

# NancyFX

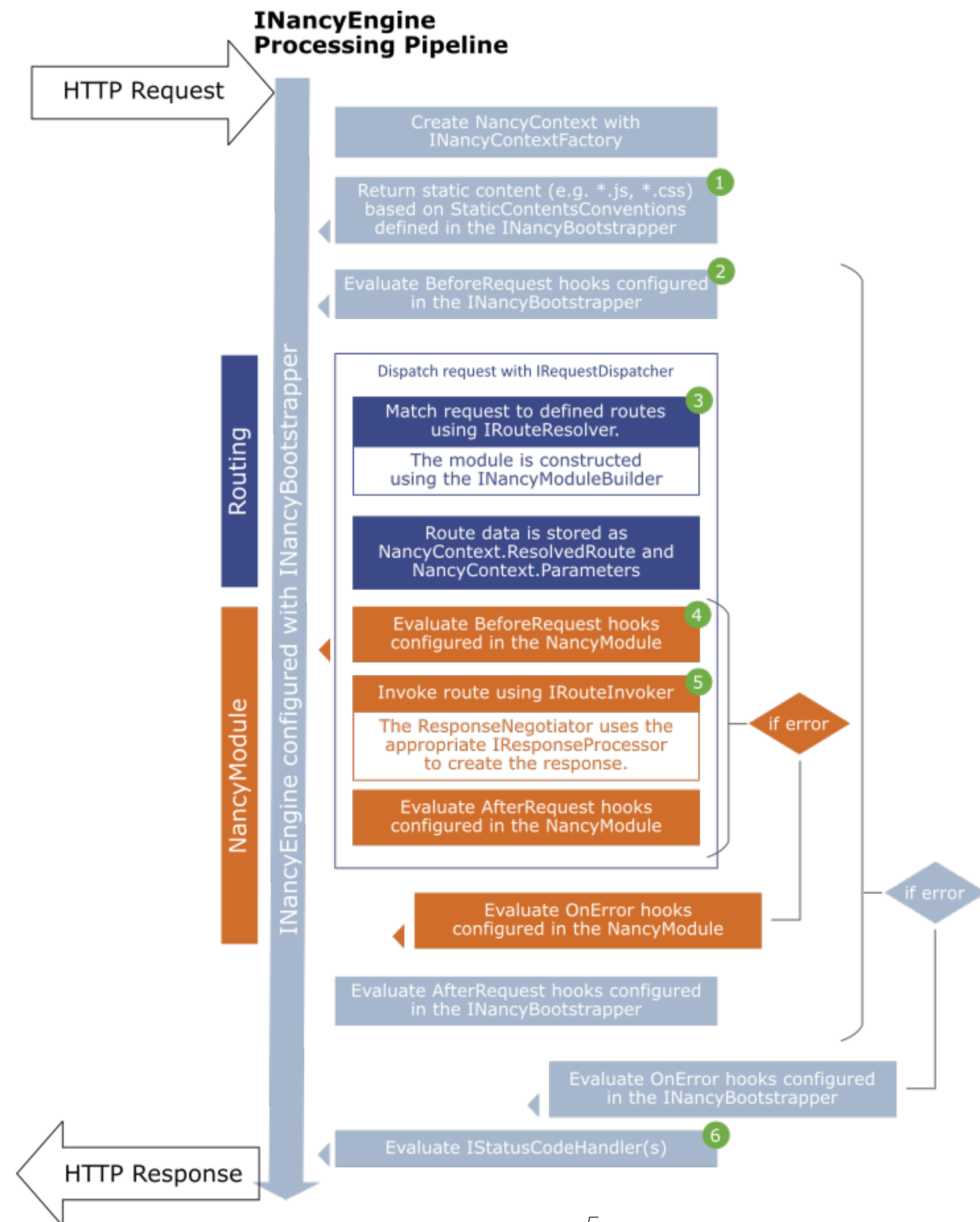
Слава Бобик

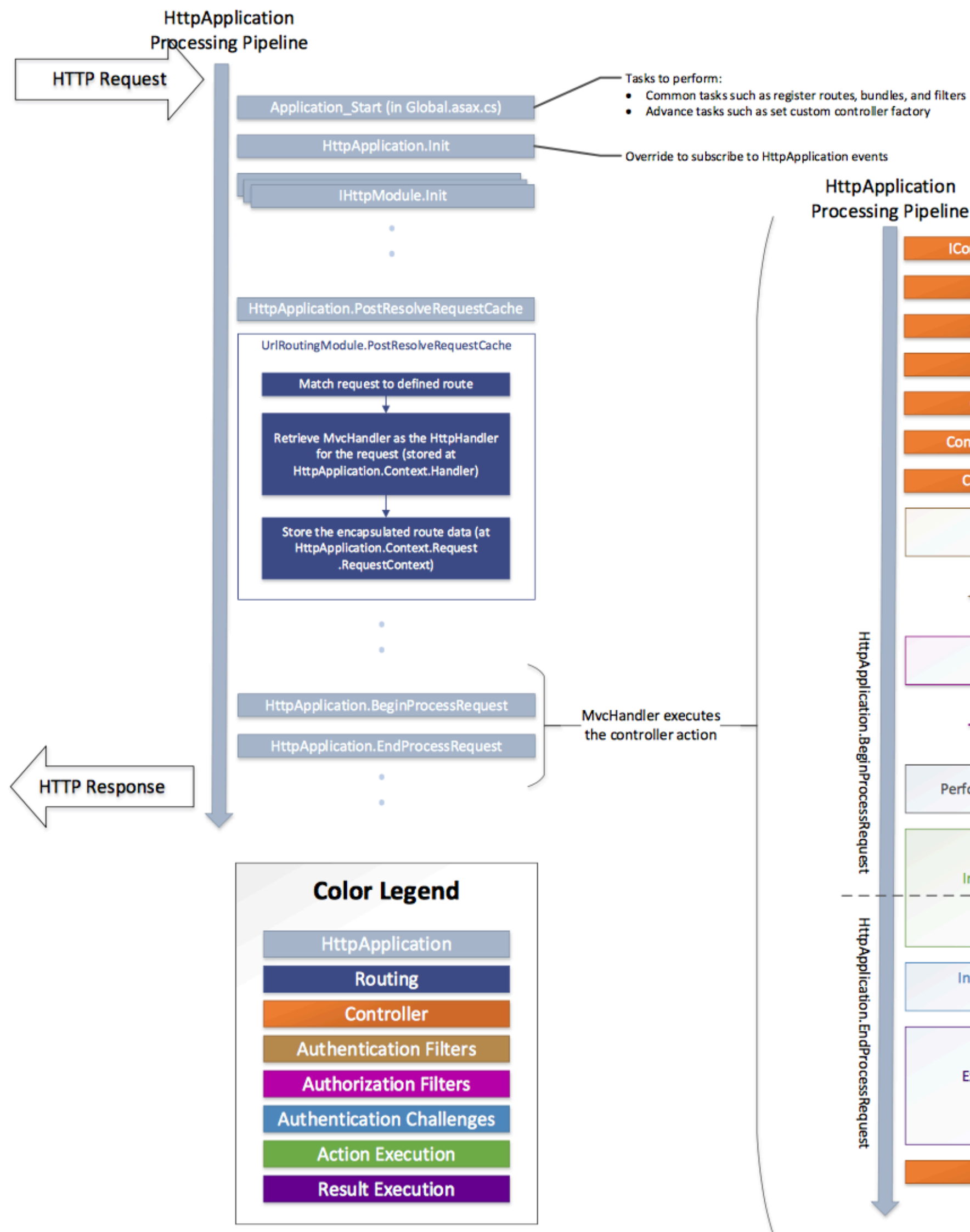
- Asp.net MVC
- FubuMVC
- OpenRasta

- Lightweight, low-ceremony
- Run everywhere
- Convention over configuration
- Testability is first class
- Easily customisable

