MVC II

Slavomír Moroz
2015

Topics

- ViewData
- Routing
- Razor blocks syntax
- Model binding
- Model validation
- Templates
- AntiForgeryToken

ViewData (ViewDataDictionary Class)

- Represents a container that is used to pass data between a controller and a view
- Controllers writes the data, view reads.

- ViewData.Model – passed model
- ViewData.ModelMetadata – set o information about model
- ViewData.ModelState – validation messages
- ViewData[“something”] – additional data
  - also accessible via ViewBag.

https://msdn.microsoft.com/en-us/library/system.web.mvc.viewdatadictionary(v=vs.118).aspx

RouteData

- Encapsulates information about a route.

```
URL: [domain:port]/en-us/help

routes.MapRoute(
    name: "Default",
    url: "{culture}/{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
);

if (RouteData.Values.TryGetValue("culture", out culture))
{
    var cultureInfo = new CultureInfo(culture as string);
}
```

https://msdn.microsoft.com/en-us/library/system.web.routing.routedata(v=vs.118).aspx

Route constraints

- If a URL contains values that are outside the constraints for a route, that route is not used to handle the request.

Regex constraints
- Defined with string value
- `new { number = "[1-9][0-9]*" }`

C# constraints
- Object that implements IRouteConstraint
- Predefined constrains located in System.Web.Mvc.Routing.Constraints namespace
- `new { number = new IntRouteConstraint() }`

```
routes.MapRoute(
    name: "Home-ShowNumber",
    url: "{number}",
    defaults: new { controller = "Home", action = "ShowNumber" },
    constraints: new { number = "[1-9][0-9]*" }
);
```

https://msdn.microsoft.com/en-us/library/cc668201.aspx#adding_constraints_to_routes

Attribute routing

- RouteAttribute - Place on an action to expose it directly via a route.
- RoutePrefixAttribute - Annotates a controller with a route prefix that applies to all actions within the controller.

Initialization

- In `App_Start\RouteConfig.cs`

```
routes.MapMvcAttributeRoutes();
```

Usage

```
public class CategoryController : Controller
{
    [Route("kategorie/{category:int}/{subCategory:int?}")]
    public ActionResult Detail(int category, int? subCategory)
    {
        …
    }
}
```

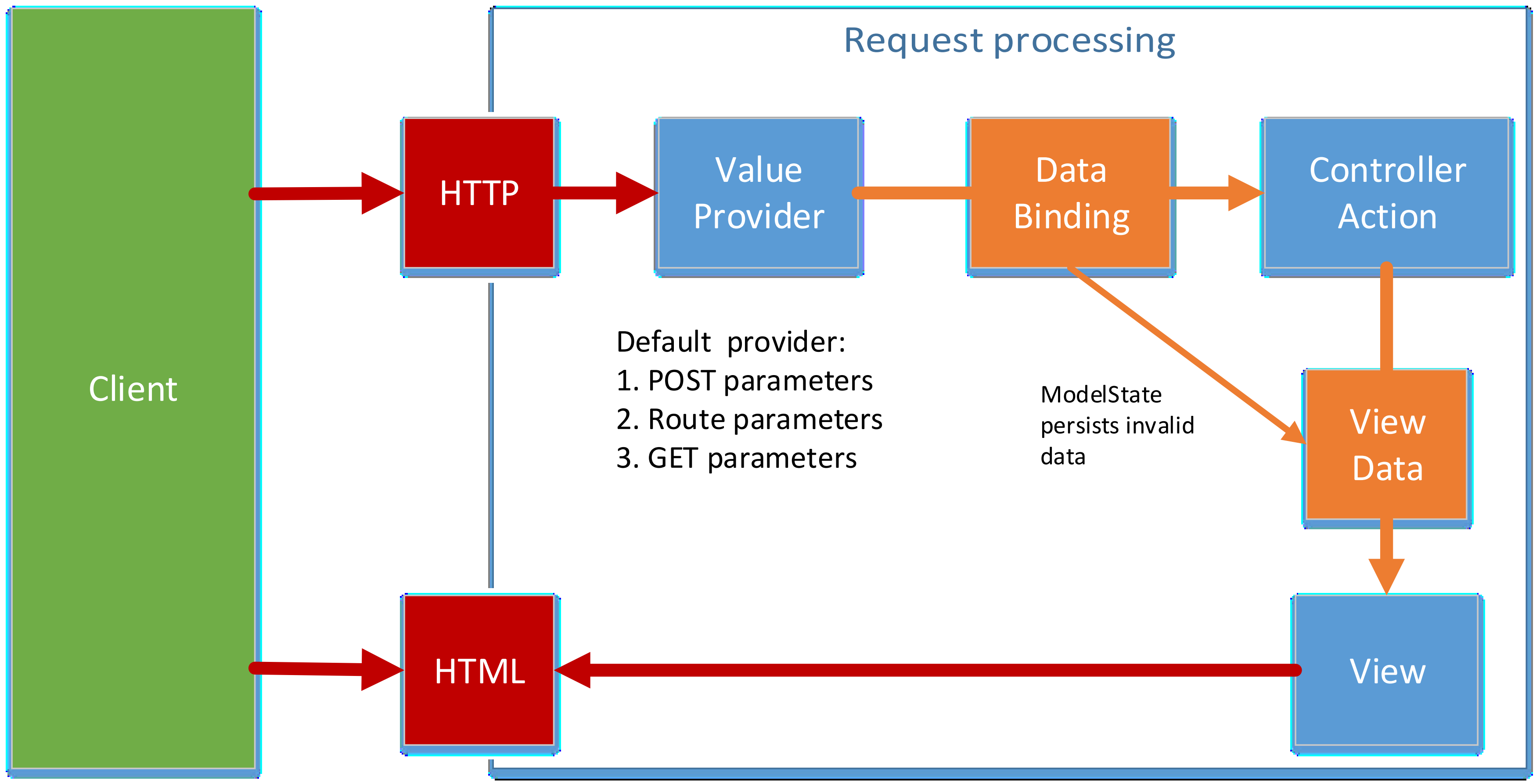
http://blogs.msdn.com/b/webdev/archive/2013/10/17/attribute-routing-in-asp-net-mvc-5.aspx

