# Today's Agenda

- Exercises - questions
- URL Routing: Group exercises
- Break

- Outgoing URLs in Views
- Model Binding
- Exercises

# URL Routing

# Intro

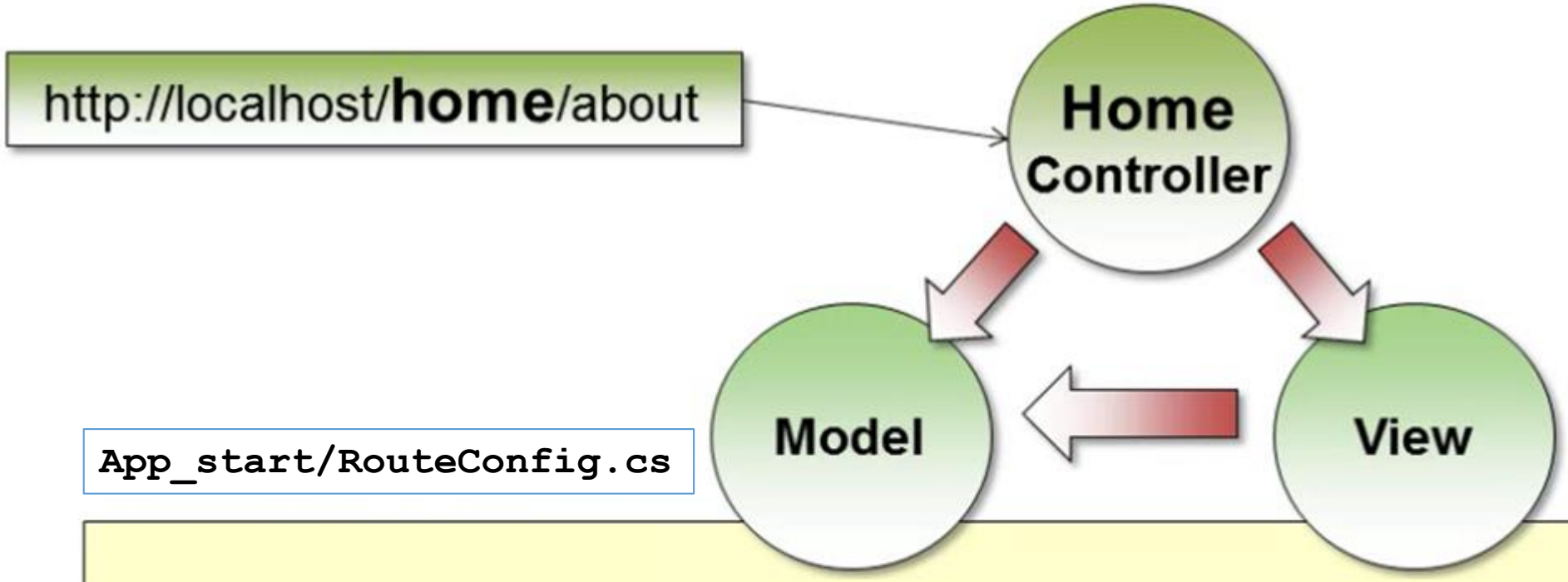
- Usability expert Jakob Nielsen (www.useit.com) urges developers to pay attention to URLs and provides the following guidelines for high-quality URLs. You should provide:
  - A domain name that is easy to remember and easy to spell
  - Short URLs
  - Easy-to-type URLs
  - URLs that reflect the site structure
  - Easy to understand. URLs that are hackable to allow users to move to higher levels of the information architecture by hacking off the end of the URL
  - Persistent URLs, which don't change

- Traditionally , in many web frameworks such as Classic ASP, JSP, PHP, and ASP.NET, the URL represents a physical file on disk:
  - http://example.com/albums/list.php
  - http://example.com/albums/list.aspx

- These URLs are not always "pretty":
  - http://example.com/albums/list.aspx?catid=17313&genreid=33723&page=3

