

OWIN and Katana

Dominick Baier & Brock Allen
dominick.baier@thinktecture.com
brockallen@gmail.com

The logo for thinktecture, featuring the word "thinktecture" in a lowercase, sans-serif font. The "think" portion is in a light blue color, and the "tecture" portion is in a light gray color. The logo is set against a dark gray rectangular background.

thinktecture

Dominick Baier

- Security consultant at thinktecture
- Focus on
 - security in distributed applications
 - identity management
 - access control
 - Windows/.NET security
 - mobile app security

- Microsoft MVP for Developer Security
- ASP.NET Web API Advisor
- dominick.baier@thinktecture.com
- <http://leastprivilege.com>



Brock Allen

- Focus on:
 - Web development
 - Web security
- Work:
 - Security consultant at thinktecture
 - Author and instructor at DevelopMentor
- Community:
 - Open source contributor to thinktecture
 - Open source contributor to ASP.NET
 - Microsoft MVP for ASP.NET

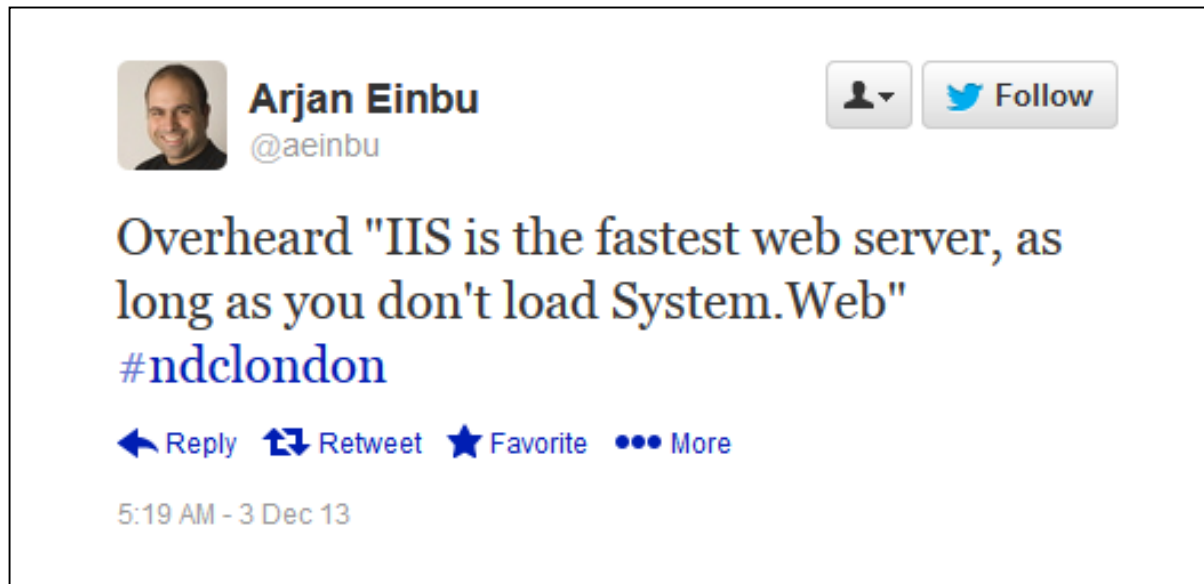


Outline

- OWIN / Katana
 - Motivation
 - Architecture
 - Specification
 - Features
- Applications
 - Web API
 - SignalR
 - Nancy

OWIN Motivation

- System.Web.dll (aka ASP.NET)
 - 12+ year old web framework
 - Unnamed Microsoft employee on System.Web:
 - “We fix one bug and open seven new ones”
 - Always executes lots of ASP.NET-specific code