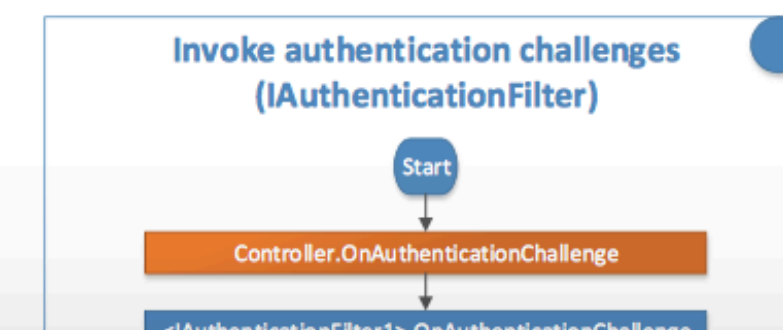
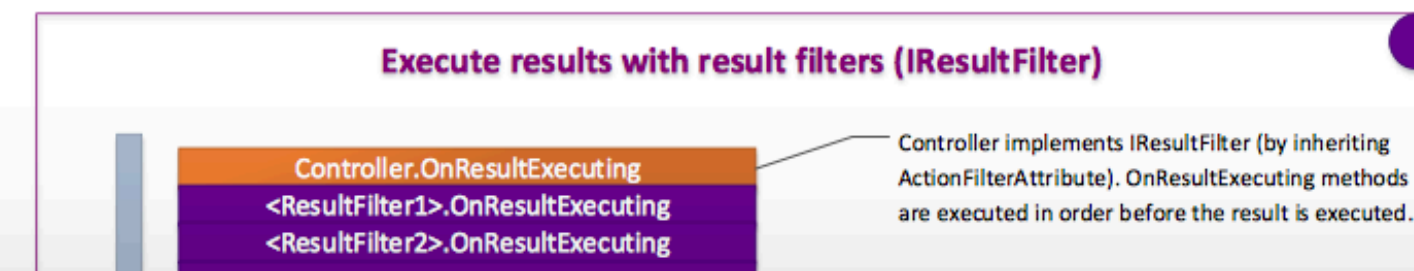
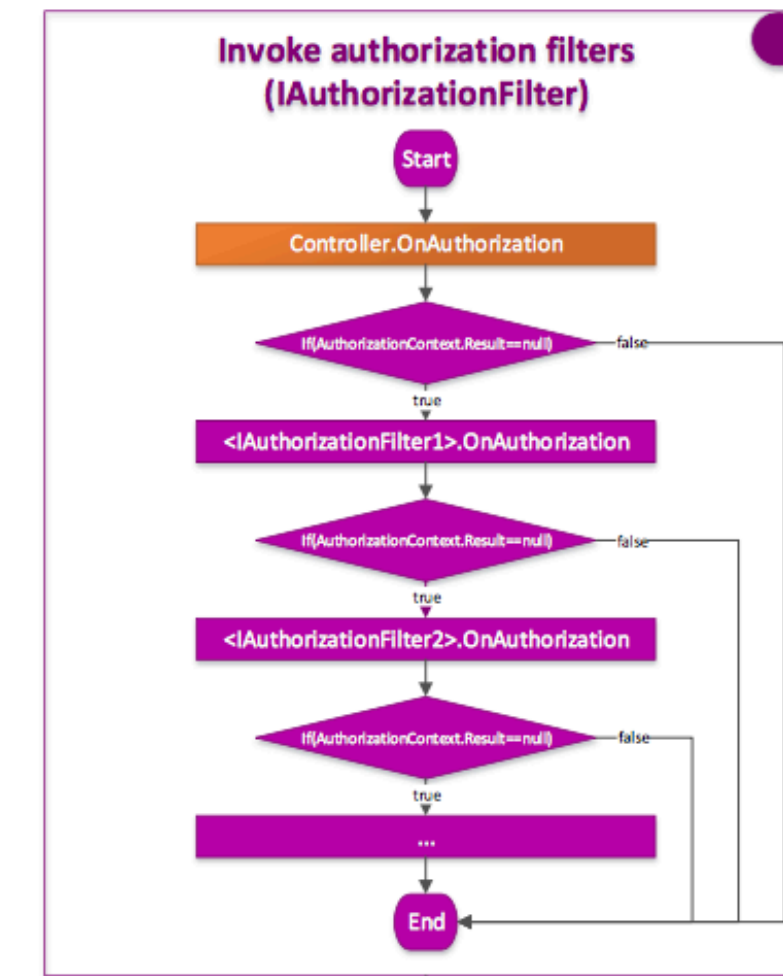
