OWIN and Katana

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

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

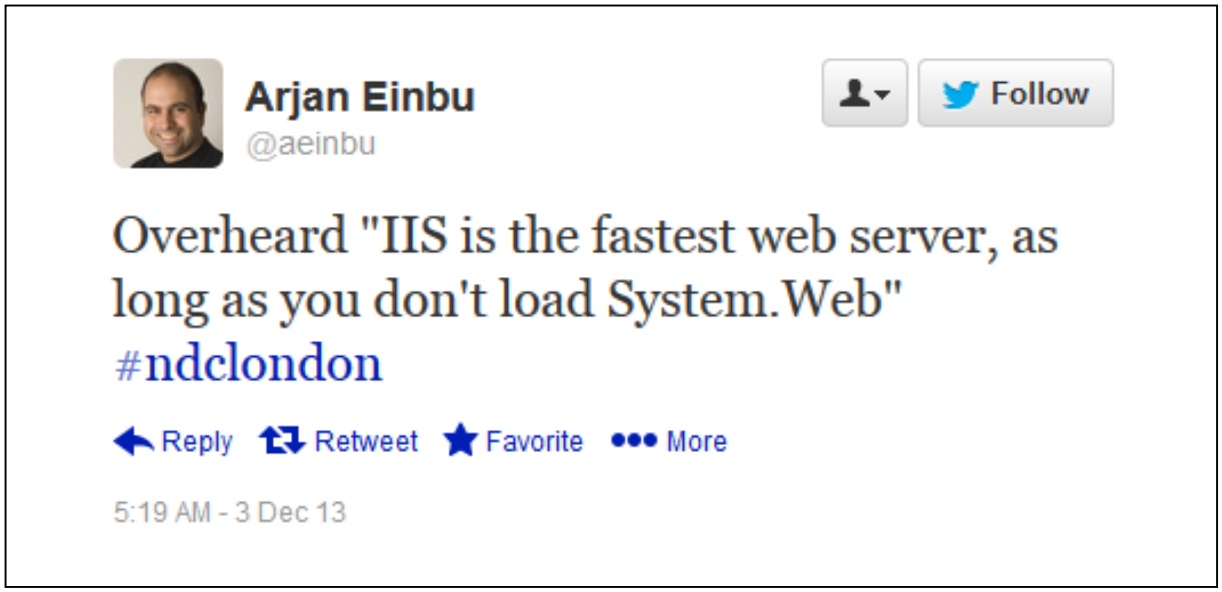
- Community:
  - Open source contributor to thinktecture
  - Open source contributor to ASP.NET
  - Microsoft MVP for ASP.NET

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



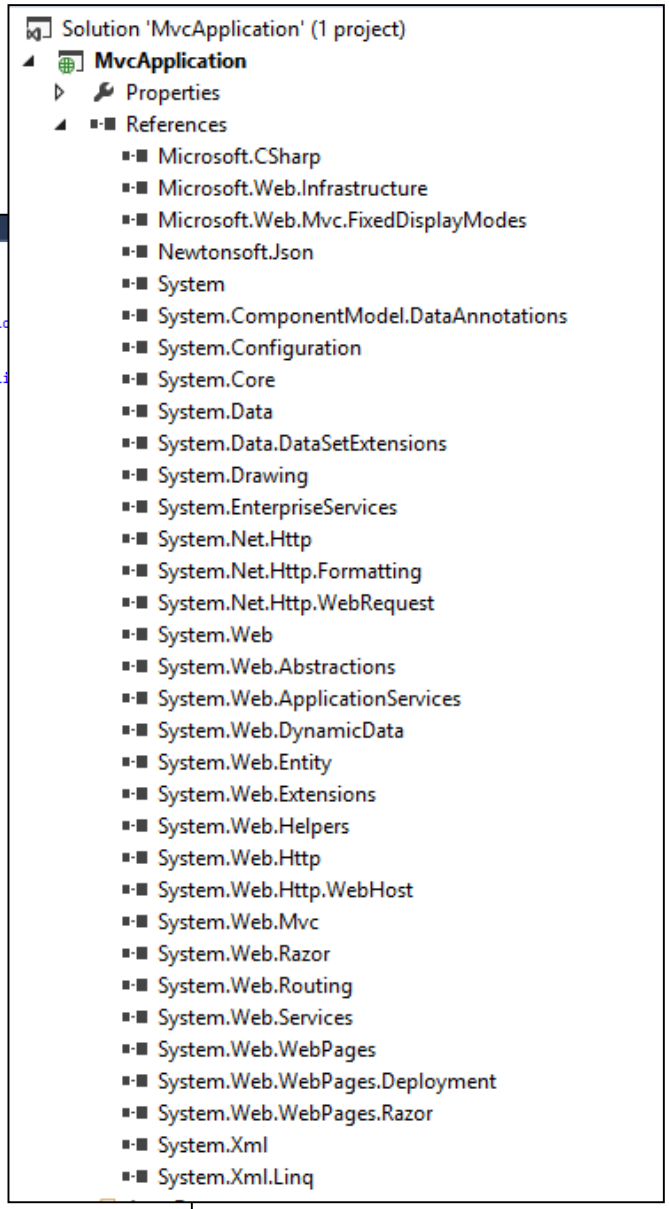
https://twitter.com/aeinbu/status/407816285058514944

