ASP.NET CORE 2.0 – Part 2

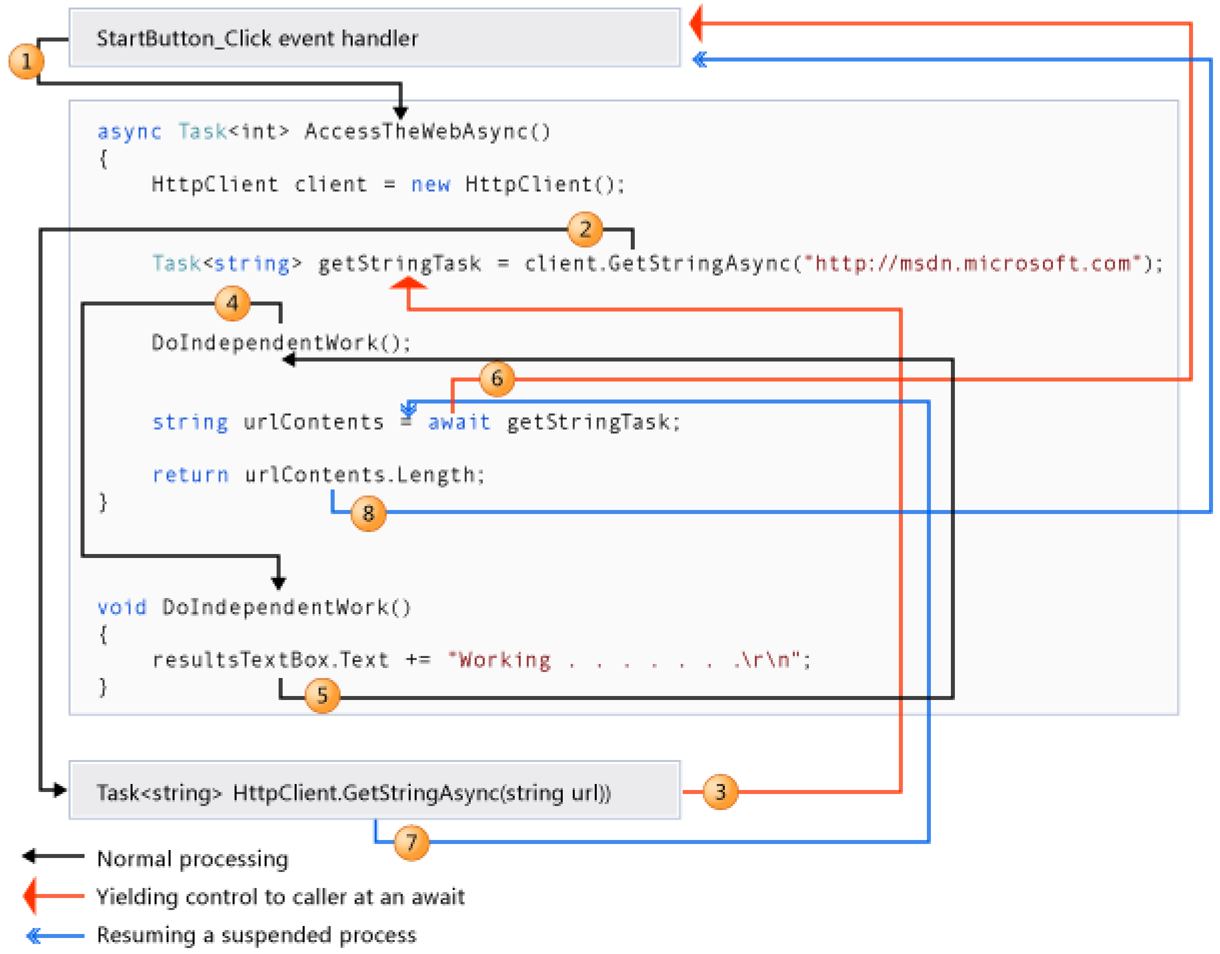
Thien Nguyen, Apr-2018

NASH TECH
The Power to Innovate

Agenda

- Async – Await
- There are NO AspNetSynchronizationContext in ASP.NET Core
- Dependency Injection
- TagHelper
- ViewComponents
- Razor pages
- Deployment options

