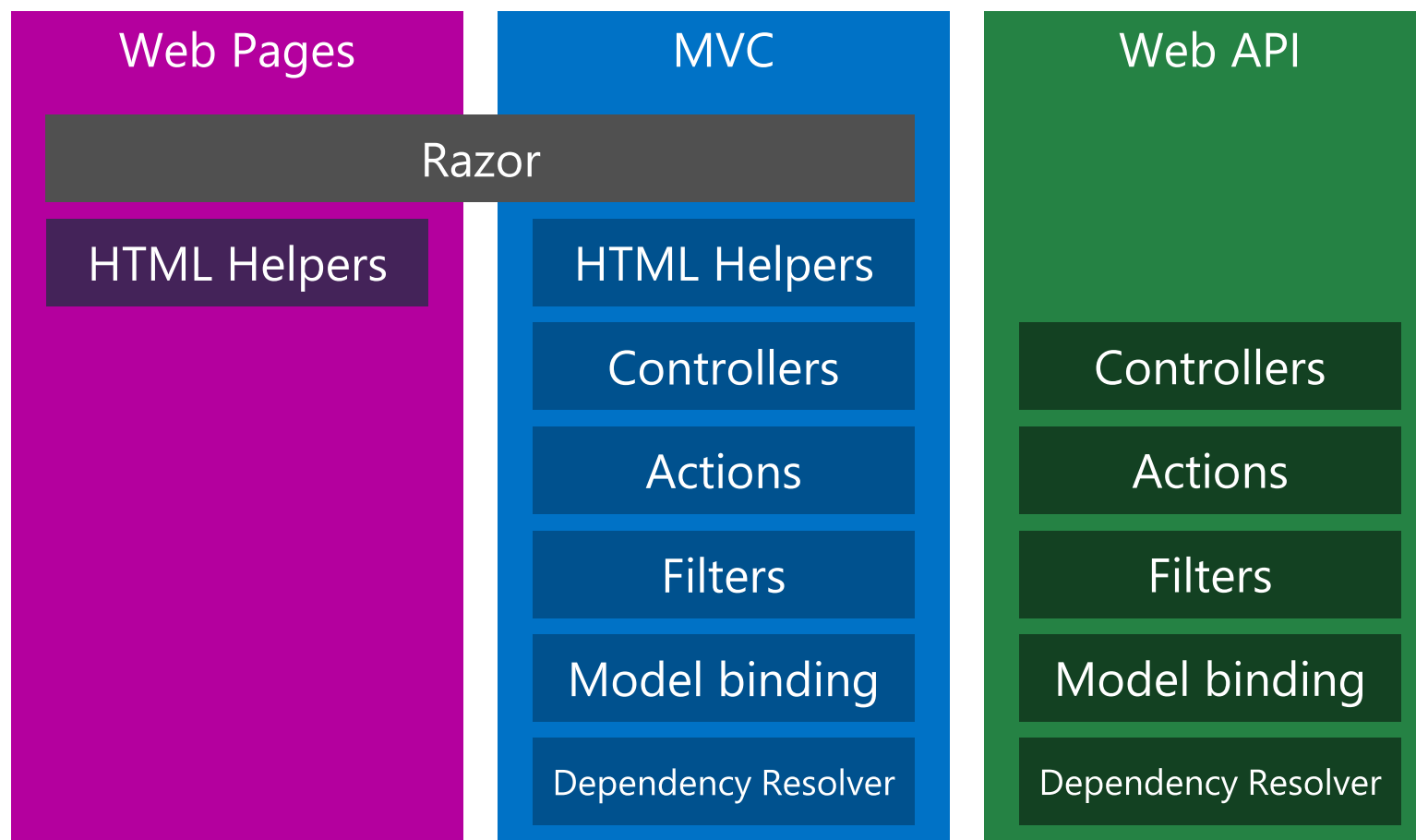


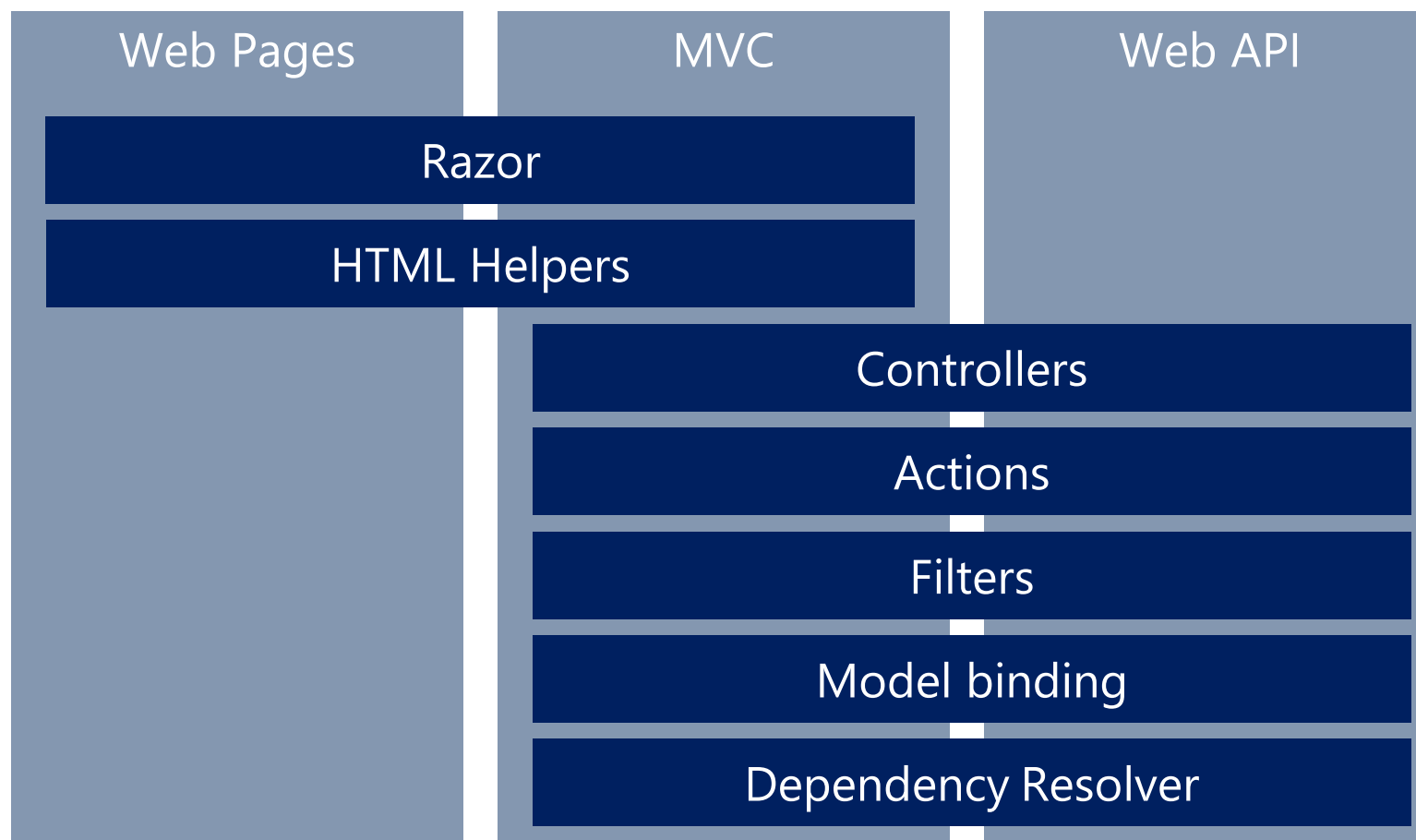
# ASP.NET Core MVC

Ing. Jan Janoušek

# ASP.NET MVC <= 5



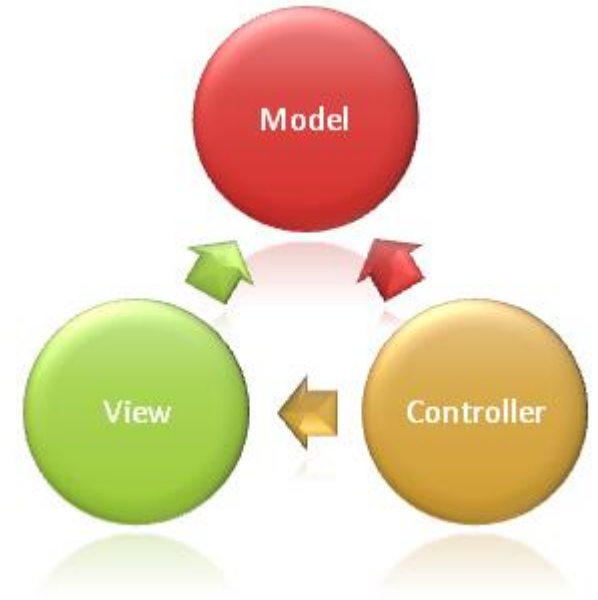
# ASP.NET Core & MVC 6



MVC + Web API + Web Pages = ASP.NET MVC 6

# MVC – obecně

- **Model** – business logika – například přístup k DB, atd... - „data“
- **View** – prezentační vrstva – jak se mají data zobrazit
- **Controller** – obsluha interakce uživatele. Pracuje s modelem a řídí výběr šablony.
- **Router** – není část MVC (jako návrhového vzoru). Slouží k mapování URL na controllery a akce.