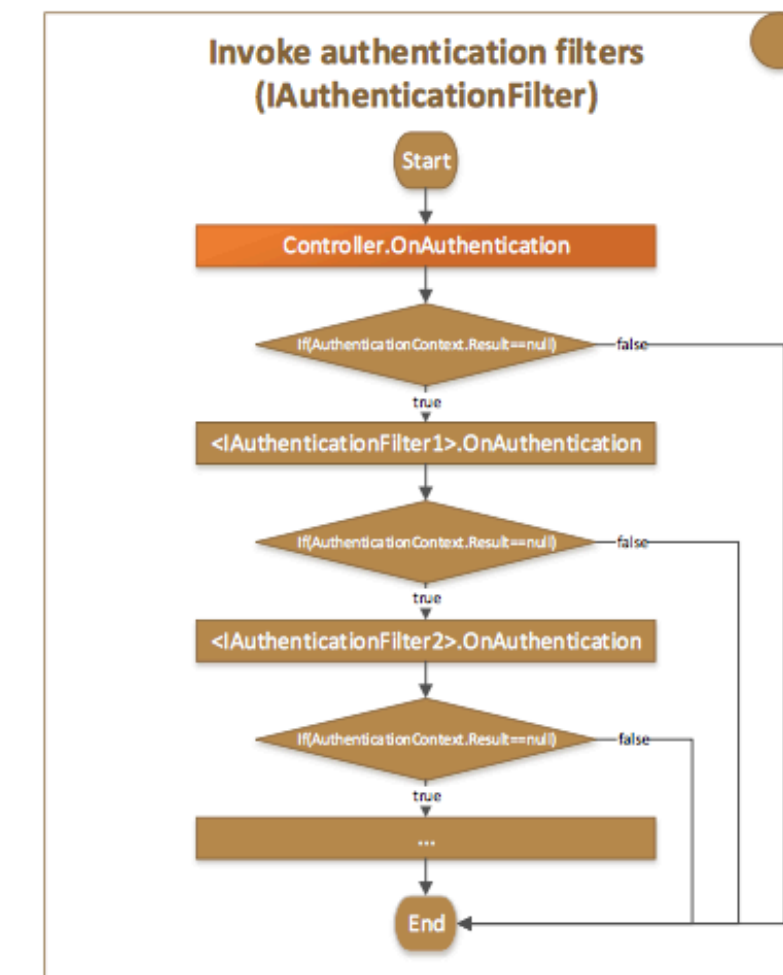
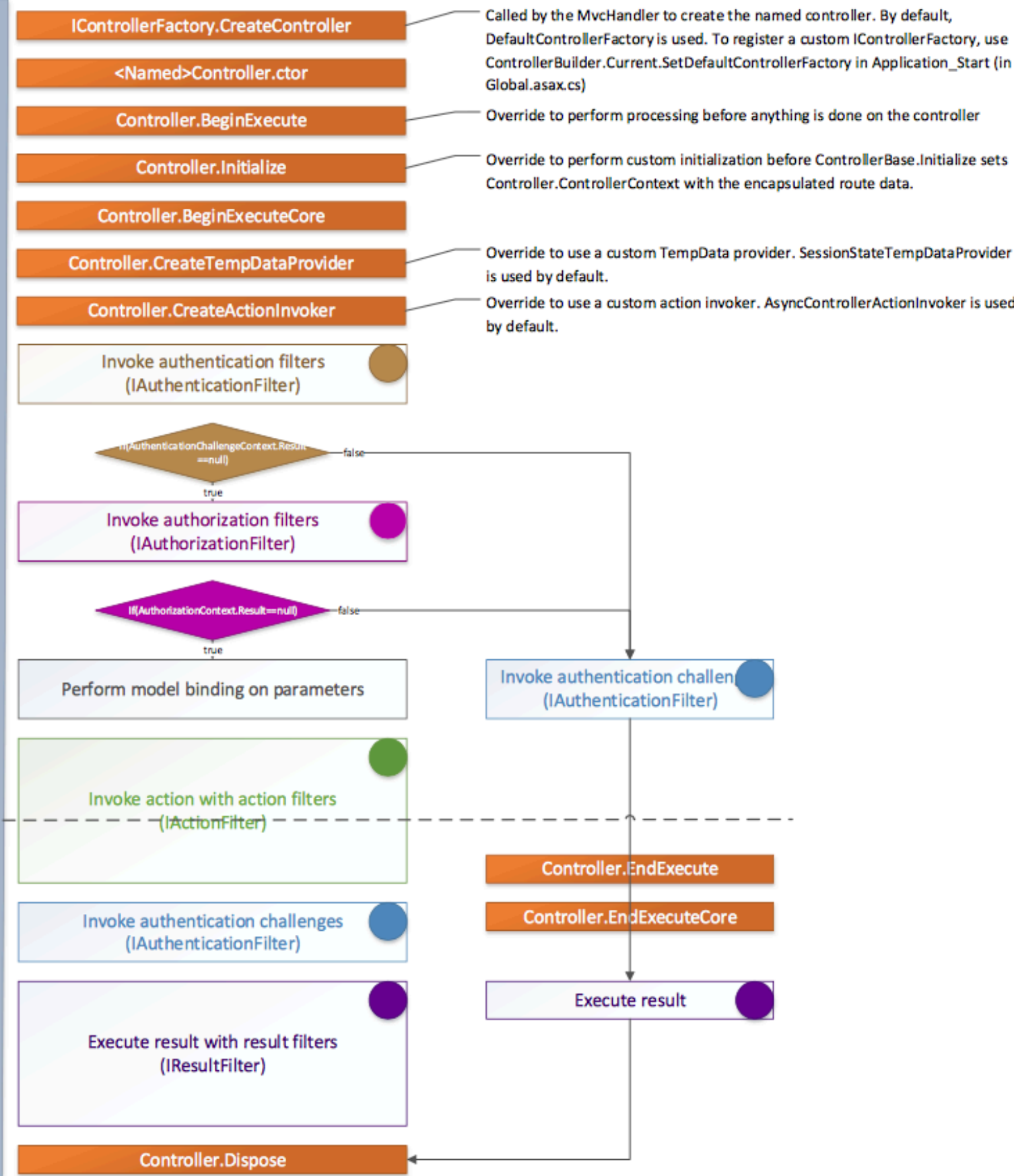
# 0.7.1