# ASP.NET Route Engine

- Is a separate API that is used by the ASP.NET MVC framework
- Maps requests to executable code
- Maps URLs to classes (of type Controller) and (Action) method calls



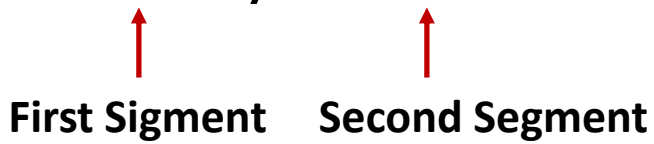

App_start/RouteConfig.cs

```
routes.MapRoute(
    name: "Books", // Route name
    url: "{controller}/{action}/{id}", // URL with parameters
    defaults: new { controller = "Home", action = "Index“,
        id = UrlParameter.Optional } // Defaults
);
```

# Segments in an URL example

- URL with segments:

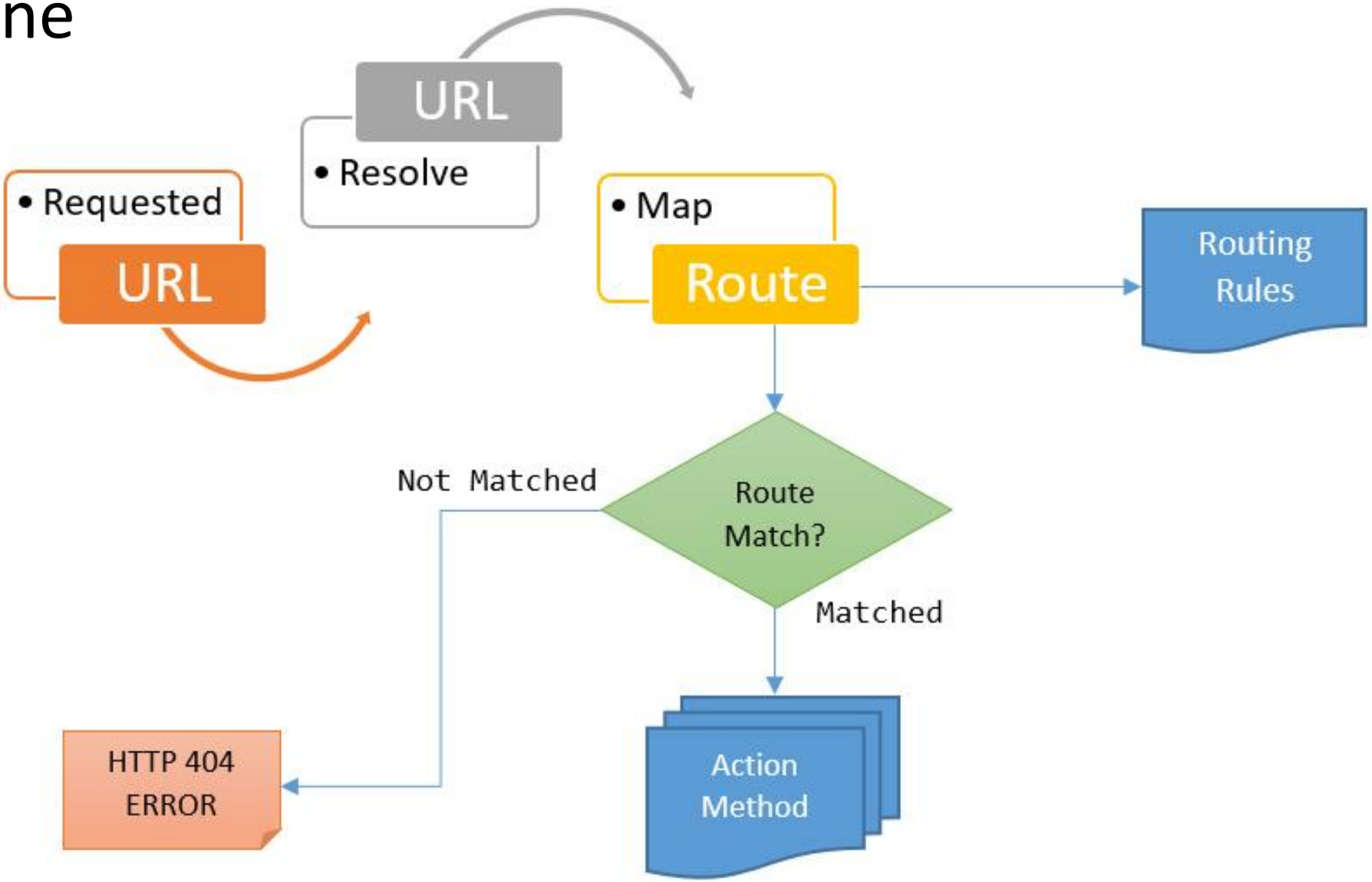
```
http://mysite.com/Admin/Index
```




- Routes transforms URLs to method calls (default setup):

```
{controller}/{action}
```

# Route Engine



# Read named parameters in the controller

```
// GET: Home
public ActionResult Index(int id)
{
    return Content(string.Format("The id
parameter is {0}", id));
}
```

HomeController.cs

# Exercise

1. Create groups with 2 persons in each
2. Download: WU-Backend 16a (1wu16bBACK) > Documents > Presentations > lesson_06 presentation_questions.pdf
3. Answer the questions in pairs

# Q1: Explain **RouteConfig.cs** and give examples of URLs matching the route

```
public static void RegisterRoutes(RouteCollection routes) {
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");


    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{id}", // URL with
parameters
        defaults: new { controller = "Home", action = "Index",
id = UrlParameter.Optional } // Defaults
    );
}
```

# Matches

| Request URL | Segment Variables |
|---|---|
| host/ | Controller: Home (default); Action: Index (default) |
| host/Product | Controller: Products; Action: Index (default) |
| host/Product/Book | Controller: Products; Action: Books |
| host/Product/Book/1218 | Controller: Products; Action: Books; Id: 1218 (optional) |
| host/Product/Book/Essays/1218 | No match, too many segments |

# Q2: Setup a Route that has an id as the third required segment

```