# Controller

- Libovolná třída, která dědí z abstraktní třídy „*Controller*“.
- Musí být uvnitř adresáře „*Controllers*“ (je možné přenastavit).
- Název třídy musí končit suffixem „*Controller*“. Například: „*ProductController*“.
- Controllery obsahují „akce“ – metody, které vracejí *IActionResult* (interface).
- Vše uvedené musí být *public*.

```
public class ProductController : Controller
{
    public IActionResult Detail(string id)
    {
        return View();
    }
    public IActionResult List()
    {
        return View();
    }
    public IActionResult Json()
    {
        return new JsonResult(new { Name = "Auto" });
    }
}
```

# Model

- Libovolná třída – záleží pouze na vývojáři...
- Data od klienta (v rámci HTTP požadavku) mohou být automaticky mapována na model a následně validována. A to pomocí speciálních validačních atributů.
- Model lze předat do View jako argument metody „*View(...);*“

```
public class LoginCustomerForm
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Heslo")]
    public string Password { get; set; }

    [Display(Name = "Trvalé přihlášení")]
    public bool RememberMe { get; set; }
}
```

# View

- Soubory s příponou „.cshtml“
- Šablonovací jazyk/engine: „**Razor**“
- Může obsahovat standardní HTML + Razor kód (případně libovolný textový obsah).
- Musí být v adresáři „Views“ (lze změnit).
- Šablony pro akci controlleru musí být v adresáři:  
„Views/{NameOfController}/{NameOfAction}.cshtml“  
Toto chování lze upravit. Šablonu lze specifikovat i manuálně v controlleru.
- Data z controller do šablony lze předat jako:
  - Model
  - ViewBag
  - ViewData

# HTML helpery

- Předpřipravené metody pro generování HTML kódu – hlavně formulářů.
- Používají se v šablonách.
- Dnes je pomalu nahrazují „Tag helpers“. Tag helpers umožňují vytvářet vlastní HTML tagy, které jsou na straně serveru transformovány do standardního HTML: <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-5.0>

Extension Method	Strongly Typed Method	Html Control
Html.ActionLink()	NA	<a></a>
Html.TextBox()	Html.TextBoxFor()	<input type="textbox">
Html.TextArea()	Html.TextAreaFor()	<input type="textarea">
Html.CheckBox()	Html.CheckBoxFor()	<input type="checkbox">
Html.RadioButton()	Html.RadioButtonFor()	<input type="radio">
Html.DropDownList()	Html.DropDownListFor()	<select> <option> </select>
Html.ListBox()	Html.ListBoxFor()	multi-select list box: <select>
Html.Hidden()	Html.HiddenFor()	<input type="hidden">
Html.Password()	Html.PasswordFor()	<input type="password">
Html.Display()	Html.DisplayFor()	HTML text: ""
Html.Label()	Html.LabelFor()	<label>
Html.Editor()	Html.EditorFor()	Generuje HTML formulářový prvek podle typu dat.



```
@model Booking.LoginUserForm
@{
    Layout = "_AccountLayout";
    ViewData["Title"] = "Přihlášení";
}

@using (Html.BeginForm("Login", "Account", FormMethod.Post))
{
    @Html.ValidationSummary(true);
    <div>
        @Html.LabelFor(model => model.Email)
        @Html.TextBoxFor(model => model.Email)
        @Html.ValidationMessageFor(model => model.Email)
    </div>
    <div>
        @Html.LabelFor(model => model.Password)
        @Html.PasswordFor(model => model.Password)
        @Html.ValidationMessageFor(model => model.Password)
    </div>
    <div>
        @Html.LabelFor(model => model.RememberMe)
        @Html.CheckBoxFor(model => model.RememberMe)
    </div>
    <button type="submit">Přihlásit se</button>
}
}
```

```
// _AccountLayout.cshtml

<!DOCTYPE html>
<html lang="cs">
    <head>
        <meta charset="utf-8">
        <title>@ViewData["Title"]</title>
    </head>
    <body>
        @RenderBody()
    </body>
</html>
```

```
@{
    Layout = "_AnotherLayout";
    ViewData["Title"] = "Ukázka";
}

<ul>
@foreach(string name in ViewBag["NameList"]){
    <li>@name</li>
}
</ul>
@Html.ActionLink("Text odkazu", "NazevAkce", "NazevControlleru")
}
```

```
@model Booking.LoginUserForm
@{
    Layout = "_AccountLayout";
    ViewData["Title"] = "Přihlášení";
}

<form asp-controller="Account" asp-action="Login" method="post">
    <div asp-validation-summary="All"></div>
    <div>
        <label asp-for="Email"></label>
        <input asp-for="Email" />
        <span asp-validation-for="Email"></span>
    </div>
    <div>
        <label asp-for="Password"></label>
        <input asp-for="Password" />
        <span asp-validation-for="Password"></span>
    </div>
    <div>
        <input asp-for="RememberMe" />
        <label asp-for="RememberMe"></label>
    </div>
    <button type="submit">Přihlásit se</button>
</form>
```