Closed on Jul 28, 2011 ⌚ Last updated over 6 years ago





The diagram shows relevant HttpApplication stages to help you understand where MVC integrates into the ASP.NET application lifecycle. In addition, for the overridable methods on the Controller object that are part the MVC lifecycle, the diagram identifies when these methods are invoked in the processing pipeline and why you might want to override them. You may or may not have the need to override any one method, but It is important for you to understand their role in the application life cycle so that you can write code at the appropriate life cycle stage for the effect you intend.







Return static content (e.g. \*.js, \*.css)  
based on StaticContentsConventions  
defined in the INancyBootstrapper

1



Evaluate BeforeRequest hooks configured  
in the INancyBootstrapper

2

```
public class Bootstrapper : DefaultNancyBootstrapper
{
    protected override void ConfigureConventions(...) {
        nancyConventions.StaticContentsConventions.Add((c, rp) => {
            return "/Static";
        });
    }
}
```



```
public class Bootstrapper : DefaultNancyBootstrapper
{
    protected override void ConfigureApplicationContainer(TinyIoCContainer container) {

    }
}

public class Bootstrapper : AutofacNancyBootstrapper {
    protected override void ConfigureApplicationContainer(ILifetimeScope existingContainer) {

    }
}
```

```
public class Bootstrapper : DefaultNancyBootstrapper
{
    protected override void ApplicationStartup(...) {

        pipelines.BeforeRequest += ctx => {};
        pipelines.AfterRequest += ctx => { };
        pipelines.OnError += (ctx, exception) => { };
    }
}
```

```
pipelines.BeforeRequest += ctx => {  
    var r = ctx.Request;  
    Console.WriteLine($"{r.Method} {r.Path}");  
  
    return null;  
};
```

```
pipelines.BeforeRequest += ctx => {  
    var response = new Response{  
        Contents = s => {  
            var data = Encoding.UTF8.GetBytes("Lorem ipsum dolor sit amet");  
            s.Write(data, 0, data.Length);  
        },  
        StatusCode = HttpStatusCode.BadRequest  
    };  
  
    return response;  
};
```

Dispatch request with IRequestDispatcher

Match request to defined routes  
using IRouteResolver.

3

The module is constructed  
using the INancyModuleBuilder

Route data is stored as  
NancyContext.ResolvedRoute and  
NancyContext.Parameters

```
public class HomeModule : NancyModule {  
    private readonly IDependency _dependency;  
  
    public HomeModule(IDependency dependency) {  
        _dependency = dependency;  
    }  
}
```





Evaluate BeforeRequest hooks  
configured in the NancyModule

4

```
public HomeModule() : base("/api") {  
    this.Before += ctx => null;  
    this.After += ctx => { };  
    this.OnError += (ctx, ex) => { };  
}
```

ctx.Items|



Items

IDictionary<string,object>

^↓ and ^↑ will move caret down and up in the editor

```
this.Before += ctx => {  
    var time = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();  
    ctx.Items.Add("start_time", time);  
  
    return null;  
};  
  
this.After += ctx => {  
    var start = (long)ctx.Items["start_time"];  
    var end = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();  
    var diff = end - start;  
};
```

Invoke route using IRouteInvoker

5

The ResponseNegotiator uses the appropriate IResponseProcessor to create the response.



```
Get("/{id}", args => args.id);
```

Get["/", **true**] = **async** (x, ct) =>

```
Get("/{id}", args => args.id);
```

# DynamicDictionary

```
Get("/" , _ => {  
    var param = this.Request.Query.query_param;  
    if (param.HasValue) {}  
});
```

```
Get("/{id:int}",  
    args => args.id + "ok",  
    ctx => ctx.  
        Request.  
        Headers.  
        UserAgent.  
        Equals("mobile"));
```



```
[Produces("application/json")]
[Route("api")]
[Wtf]
public class MyController : Controller
{
    [HttpPost("Create")]
    public async Task<HttpResponseMessage> Create([FromBody] Person data)
    {
    }
}
```

```
public HomeModule() : base("/api") {  
    Post("/create", _ => {  
        var model = this.Bind<Person>(bl => bl.X);  
        return model;  
    });  
}
```

```
public void ConfigureServices(IServiceCollection...)
{
    services.AddMvc()
        .AddXmlSerializerFormatters();
}
```

```
public class Movie
{
    public int ID { get; set; }

    [StringLength(60, MinimumLength = 3)]
    [Required]
    public string Title { get; set; }

    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }
}
```

```
public class PersonValidator : AbstractValidator<Person> {  
    public PersonValidator() {  
        this.RuleFor(x => x.Name).NotEmpty().Length(1, 2);  
    }  
}
```

```
Post("/person", _ => {  
    var model = this.BindAndValidate<Person>();  
    if (!this.ModelValidationResult.IsValid)  
    {  
        return Response.AsJson(this.ModelValidationResult.Errors);  
    }  
});
```



# Content Negotiation

- JsonProcessor
- ViewProcessor
- XmlProcessor

```
Get("/person", args => new Person());
```

```
return Response.AsJson(model)
    .WithStatusCode(HttpStatusCode.Created)
    .WithHeader("X-Custom", "value");
```

Evaluate AfterRequest hooks  
configured in the NancyModule

Evaluate OnError hooks  
configured in the NancyModule

```
this.OnError += (ctx, ex) => {};
```