OWIN Motivation

- Empty MVC project
 - Lots of baggage

```
Web.config*  X
1 <?xml version="1.0" encoding="utf-8"?>
2 <configuration>
3   <configSections>
4     <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
5     <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Versio
6   </configSections>
7   <connectionStrings>
8     <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MvcAppl
9     providerName="System.Data.SqlClient" />
10  </connectionStrings>
11  <appSettings>
12    <add key="webpages:Version" value="3.0.0.0" />
13    <add key="webpages:Enabled" value="false" />
14    <add key="ClientValidationEnabled" value="true" />
15    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
16  </appSettings>
17  <system.web>
18    <authentication mode="None" />
19    <compilation debug="true" targetFramework="4.5" />
20    <httpRuntime targetFramework="4.5" />
21  </system.web>
22  <system.webServer>
23    <modules>
24      <remove name="FormsAuthenticationModule" />
25    </modules>
26  </system.webServer>
27  <runtime>
28    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
29      <dependentAssembly>
30        <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
31        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
32      </dependentAssembly>
33      <dependentAssembly>
34        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
35        <bindingRedirect oldVersion="1.0.0.0-5.0.0.0" newVersion="5.0.0.0" />
36      </dependentAssembly>
37      <dependentAssembly>
38        <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
39        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
40      </dependentAssembly>
41      <dependentAssembly>
42        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
43        <bindingRedirect oldVersion="1.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
44      </dependentAssembly>
45    </assemblyBinding>
46  </runtime>
47  <entityFramework>
48    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
49      <parameters>
50        <parameter value="v11.0" />
51      </parameters>
52    </defaultConnectionFactory>
53    <providers>
54      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
55  </providers>
56 </entityFramework>
```

Solution 'MvcApplication' (1 project)

- MvcApplication
 - Properties
 - References
 - Microsoft.CSharp
 - Microsoft.Web.Infrastructure
 - Microsoft.Web.Mvc.FixedDisplayModes
 - Newtonsoft.Json
 - System
 - System.ComponentModel.DataAnnotations
 - System.Configuration
 - System.Core
 - System.Data
 - System.Data.DataSetExtensions
 - System.Drawing
 - System.EnterpriseServices
 - System.Net.Http
 - System.Net.Http.Formatting
 - System.Net.Http.WebRequest
 - System.Web
 - System.Web.Abstractions
 - System.Web.ApplicationServices
 - System.Web.DynamicData
 - System.Web.Entity
 - System.Web.Extensions
 - System.Web.Helpers
 - System.Web.Http
 - System.Web.Http.WebHost
 - System.Web.Mvc
 - System.Web.Razor
 - System.Web.Routing
 - System.Web.Services
 - System.Web.WebPages
 - System.Web.WebPages.Deployment
 - System.Web.WebPages.Razor
 - System.Xml
 - System.Xml.Linq

OWIN Motivation

- Node.js envy
 - Start small and add as needed

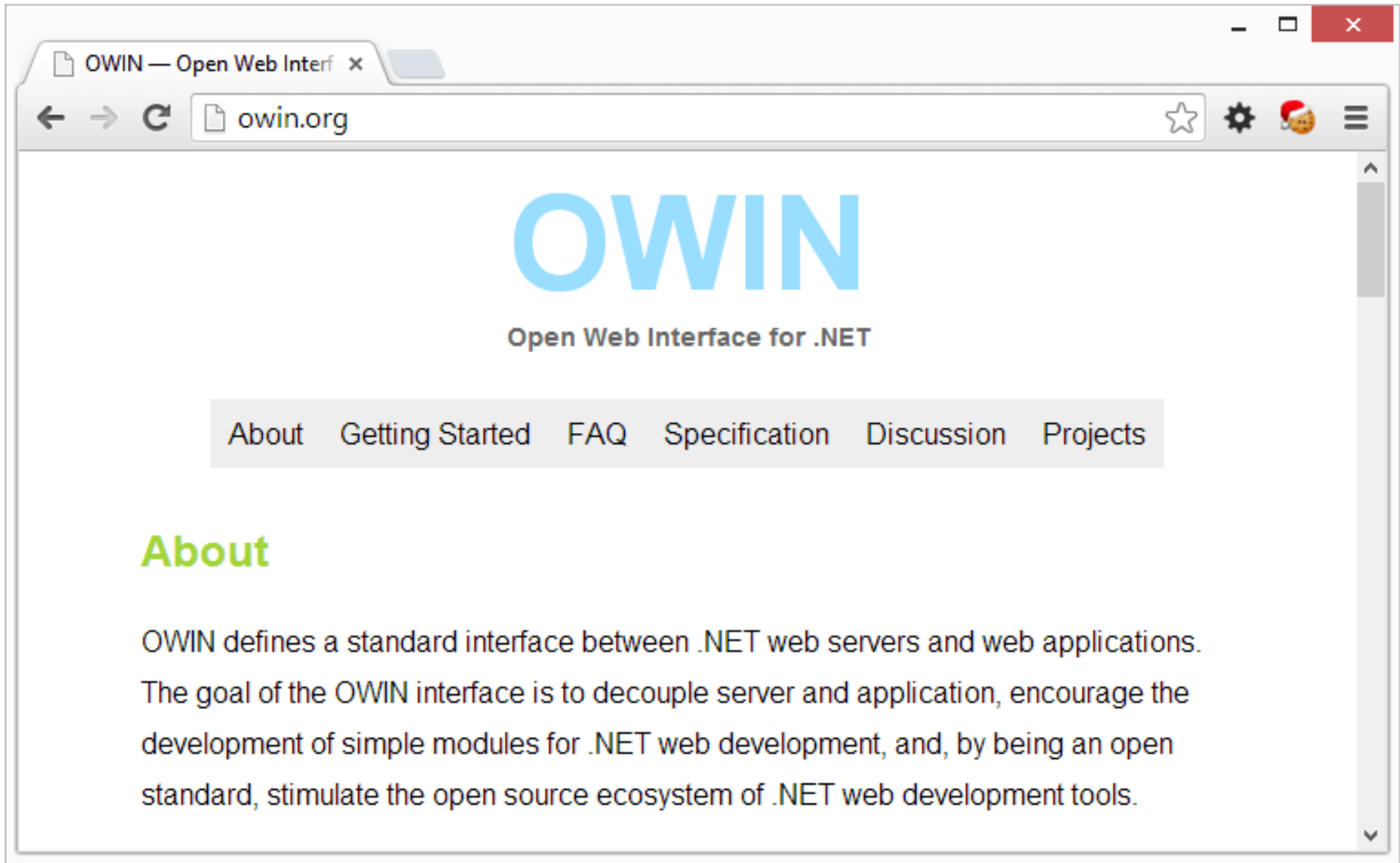
An example of a [web server](#) written with Node which responds with 'Hello World':

```
var http = require('http');

http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(8124);

console.log('Server running at http://127.0.0.1:8124/');
```