public static void RegisterRoutes(RouteCollection routes) {
    ...
    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{id}", // URL with
parameters
        defaults: new { controller = "Home", action = "Index"}
// Defaults
    );
}
```

# Q3: Setup a Route that has an id as an optional segment with “100” as its default value

```
public static void RegisterRoutes(RouteCollection routes) {
    ...
    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{id}", // URL with
parameters
        defaults: new { controller = "Home", action = "Index",
id = "100" } // Defaults
    );
}
```

# Q4: Setup a Route that has an id as the third optional segment that accepts only integers as value

```
using System.Web.Mvc.Routing.Constraints;
public static void RegisterRoutes(RouteCollection routes) {
    ...
    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{id}", // URL
        defaults: new { controller = "Home", action = "Index",
id = UrlParameter.Optional }, // Defaults
        constraints : new { id = new IntRouteConstraint() } //
Constraints
    );
}
```

# Is there more than one way of doing it?

```
using System.Web.Mvc.Routing.Constraints;
public static void RegisterRoutes(RouteCollection routes) {
    ...
    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{id}", // URL
        defaults: new { controller = "Home", action = "Index",
id = UrlParameter.Optional}, // Defaults
        constraints : new { id = @"\d+" } // Constraint with
regular expression
    );
}
```

# Q5: Setup a Route with an id segment that accepts only integers between 10 and 100

```
using System.Web.Mvc.Routing.Constraints;
public static void RegisterRoutes(RouteCollection routes) {
    ...
    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{id}", // URL
        defaults: new { controller = "Home", action = "Index",
id = "1" }, // Defaults
        constraints : new { id = new RangeRouteConstraint(10,
100) } // Constraints
    );
}
```