Evaluate AfterRequest hooks configured  
in the INancyBootstrapper



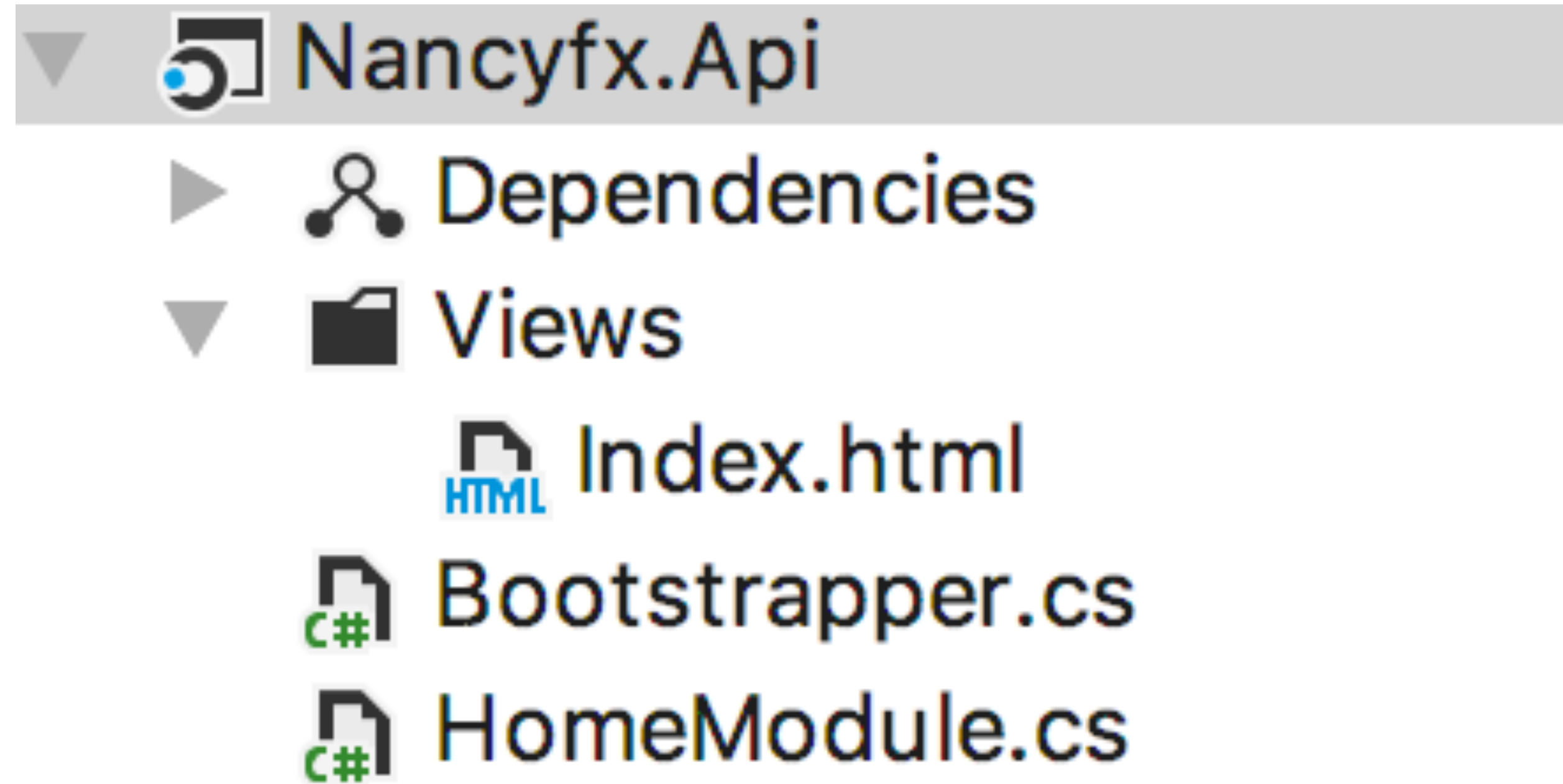
Evaluate OnError hooks configured  
in the INancyBootstrapper

Evaluate IStatusCodeHandler(s)

6

```
public class Handle418 : IStatusCodeHandler {  
    public bool HandlesStatusCode(HttpStatusCode statusCode, NancyContext context) {  
        return statusCode == HttpStatusCode.ImATeapot;  
    }  
  
    public void Handle(HttpStatusCode statusCode, NancyContext context) {  
        context.Response = "Coffeepot";  
    }  
}
```

# Super Simple View Engine



```
Get("/", _ => View["Index"]);
```

```
@Master[ 'MasterPage' ]
@Section[ 'Content' ]

@ViewBag.Test
<div id="login">
    @Partial[ 'login' ];
</div>

<div>
    @Partial[ 'user', Model.Users ];

    @!Model.BoolProperty

    @Context.Request.Path

    @Each
        <!--loop code-->
    @EndEach
</div>
@EndSection
```