What is OWIN

A screenshot of a web browser window displaying the OWIN website. The browser's address bar shows "owin.org". The page features the OWIN logo in large blue letters, with the subtitle "Open Web Interface for .NET" below it. A navigation menu is visible, containing links for "About", "Getting Started", "FAQ", "Specification", "Discussion", and "Projects". The "About" section is highlighted in green. The text below the "About" heading explains that OWIN defines a standard interface between .NET web servers and web applications, aiming to decouple server and application, encourage the development of simple modules, and stimulate the open source ecosystem of .NET web development tools.

OWIN — Open Web Interf x

owin.org

OWIN

Open Web Interface for .NET

[About](#) [Getting Started](#) [FAQ](#) [Specification](#) [Discussion](#) [Projects](#)

About

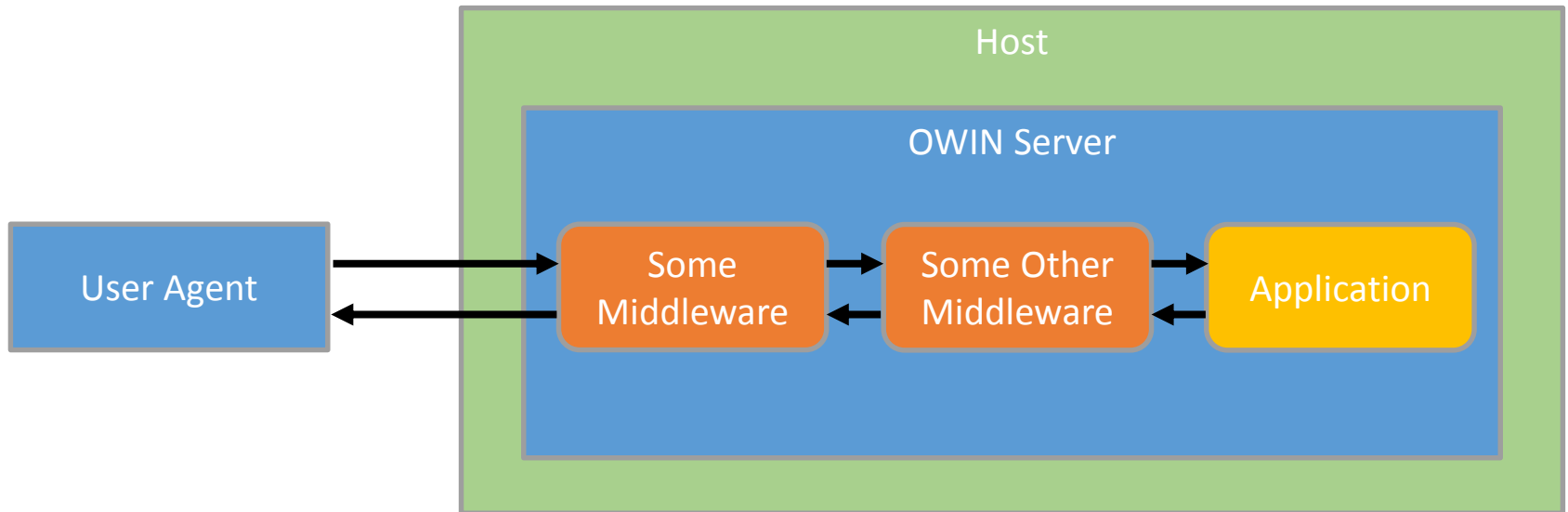
OWIN defines a standard interface between .NET web servers and web applications. The goal of the OWIN interface is to decouple server and application, encourage the development of simple modules for .NET web development, and, by being an open standard, stimulate the open source ecosystem of .NET web development tools.