# Model binding

- Mapuje HTTP požadavek na objekty/argumenty akcí.

```
// In controller
public IActionResult Login(LoginCustomerForm model, string returnUrl = null) {
    if(ModelState.IsValid)
    {
        // model is valid

        // custom errors can be added
        if(model.Email.IndexOf("vsb.cz") == -1){
            ModelState.AddModelError("Email", „Email has to contain \"vsb.cz\"");
        }

    }
    return View(model);
}
```

# Routing

- Mapuje URL na akce controlleru – co se má spustit, když uživatel zadá určitou URL.

```
// Startup.cs
public void Configure(IApplicationBuilder app)
{
    app.UseRouting();

    app.MapControllerRoute (
        name: "adminRoute",
        pattern: "admin/{lang}/{controller}/{action}/{id?}",
        defaults: new { lang = "cs", area = "subject", controller = "Home", action = "Index" },
        constraints: new { lang = "[a-z]{2}" }
    );

    app.MapControllerRoute (
        name: "default",
        pattern : "{controller}/{action}/{id?}",
        defaults: new { controller = "Home", action = "Index" }
    );
}
```

# Routing - HTTP verb templates

- Atributy, které umožňují „omezit“ volání akce controlleru jen na určitou HTTP metodu.
- [HttpGet]
- [HttpPost]
- [HttpPut]
- [HttpDelete]
- [HttpHead]
- [HttpPatch]

```
public class HomeController : Controller {  
  
    [HttpGet]  
    public IActionResult Index()  
    {  
        return View();  
    }  
  
    [HttpPost]  
    public IActionResult Index(FormModel model)  
    {  
        return View();  
    }  
}
```

# Základní třídy, které implementují IActionResult

- ViewResult
- JsonResult
- ContentResult
- FileStreamResult
- FileContentResult
- RedirectResult
- RedirectToActionResult
- OkResult, NotFoundResult, ForbidResult, UnauthorizedResult, ....
- Atd..

# Autorizace

- Založená na rolích.
- Založená na Policy.
- Policy může obsahovat libovolnou vlastní logiku.
- Policy může obsahovat:
  - Role
  - Tvrzení (Claims)
  - Požadavky (Requirements) – vlastní logika

```
[Authorize(Roles = "Admin")]  
public class AdministrationController : Controller  
{  
}
```

```
[Authorize(Policy = "MyPolicyName")]  
public class AdministrationController : Controller  
{  
}
```

```
// Startup.cs - uvnitř funkce „ConfigureServices“  
services.Configure<AuthorizationOptions>(options =>  
{  
    options  
        .AddPolicy("SomePolicyName", policy => policy.....));  
    options  
        .AddPolicy(  
            "MyPolicyName",  
            policy => policy.Requirements.Add(new AdminRequirement())  
        );  
});
```

# Vlastní Policy (Requirement)

```
public class AdminRequirement : IAuthorizationHandler, IAuthorizationRequirement
{
    public async Task HandleAsync(AuthorizationHandlerContext context)
    {
        var ctx = context.Resource as HttpContext;

        if(ctx.Session.GetString("role") == "admin")
        {
            context.Succeed(this);
        }
        else
        {
            context.Fail();
        }

        return Task.CompletedTask;
    }
}
```

```
Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme).AddCookie();
```



## Další informace

- Client side validation – automaticky podporována u formulářů – stačí přidat JS knihovnu „jQuery Unobtrusive Validation“
- Razor lze používat i bez MVC
- Razor syntax: <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor>
- **Open-source-** <https://github.com/aspnet>

## Documentation of ASP.NET

<https://docs.asp.net/en/latest/index.html>