Async - Await


StartButton_Click event handler
Task<string> HttpClient.GetStringAsync(string url))
1
2
3
4
5
6
7
8

```
async Task<int> AccessTheWebAsync()
{
    HttpClient client = new HttpClient();

    Task<string> getStringTask = client.GetStringAsync("http://msdn.microsoft.com");

    DoIndependentWork();

    string urlContents = await getStringTask;

    return urlContents.Length;
}


void DoIndependentWork()
{
    resultsTextBox.Text += "Working . . . . . . .\r\n";
}
```

← Normal processing
← Yielding control to caller at an await
← Resuming a suspended process

https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2013/hh191443(v=vs.120)

AspNetSynchronizationContext

- When an asynchronous handler resumes execution on legacy ASP.NET, the continuation is queued to the request context
  - One run at a time
  - That “re-entering” the request context involves a number of housekeeping tasks, such as setting HttpContext.Current and the current thread’s identity and culture
  - Meaning that within a request context, only one thread can actually execute code at a time
- There is no such SynchronizationContext in ASP.NET Core

https://blog.stephencleary.com/2017/03/aspnetcore-synchronization-context.html

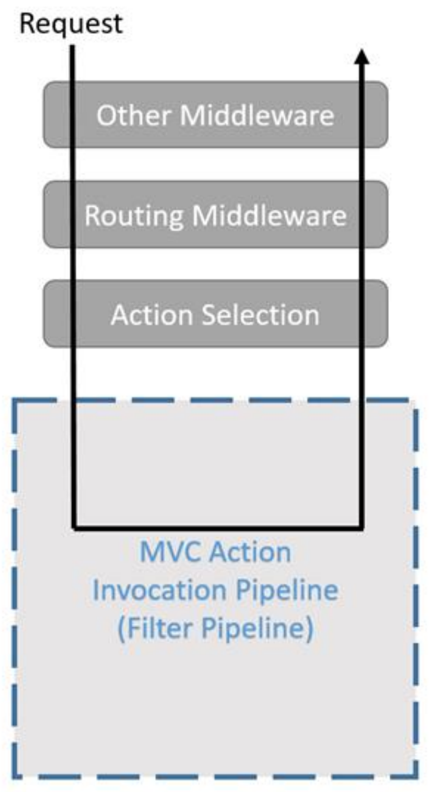
Manage Dependencies

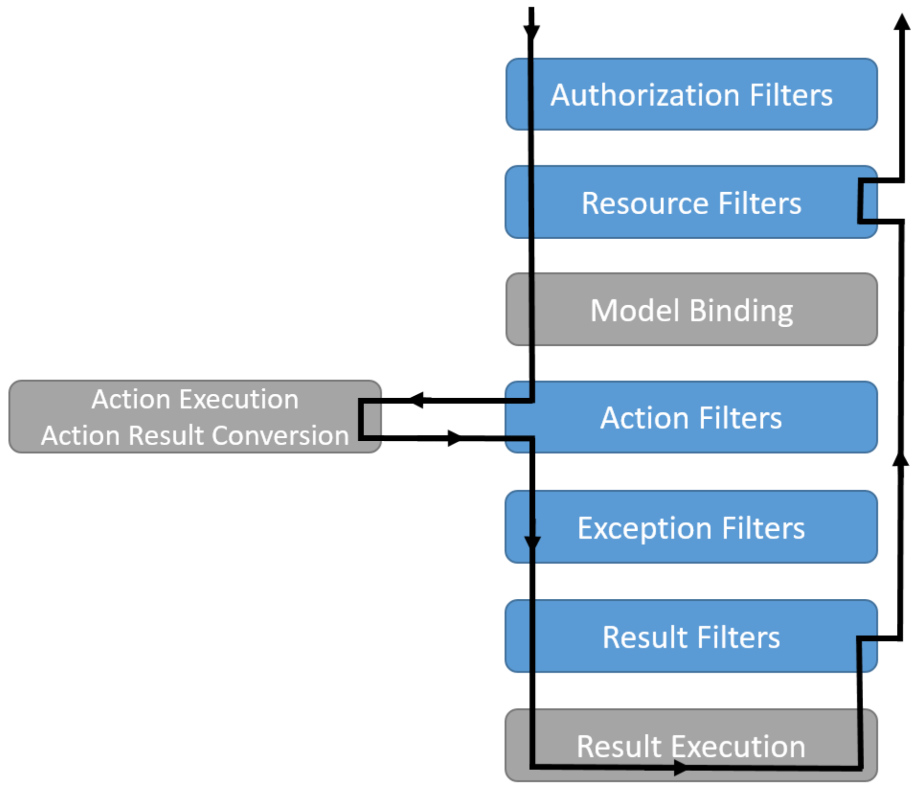
- Package reference
- Package folder: C:\Users\[YourUsername]\.nuget\packages
- .deps.json file
- Runtime package store