# Route Constraint Classes

| Name | Description | Attribute Constraint |
|---|---|---|
| AlphaRouteConstraint() | Matches alphabet characters, irrespective of case (A–Z, a–z) | alpha |
| BoolRouteConstraint() | Matches a value that can be parsed into a bool | bool |
| DateTimeRouteConstraint() | Matches a value that can be parsed into a DateTime | datetime |
| DecimalRouteConstraint() | Matches a value that can be parsed into a decimal | decimal |
| DoubleRouteConstraint() | Matches a value that can be parsed into a double | double |
| FloatRouteConstraint() | Matches a value that can be parsed into a float | float |
| IntRouteConstraint() | Matches a value that can be parsed into an int | int |
| LengthRouteConstraint(len)<br>LengthRouteConstraint(min, max) | Matches a value with the specified number of characters or that is between min and max characters in length. | length(len)<br>length(min, max) |
| LongRouteConstraint() | Matches a value that can be parsed into a long | long |
| MaxRouteConstraint(val) | Matches an int value if the value is less than val | max(val) |
| MaxLengthRouteConstraint(len) | Matches a string with no more than len characters | maxlength(len) |
| MinRouteConstraint(val) | Matches an int value if the value is more than val | min(val) |
| MinLengthRouteConstraint(len) | Matches a string with at least len characters | minlength(len) |
| RangeRouteConstraint(min, max) | Matches an int value if the value is between min and max | range(min, max) |

Freeman: Pro ASP NET MVC 5, pp. 403-04

# Q6: Setup a Route that accepts any number of segments

```
public static void RegisterRoutes(RouteCollection routes) {
    ...
    routes.MapRoute(
        name: "Default", // Route name
        url: "{controller}/{action}/{*catchall}" // URL
    );
}
```

CatchAllController.cs

```
public string Index(string catchall)
{
    return "Controller: Question06 <br /> Action: Index <br /> Catchall: " + catchall;

}
```

# Q7: Setup a Route that uses “shop” as static URL segment

- Matched URL’s
  - *host*/shop/books/
  - *host*/shop/books/programming/pro-asp-net-mvc5
  - *host*/shop/movies/action
  - *host*/shop/movies/drama/the-shining

# Q7: Solution – the Route

```
routes.MapRoute(
   // Route name
   name: "Shop",
   // URL with parameters
   url: "shop/{controller}/{category}/{title}",
   // Defaults
   defaults: new { action = "Index",
         category = UrlParameter.Optional,
         title = UrlParameter.Optional }
);
```

# The controller

```
public class BooksController : Controller {
    public ActionResont Index(string category = "", string title = "") {
        string s = "Books controller";
        if (category.Length > 0) {
            s += "<br>Category: " + category;
        }
        if (title.Length > 0) {
            s += "<br>Title: " + title;
        }
        return Content(string.Format("The action result is sent from:<br/> {0}", s));
        }
}
```

# Q8: What is the correct order of routes when you have multiple routes?

- The route system matches incoming URLs against the URL pattern of routes in the sequence in which they are defined starting with number one.

  =>

- The most specific routes must be defined first!
- The most general routes must be defined last!

# Q9: From which file in an ASP.NET MVC application is the `RegisterRoutes` method in the `RouteConfig` class called?

- **`Global.asax`**

```
public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
```

# Q10: What is attribute routing and how do you enable it?

- Enables you to specify routing as attributes on Controllers and Action Methods
- Enabling (`RouteConfig.cs`):

```
public class RouteConfig
{
   public static void RegisterRoutes(RouteCollection routes)
    {
         routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
         routes.MapMvcAttributeRoutes();
         ...
     }
}
```

# Q11: The Route

Setup an attribute route that maps URLs like:

- /books
- /books/1430265299

```
public class BooksController : Controller {

    [Route("books/{isbn?}")]
    public ActionResult BookView(string isbn) {
        if (!String.IsNullOrEmpty(isbn)) {
            return View("OneBook", GetBook(isbn));
        }
        return View("AllBooks", GetBooks());
    }
}
```