What is Katana

- Microsoft's OWIN implementation
 - <https://katanaproject.codeplex.com/>
- Set of hosts and servers
 - IIS or self-hosting
- Set of convenience classes
 - *OwinContext*, *OwinRequest*, *OwinResponse*, etc.
 - *ApplicationBuilderUseExtensions*
 - *AuthenticationManager*
- Set of middleware for common features
 - Authentication
 - Hosting content (e.g. static files)
 - CORS

OWIN architecture

- **Host** manages process lifetime
- **Server** hosts HTTP and implements OWIN API
- **Middleware** are linked components that process requests
- **Application** code targeting a framework (e.g. Web API)



OWIN specification

- **Environment** models HTTP request/response
 - *IDictionary<string, object>*
 - All .NET primitives so no framework dependencies
 - Standard set of key/value pairs

Key	Type
owin.RequestScheme	string
owin.RequestMethod	string
owin.RequestPath	string
owin.RequestBody	Stream
owin.RequestHeaders	<i>IDictionary<string, string[]></i>
owin.ResponseStatusCode	int
owin.ResponseHeaders	<i>IDictionary<string, string[]></i>
owin.ResponseBody	Stream

OWIN specification

- **Middleware** is code that can process requests
 - *Func<IDictionary<string, object>, Task>* (aka "AppFunc")
 - Method named *Invoke* by convention

```
public async Task Invoke(IDictionary<string, object> env)
{
    var responseBody = (Stream)env["owin.ResponseBody"];
    using (var sw = new StreamWriter(responseBody))
    {
        await sw.WriteLineAsync("<h1>Hello OWIN!</h1>");
    }
}
```

Chaining middleware

- Can use multiple middleware in an application
 - Constructor must accept reference to "next"

```
public class SomeMiddleware
{
    Func<IDictionary<string, object>, Task> next;

    public SomeMiddleware(
        Func<IDictionary<string, object>, Task> next)
    {
        this.next = next;
    }

    public async Task Invoke(IDictionary<string, object> env)
    {
        // do pre-processing here
        await next(env);
        // do post-processing here
    }
}
```

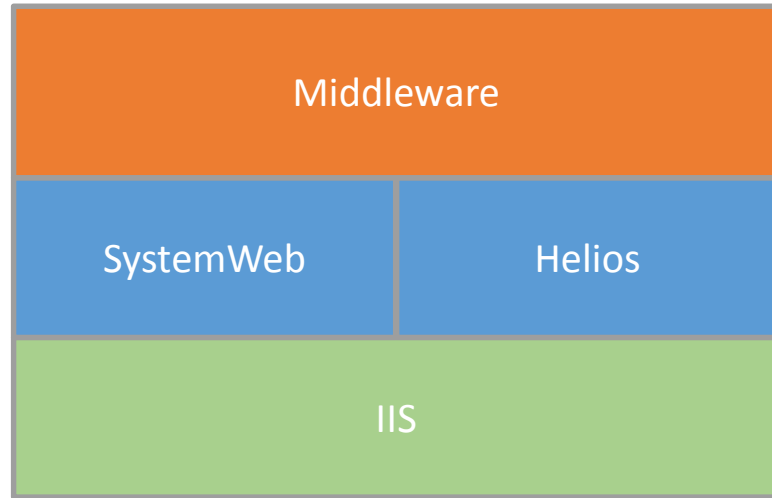
Configuring middleware

- OWIN server provides means for registering middleware
 - *IApplicationBuilder.Use(typeof(Middleware))*
 - *IApplicationBuilder.Use(instance)*
- Convention to provide *Startup* class with *Configuration* method
 - Invoked by OWIN server