---

🐼 Nancy.Viewengines.Razor	2.0.0-clinteastwood
🐼 Nancy.Viewengines.Spark	2.0.0-clinteastwood
🐼 Nancy.Viewengines.DotLiquid	2.0.0-clinteastwood
🐼 Nancy.Viewengines.Nustache	2.0.0-clinteastwood
🐼 Nancy.Viewengines.Markdown	2.0.0-clinteastwood

- **Nancy.Authentication.Forms**
- **Nancy.Authentication.Stateless**
- **Nancy.Authentication.Basic**



```
var config = new StatelessAuthenticationConfiguration(ctx => {  
    var token = ctx.Request.Headers.Authorization;  
    var payload = Jose.JWT.Decode<JwtToken>(token, @"~\_(\ツ)\_/~");  
    //JwtToken{ string id, long exp }  
    var expires = DateTime.FromBinary(payload.exp);  
    if (expires < DateTime.UtcNow) return null;  
  
    return new ClaimsPrincipal(new HttpListenerBasicIdentity(payload.id, null));  
});
```

```
protected override void ApplicationStartup(TinyIoCContainer container, IPipelines pipelines)
{
    StatelessAuthentication.Enable(pipelines, config);
}
```

```
public HomeModule() {  
    this.RequiresAuthentication();  
  
    Get("/", _ => {  
        var id = this.Context.CurrentUser.Identity.Name;  
        return View("Index", id);  
    });  
}
```

# Middleware

```
[Fact]
public void TestMainRoute() {
    var bootstrapper = new DefaultNancyBootstrapper();
    var browser = new Browser(bootstrapper);

    var result = browser.Get("/test", with => {
        with.HttpRequest();
    });

    Assert.Equal(HttpStatusCode.ImATeapot, result.Result.StatusCode);
}
```

```
var result = browser.Post("/test", with => {  
    with.HttpRequest();  
    with.Accept("application/json");  
    with.Header("X", "Y");  
    with.JsonBody(new { some_key = "value" });  
});
```

- Kestrel
- Self
- ASP.NET
- ...

```

type App() as this =
    inherit NancyModule()
    do
        this.Get.["/" ] <- fun _ -> "Hello World!" :> obj
        this.Get.["/Fsharp"] <- fun _ -> "I can into F#" :> obj
        this.Get.["/Json"] <- fun _ -> this.Response.AsJson( [ "Test" ] ) :> obj

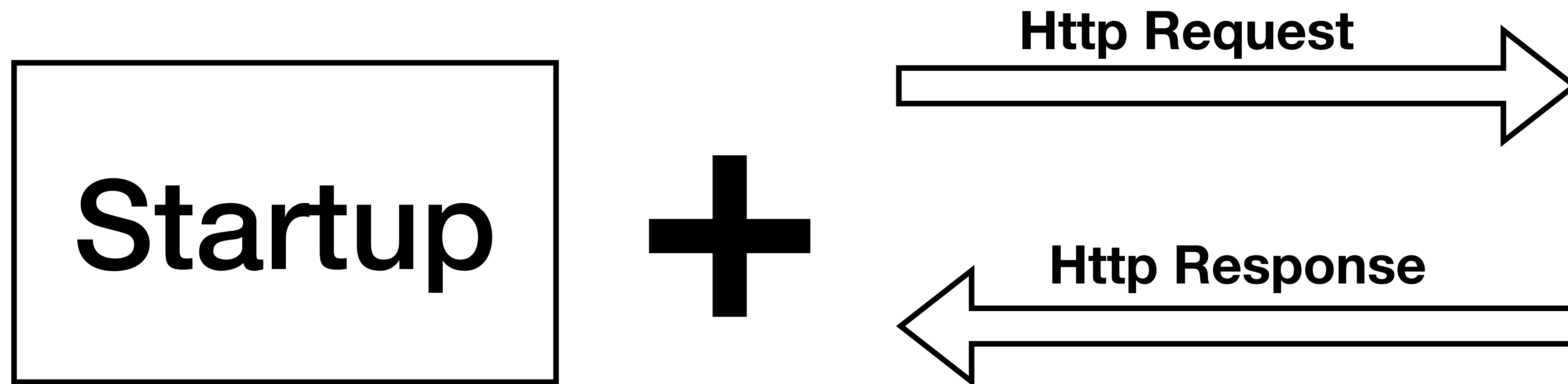
[<EntryPoint>]
let main args =

    let nancy = new Nancy.Hosting.Self.NancyHost(new Uri("http://localhost:" + "8100"))
    nancy.Start()
    while true do Console.ReadLine() |> ignore
    0