Dependency Injection

- ASP.NET Core is designed to support dependency injection
- Default service container
- Can integrated with other IOC container: Autofac, Unity, StrutureMap, etc.
- Can Inject to Razor

Filter





https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-2.1#dependency-injection

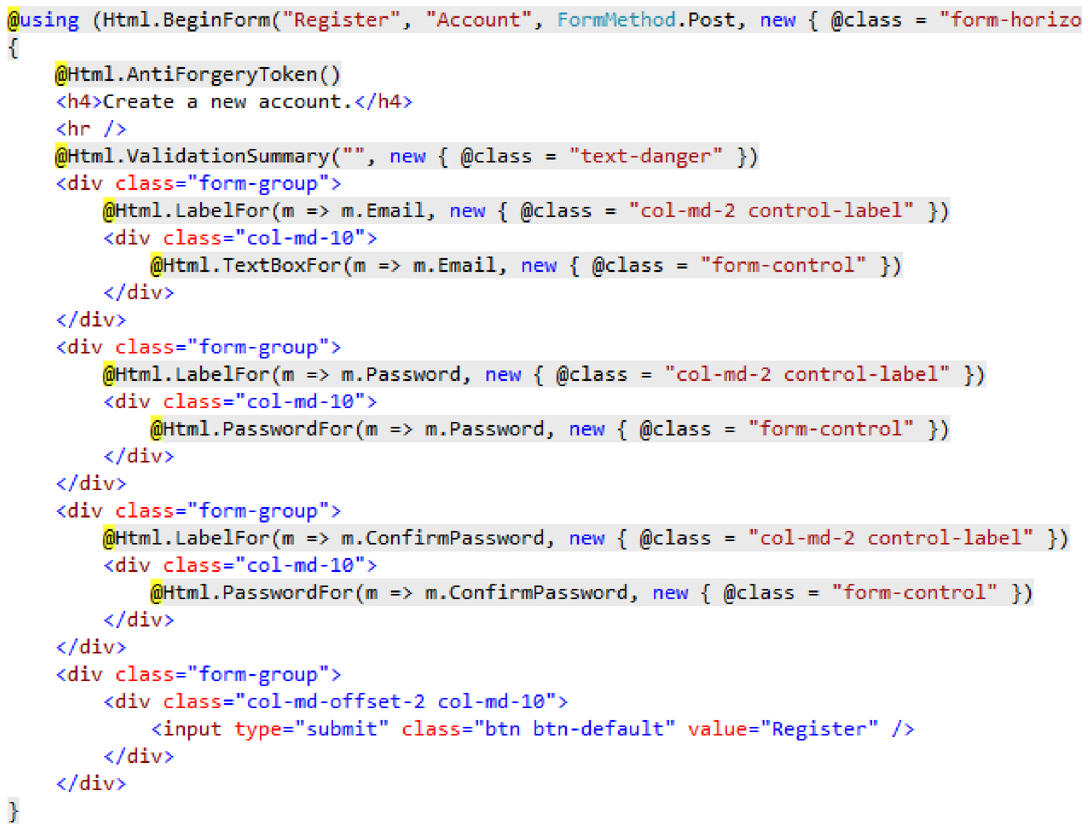
View compilation and precompilation

- Razor views are compiled at runtime when the view is first invoked

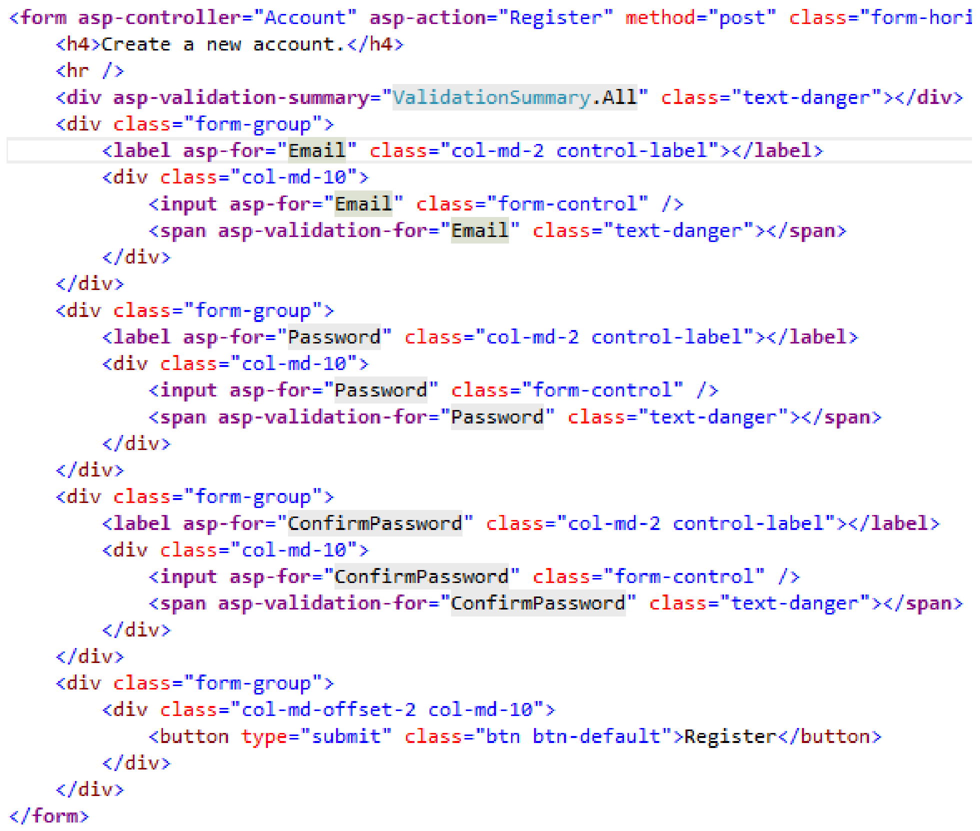
- Precompilation
  - Precompiling views results in a smaller published bundle and faster startup time.
  - You can't edit Razor files after you precompile views.

```
<MvcRazorCompileOnPublish>true</MvcRazorCompileOnPublish>
```

TagHelper



```
@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizo
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}
```



```
<form asp-controller="Account" asp-action="Register" method="post" class="form-hori
    <h4>Create a new account.</h4>
    <hr />
    <div asp-validation-summary="ValidationSummary.All" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="Email" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <input asp-for="Email" class="form-control" />
            <span asp-validation-for="Email" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Password" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <input asp-for="Password" class="form-control" />
            <span asp-validation-for="Password" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="ConfirmPassword" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <input asp-for="ConfirmPassword" class="form-control" />
            <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <button type="submit" class="btn btn-default">Register</button>
        </div>
    </div>
</form>
```