OWIN Motivation

- Empty MVC project
  - Lots of baggage

Web.config*

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Versio
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MvcAppli
      providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>
    <authentication mode="None" />
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>
  <system.webServer>
    <modules>
      <remove name="FormsAuthenticationModule" />
    </modules>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-5.0.0.0" newVersion="5.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
      <parameters>
        <parameter value="v11.0" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
```



OWIN Motivation

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

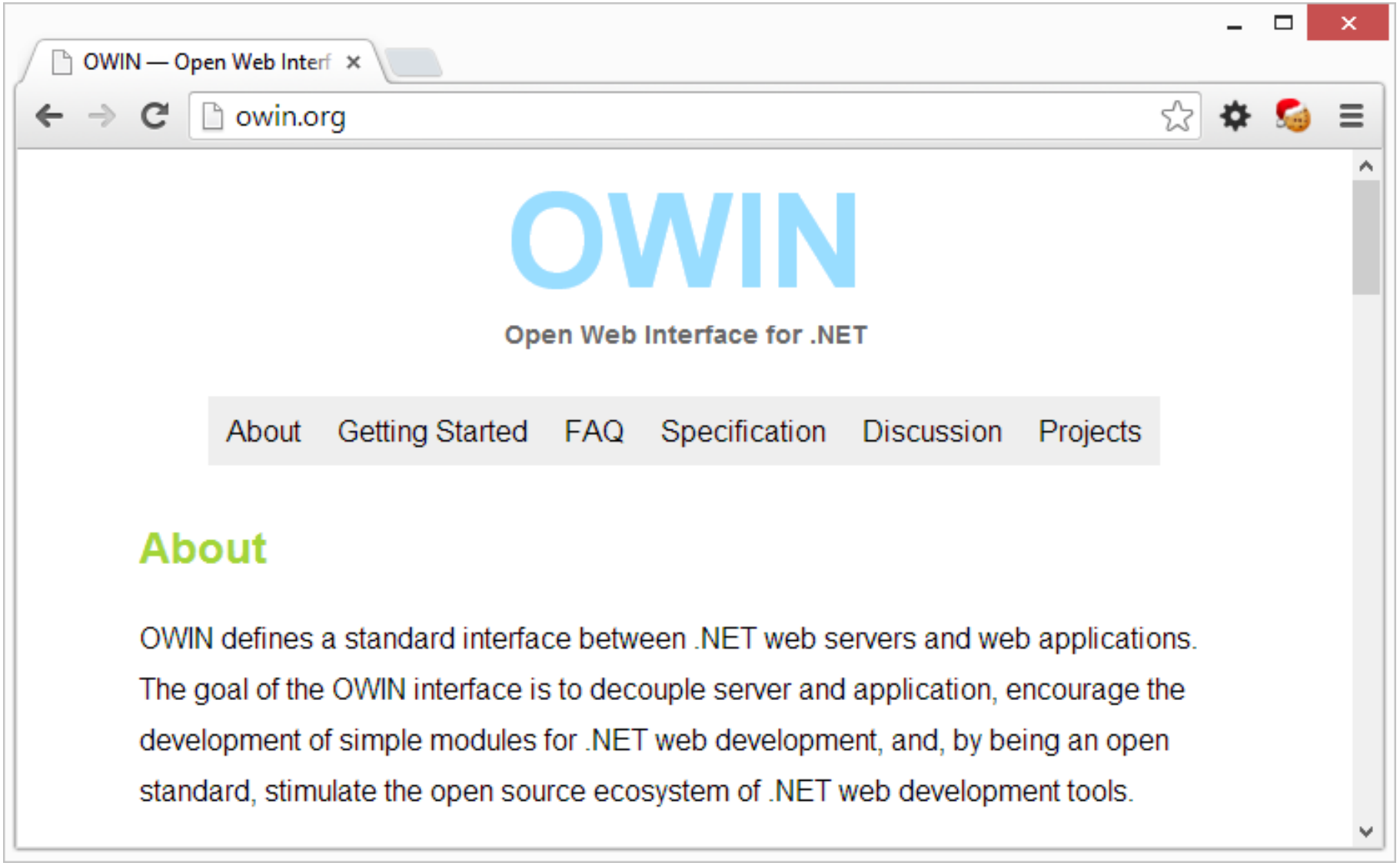
An example of a web server written with Node which responds with 'Hello World':