```



# Performance



- Минимализм
- Низкий порог входа
- Хорошо тестируемый
- Работает поверх .net core
- Все еще в стадии alpha :(

## 2.0-clinteastwood

**Closed** on Dec 22, 2016 ⌚ Last updated about 1 year ago

 [aspnet](#) / [Mvc](#)

 Watch ▾

829

★ Unstar

4,939

 Fork

1,944

↔ Code

⚠ Issues **279**


🔗 Pull requests **15**

📁 Projects **2**

📖 Wiki

📊 Insights


 [NancyFx](#) / [Nancy](#)

 Unwatch ▾

450

★ Unstar

5,854

 Fork

1,365

↔ Code

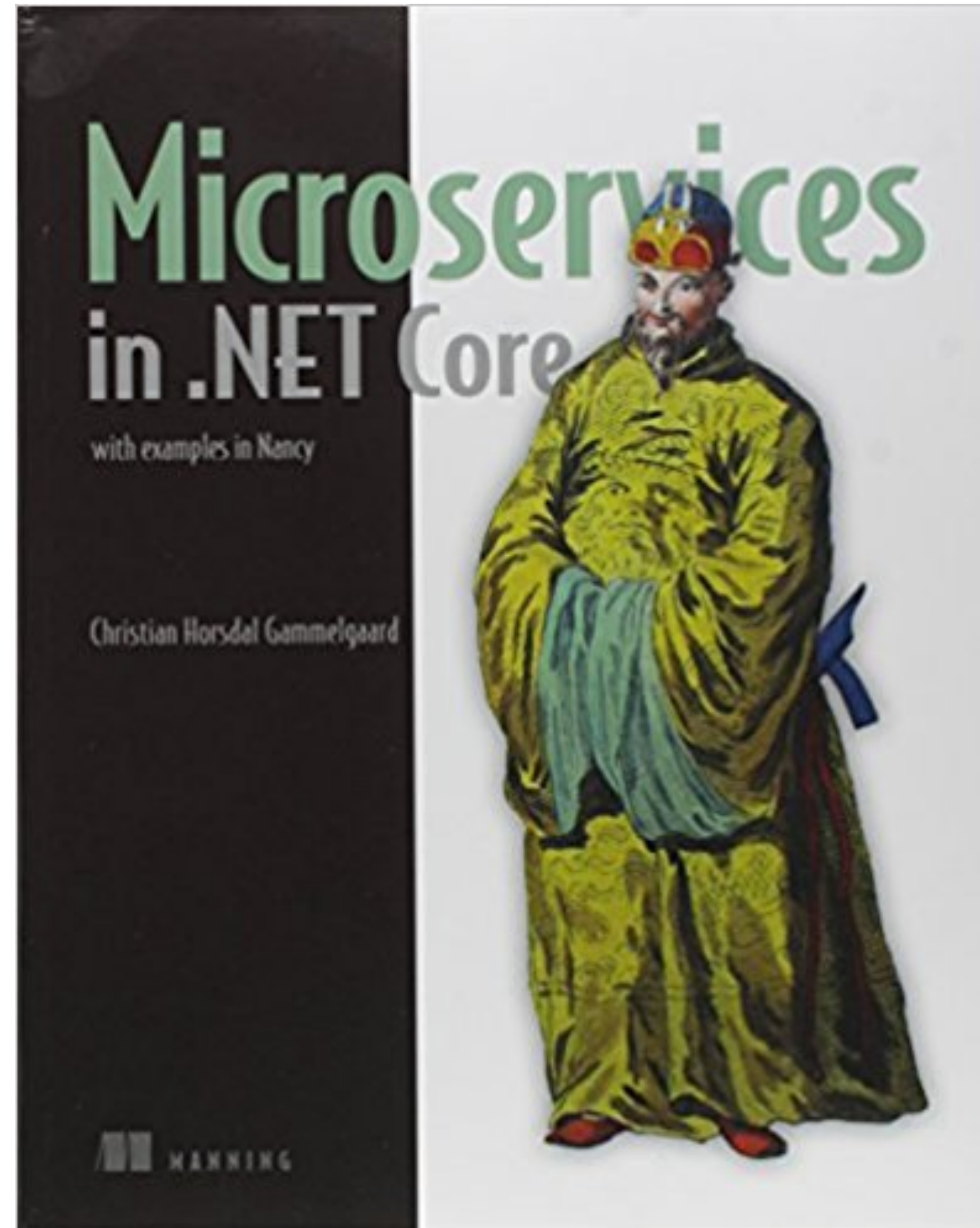
⚠ Issues **125**

🔗 Pull requests **15**

📁 Projects **0**

📖 Wiki

📊 Insights



- <https://github.com/nancyfx>
- <http://blog.nancyfx.org>
- <http://kristian.hellang.com>

- <https://twitter.com/slavabobik>
- [slava.b@okmeter.io](mailto:slava.b@okmeter.io)