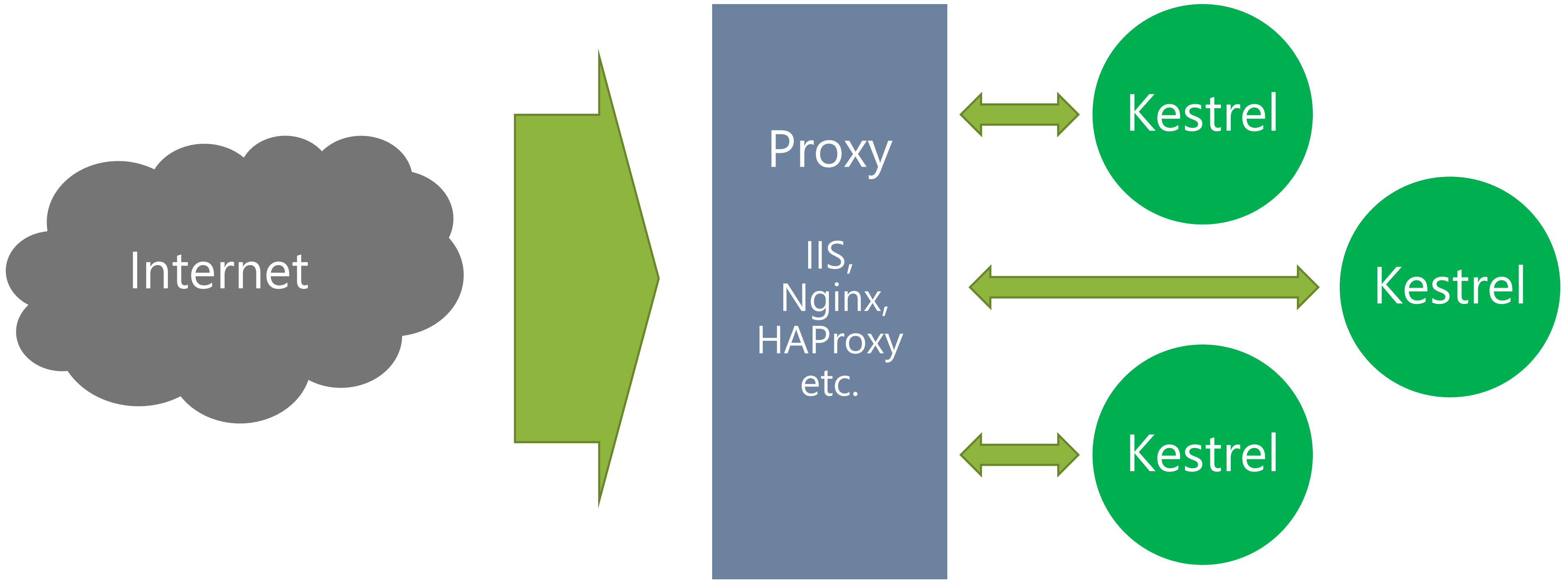
View component

- There is no child action in MVC Core
- How they are different?
  - No model binding
  - No action filters
  - Not reachable from HTTP
  - ViewBag/ViewData are shared with controller

Razor Page

- Razor Pages is a new feature of ASP.NET Core MVC that makes coding page-focused scenarios easier and more productive.

Deployment


Internet
Proxy
IIS,
Nginx,
HAProxy
etc.
Kestrel
Kestrel
Kestrel

Deployment Options

Self Contained

- App carries everything with it
  .NET Core doesn't need to be pre-installed
- Runs like a normal native executable
  Built for specific platform runtime (Windows, Linux distro, etc.)
- Bigger size

Framework Dependent

- App contains only its own code
  .Net Core need to be pre-installed in target system
- Run by dotnet cli
  Portable between installations of .NET Core
- Smaller size

Q&A

THANK YOU

www.nashtechglobal.com