Razor syntax - blocks

```
@if (true) {
    WriteLiteral("<p>Test</p>");
}

@if (true) {
    <p>Text</p>
}

@if (true) {
    @:This is text.
}

@if (true)
{
    <Text>This is also text.</Text>
}
```

```
@if (condition1) {
    if (condition2) { <p>Text</p> }
}


@if (condition1) {
    <div>
        @if (condition2) { @:Text }
    </div>
}
```

Model binding - passing data



Default provider:
1. POST parameters
2. Route parameters
3. GET parameters

Partial binding

- Technique where only a subtree of view model is sent to the server

Model

```
public class CreateBookModel
{
    public string[] Genres { get; set; }
    public string[] Authors { get; set; }
    public Book Item { get; set; }
}
```

PostData

```
Item.Author=John Smith
Item.Title=Johns book
Item.Price=8
Item.Genre=Fantasy
```

Binding

- `public ActionResult Create([Bind(prefix = "Item")] Book book) {…}`
- `UpdateModel(book, "Item");`

Collections binding

Primitive type array

```
ActionResult Edit(string[] array) {…}
```

PostData

```
array="John"
array="Mark"
array="Zoey"
```

Index array (complex type)

```
ActionResult Edit(Employee[] array) {…}
```

PostData

```
array[0].FirstName="John"
array[0].LastName="Smith"
array[1].FirstName="Zoey"
array[1].LastName="Castillo"
```

Dictionary

```
ActionResult Edit(
    Dictionary<string, Employee> empls
)
```

PostData

```
employees[Emp1035].FirstName="John"
employees[Emp1035].LastName="Smith"
employees[Emp2535].FirstName="Zoey"
employees[Emp2535].LastName="Castillo"
```

Model validation (server)

Setup

- Data annotations validation attributes
  - Required, DisplayName, StringLength, Range…
  - Custom attribute that inherits ValidationAttribute.
- Model implementing IValidatableObject
- Custom : ViewData.ModelState.AddModelError()

Check

```
if (!ViewData.ModelState.IsValid)
{
    return View(model);
}

repository.Save();
return RedirectToAction("Detail", new { id = id });
```

View

```
Html.ValidationMessageFor(…)
Html.ValidationSummary()
```

Model validation (client)

- Unobtrusive validation (linked with JQuery)
- Supports only attribute validators (doesn’t support IValidatableObject)
- Hard to localize (JQuery globalize project)

Setup

- Install nugget package Microsoft.JQuery.Unobtrusive.Validation
- Link scripts in your layout:
    - JQuery
    - JQuery-validate
    - JQuery-validate-unobtrusive

http://bradwilson.typepad.com/blog/2010/10/mvc3-unobtrusive-validation.html
http://jqueryvalidation.org/documentation/
https://github.com/jquery/globalize

Templates

- You can create custom templates for displaying or editing objects
- Templates must be placed in folder
    - DisplayTemplates
    - EditorTemplates

- Rendered with command
    - Html.DisplayFor()
    - Html.EditorFor()

- How is template selected
    1. Explicit
    2. [DataType] attribute
    3. By type

(Example: see FilterIndexedArrayWithTemplate.cshtml & views under Shared folder in demo app)

AntiForgeryToken

- Protection against CSRF attacks.
- Render token in form
  - Html.AntiForgeryToken()
- Validation in controller
  - [ValidateAntiForgeryToken] attribute

https://en.wikipedia.org/wiki/Cross-site_request_forgery