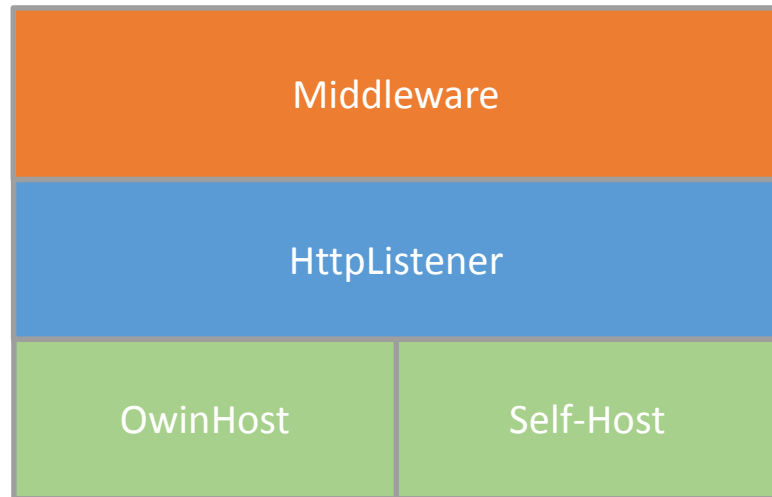
```
public class Startup
{
    public void Configuration(IApplicationBuilder app)
    {
        app.Use(typeof(SomeMiddleware));
        app.Use(typeof(SomeOtherMiddleware));
    }
}
```

Katana hosts and servers

- IIS Host
- IIS-based servers
 - SystemWeb
 - Helios



- Non-IIS Host
 - OwinHost
 - Self-Host
- HttpListener server



OwinContext

- Provides much easier API for working with environment
 - Strongly typed wrappers for environment dictionary

```
public Task Invoke(IDictionary<string, object> env)
{
    var ctx = new OwinContext(env);

    var url = ctx.Request.Uri;
    var accept = ctx.Request.Headers["Accept"];

    var username = ctx.Request.User.Identity.Name;

    ctx.Response.StatusCode = 200;
    ctx.Response.Headers.Add("Foo", new[]{"Bar"});

    using (var sw = new StreamWriter(ctx.Response.Body))
    {
        return sw.WriteLineAsync("<h1>Hello Katana!</h1>");
    }
}
```


ApplicationBuilderExtensions

- *Use<T>()*
 - Generic version of *Use(typeof(T))*
- *Map(requestPath, IApplicationBuilder)*
 - Allows branching in middleware based upon URL
- *Run(AppFunc)*
 - Delegate invoked that terminates OWIN pipeline

```
public void Configuration(IApplicationBuilder app)
{
    app.Use<SomeMiddleware>();

    app.Map("/foo", fooApp => {
        fooApp.Use<SomeMiddleware2>();
    });

    app.Run(async ctx => {
        await ctx.Response.WriteAsync("<h1>End of pipeline!</h1>");
    });
}
```

Katana static file middleware

- When self-hosting its sometimes useful to serve files
 - File system
 - Embedded resource

```
app.UseStaticFiles("/files");

app.UseFileServer(new FileServerOptions()
{
    RequestPath = new PathString("/virtualPath"),
    FileSystem = new PhysicalFileSystem(@".\realPath"),
    EnableDirectoryBrowsing = true
});

app.UseFileServer(new FileServerOptions
{
    RequestPath = new PathString("/assets"),
    FileSystem =
        new EmbeddedResourceFileSystem(typeof(Startup).Assembly)
});
```

Katana CORS middleware

- CORS allows cross-origin Ajax calls
 - Must be configured/allowed by server

```
app.UseCors(new CorsOptions {
    PolicyProvider = new CorsPolicyProvider {
        PolicyResolver = async request => {
            var origin = request.Headers.Get(CorsConstants.Origin);
            if (origin == "http://HostYouTrust")
            {
                var policy = new CorsPolicy() {
                    AllowAnyHeader = true,
                    AllowAnyMethod = true,
                };
                policy.Origins.Add(origin);
                return policy;
            }
            return null;
        }
    }
});
```

Application frameworks

- Many application frameworks support OWIN/Katana
 - Web API
 - SignalR
 - Nancy
 - FubuMVC
 - Simple.Web
 - RavenDB
 - Thinktecture IdentityServer v3

Summary

- OWIN defines a specification for HTTP applications
- Katana is Microsoft's OWIN implementation
- OWIN is the future for building HTTP applications in .NET