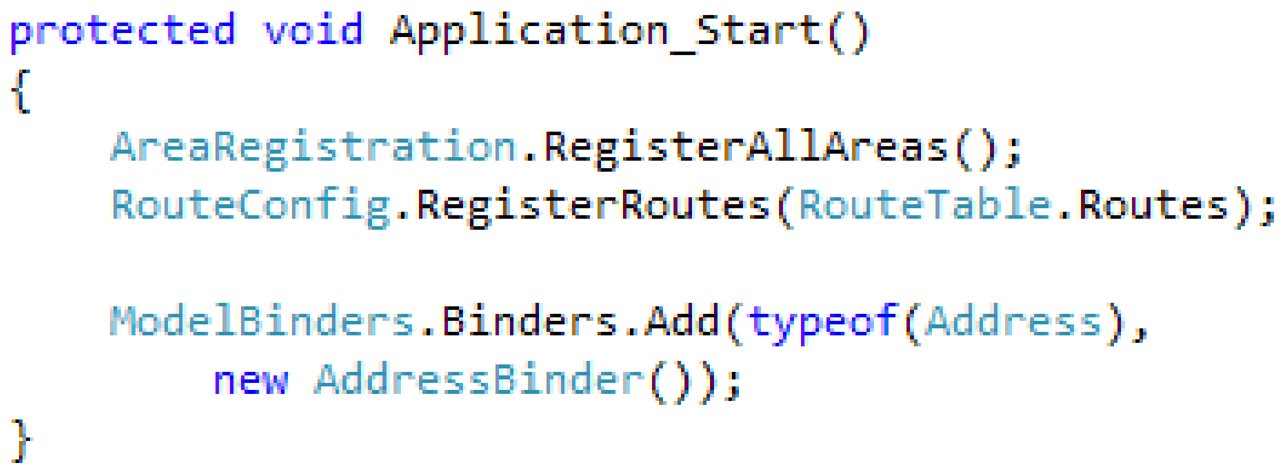
```
  <label>City:</label>
  <input class="text-box single-line" name="[0].City" type=
  "text" value>
</div>
<div>
  <label>Country:</label>
  <input class="text-box single-line" name="[0].Country"
  type="text" value>
```



```
name = (context.ModelName == "" ? ""
    : context.ModelName + ".") + name;
```

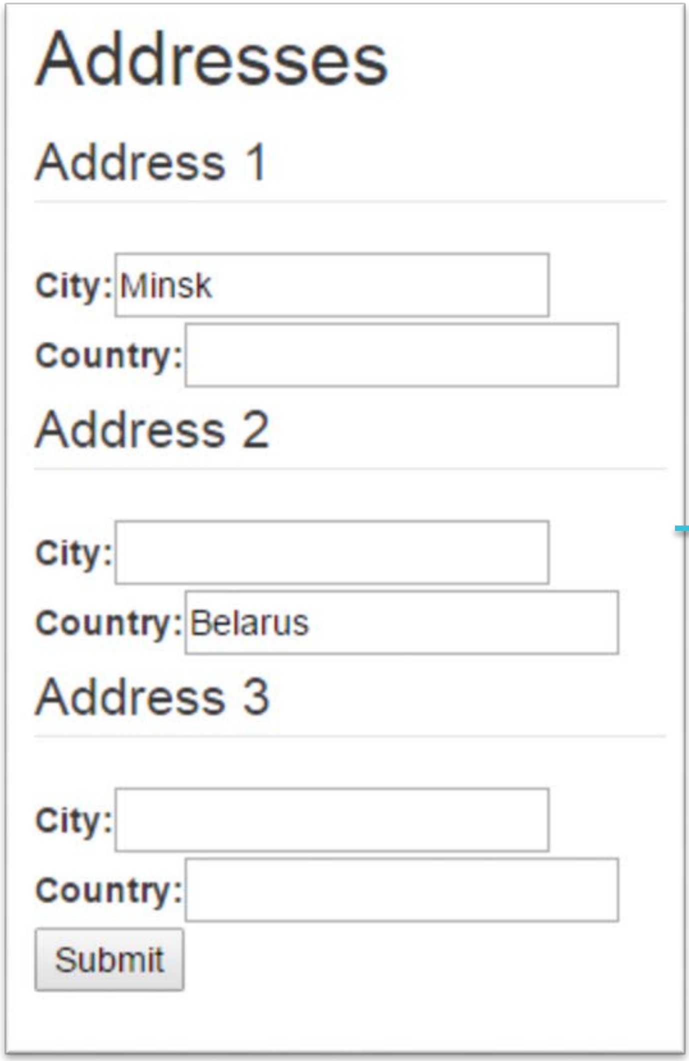
# Custom Model Binder

Registering AddressBinder in Golbas.asax:



```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    RouteConfig.RegisterRoutes(RouteTable.Routes);

    ModelBinders.Binders.Add(typeof(Address),
        new AddressBinder());
}
```

