Configuring and Using Feature Flags in ASP.NET Core Apps

Jason Roberts
.NET DEVELOPER

@robertsjason dontcodetired.com

Overview

- Installing and configure Microsoft Feature Management in ASP.NET Core
- Programmatically querying feature flags in controllers
- Managing controllers and controller actions with feature flags
- Disabled feature action handling
- Using a feature flag tag helper to conditionally render HTML content
- Conditionally executing MVC action filters based on feature flags
- Conditionally execute middleware based on feature flags

Conditional Middleware Execution

```
// Startup.cs

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{

    …

    app.UseMiddlewareForFeature<MiddlewareXYZ>(nameof(FeatureFlags.Printing));

    …
}
```

```
// The middleware component MiddlewareXYZ which will only appear in the request
pipeline if the feature flag FeatureFlags.Printing is enabled. This means the
middleware pipeline can be changed dynamically at runtime based on a feature flag
(and any configured feature filters).
```

Summary

services.AddFeatureManagement();

Inject IFeatureManager into controller constructor

_featureManager.IsEnabledAsync(...)

[FeatureGate("Printing")]

IDisabledFeaturesHandler

UseDisabledFeaturesHandler(new CustomDisabledFeaturesHandler());

<feature name="Printing">

options.Filters.AddForFeature(...)

app.UseMiddlewareForFeature(...)

Up Next:
Controlling Feature Flag Consistency across Multiple ASP.NET Core Requests