# Q12: Pros and cons of attribute routing?

| | Pro | Cons |
|---|---|---|
| **Convention-based Routing** | • Separation of concerns – controllers have no knowledge or dependency on routing configuration<br>• All routing info in one file! | • More work to setup |
| **Attribute Routing** | • Easier to grasp & understand | • No separation of concerns |

# Examples of Valid Route Patterns in ASP.NET MVC

| Route Pattern | URL Example |
|---|---|
| `mysite/{username}/{action}` | `~/mysite/jatten/login` |
| `public/blog/{controller}-{action}/{postId}` | `~/public/blog/posts-show/123` |
| `{country}-{lang}/{controller}/{action}/{id}` | `~/us-en/products/show/123` |
| `products/buy/{productId}-{productName}` | `~/products/buy/2145-widgets` |

# Outgoing URLs in Views

# @Html.ActionLink

```
// Action
@Html.ActionLink("Same controller and named action", "index")
http://localhost:2405/OutGoingURL


// Action + paramters
@Html.ActionLink("Same controller and action with URL parameters", "index",
new { page = 2, sortorder = "Author" })
http://localhost:2405/OutGoingURL?page=2&sortorder=Author


// Action + controller
@Html.ActionLink("Another controller and action", "AnotherAction",
"AnotherController")
http://localhost:2405/AnotherController/AnotherAction
```

# @Html.ActionLink

```
// Action + named segment

@Html.ActionLink("Same controller with extra segment", "index", new
{ id = "1012" }, null)

http://localhost:2405/OutGoingURL/index/1012



// Action + controller + named segments

@Html.ActionLink("Another controller with extra segments", "index",
"books", new { category = "Fiction", title = "Crime-and-Punishment"
}, null)

http://localhost:2405/index/books/Fiction/Crime-and-Punishment
```

# Call a named Route

```
@Html.RouteLink(
      "RouteLink with named route", // link name
      "Default", // Route name
      new { controller = "list", action = "index" } // Controller
)
```

# External links

```
@Html.ActionLink(
   "External Link",
   "Index", // Action method
   "Home", // Controller
   "https", // Protocol
   "myserver.mydomain.com", // Domain
   "segment", // Html segment
   new { id = "MyId" }, // additional segments
   new { id = "myAnchorID", @class = "myCSSClass" }
   // HTML attributes
)
```

# Bypass the Routing System

- An example:

```
Routes.IgnoreRoute("StaticContent/{filename}.html");
```

- Default setup:

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

# Model Binding

# Binding to Simple Types

```
// incoming URL: /Index1/24 || /Index1?id=24
public string Index1(int id) { // Required
     return id.ToString();
}
// incoming URL: /Index2 || /Index2/24 || /Index2?id=24
public string Index2(int? id) { // Nullable
     return id.ToString();
}
// incoming URL: /Index3 || /Index3/24 || /Index3?id=24
public string Index3(int id = 12) { // Default
     return id.ToString();
}
```

BindSimpleTypesController

# The Default Model Binder in ASP.NET MVC `System.Web.Mvc.DefaultModelBinder`

- Maps a browser request to a data object

- Where the Default Model Binder Looks for Data

| Order | Source | Description |
|---|---|---|
| 1. | `Request.Form` | Values provided by the user in HTML form elements |
| 2. | `RouteData.Values` | The values obtained using the application routes |
| 3. | `Request.QueryString` | Data included in the query string portion of the request URL |
| 4. | `Request.Files` | Files that have been uploaded as part of the request |

# Binding to Complex Types


```
[HttpPost]
public ViewResult Create(EditRecipeViewModel model)
{
    // ...
}
```

# Customer Class Example (the model)

```
public class Person {
   public int PersonId { get; set; }
   public string Name { get; set; }
   public string Email { get; set; }
}
```

# Person class example (the view)

```
<h2>Person</h2>
@using (Html.BeginForm()) {
    <div>
       @Html.LabelFor(m => m.PersonId)
       @Html.EditorFor(m => m.PersonId)</div>
    <div>
       @Html.LabelFor(m => m.Name)
       @Html.EditorFor(m => m.Name)</div>
    <div>
       @Html.LabelFor(m => m.Email)
       @Html.EditorFor(m => m.Email)</div>
    <div>
   <label> </label><input type="submit" value="Create" /> </div>
}
```