# Custom Model Binder



Addresses

Address 1

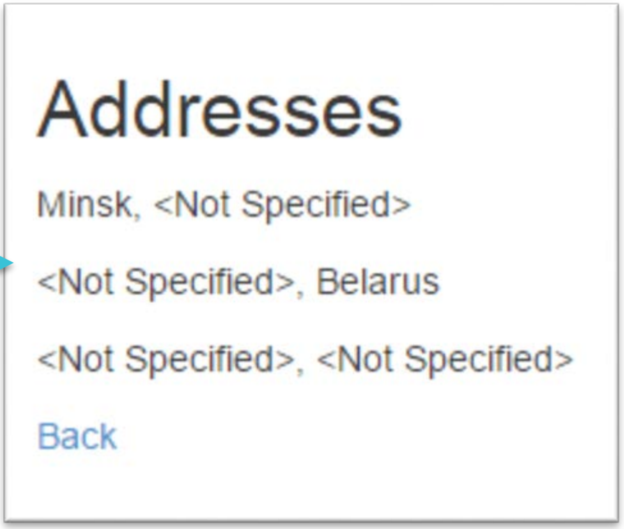
| | |
|---|---|
| City: | Minsk |
| Country: | |

Address 2

| | |
|---|---|
| City: | |
| Country: | Belarus |

Address 3

| | |
|---|---|
| City: | |
| Country: | |



Addresses

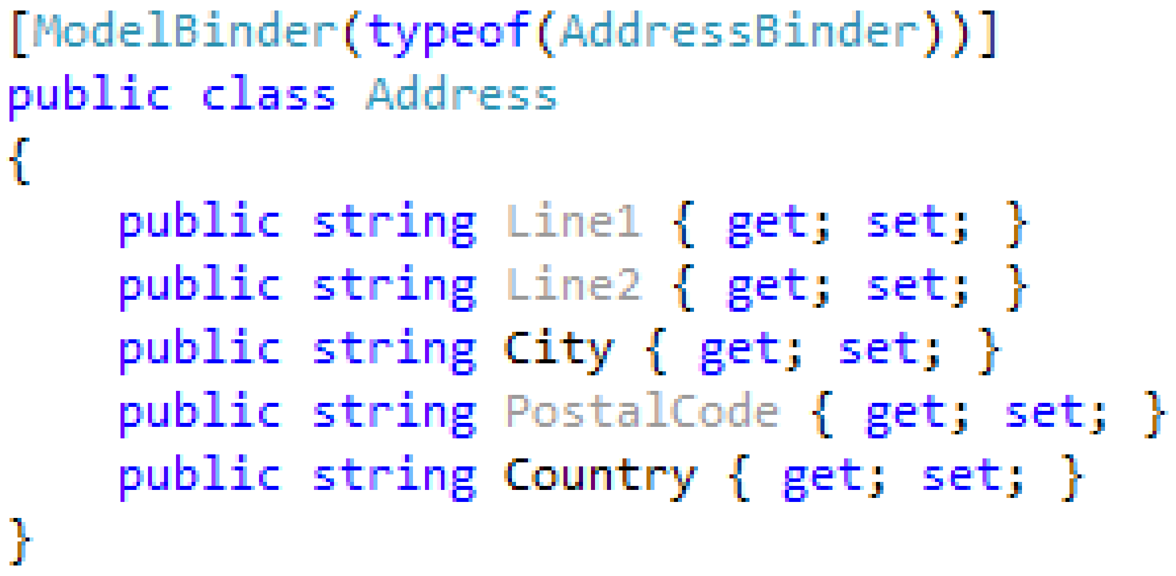
Minsk, <Not Specified>

<Not Specified>, Belarus

<Not Specified>, <Not Specified>

## Custom Model Binder

Registering AddressBinder with an attribute:



```
[ModelBinder(typeof(AddressBinder))]
public class Address
{
    public string Line1 { get; set; }
    public string Line2 { get; set; }
    public string City { get; set; }
    public string PostalCode { get; set; }
    public string Country { get; set; }
}
```

# Task

**Datamapping should be restricted to form data provider only**

- First Name, Last Name
- DoB - input text should work with uncommon date format.
- Role - map guest if not specified, change admin to user if not local
- Address lines should map '<not-defined>' if it contains 'PO BOX‘
- If line2 is empty populate as '<not-defined>‘
- If Postal code is less then 6 chars then map it as '<not-defined>‘
- Add 'address summary' property to the mapping. It should be either automapped as ‘PostalCode City, Line1' or 'No personal address'

**Implement second scenario using only data from the querry string.**