# Person class example (the controller)

```
public ActionResult Index()
{
   return View();
}

[HttpPost]
public ActionResult Index(Person p) {
   return View(p);
}
```

BindComplexTypesController.cs

# Customer Class Example

## View

```
1   @model lesson06_examples.Models.Person
2
3   <h2>Person</h2>
4
5   @using (Html.BeginForm()) {
6
7       <div>@Html.LabelFor(m => m.PersonId) @Html.EditorFor(m => m.PersonId)</div>
8       <div>@Html.LabelFor(m => m.Name) @Html.EditorFor(m => m.Name)</div>
9       <div>@Html.LabelFor(m => m.Email) @Html.EditorFor(m => m.Email)</div>
10      <div><label> </label><input type="submit" value="Create" /> </div>
11  }
12
```

**Person**

PersonId 1
Name Peter
Email Nielsen
Create

## Model

```
6   namespace lesson06_examples.Models {
7       public class Person {
8           public int PersonId { get; set; }
9           public string Name { get; set; }
10          public string Email { get; set; }
11      }
12  }
```

## Controller

```
10      public class BindingComplexTypeController : Controller
11      {
12          // GET: BindingComplexType
13          public ActionResult Index()
14          {
15              return View();
16          }
17
18
19          [HttpPost]
20          public ActionResult Index(Person p) {
21              return View(p);
22          }
23      }
24  }
```

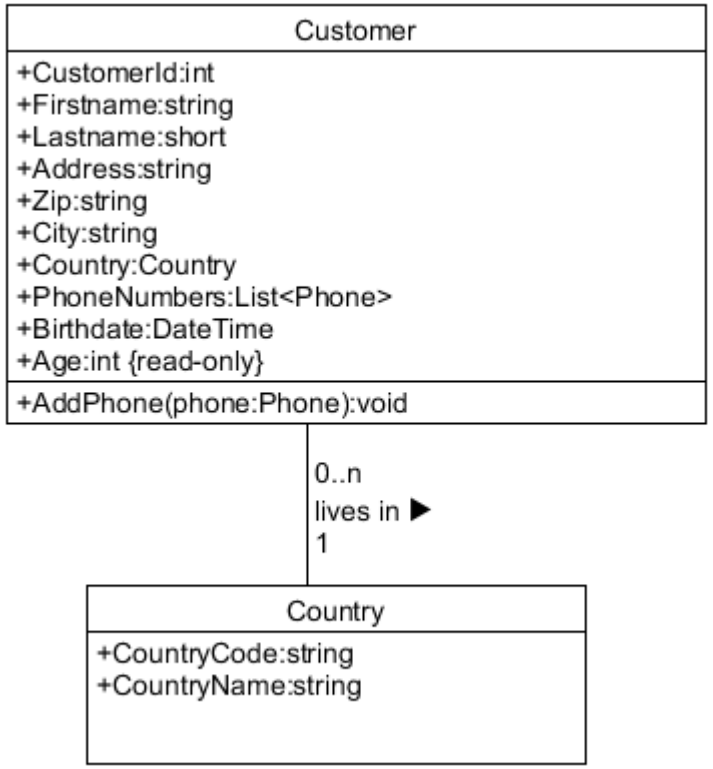
p {lesson06_examples.Models.Person}
Email "Nielsen"
Name "Peter"
PersonId 1

# ViewModels

# Binding complex types with ViewModels – in folder **ViewModels**

```
public class CustomerInfo { // viewmodel class
    public Customer Customer { get; set; }
    public List<Country> Countries { get; set; }

    public CustomerInfo() { // constructor
        Customer = new Customer();
         List<Country> Countries = new List<Country>();
    }
}
```



Use a ViewModel
when your view/form
does not map
to a single class
in the domain model

# Exercises

Mandatory assignment 2: Building a Shopping Cart for MbmStore