```
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

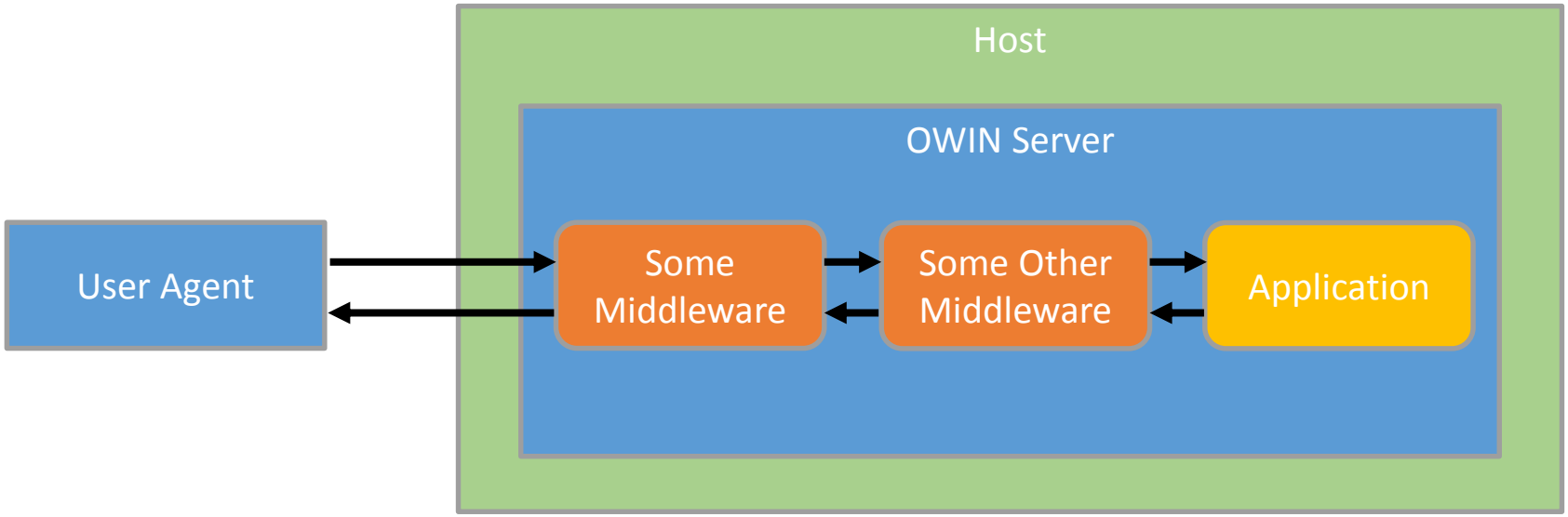

OWIN — Open Web Interf
owin.org
OWIN
Open Web Interface for .NET
About
Getting Started
FAQ
Specification
Discussion
Projects
About
OWIN defines a standard interface between .NET web servers and web applications. The goal of the OWIN interface is to decouple server and application, encourage the development 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.
  - *AppBuilderUseExtensions*
  - *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	IDictionary<string, string[]>
owin.ResponseStatusCode	int
owin.ResponseHeaders	IDictionary<string, string[]>
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
    }
}
```

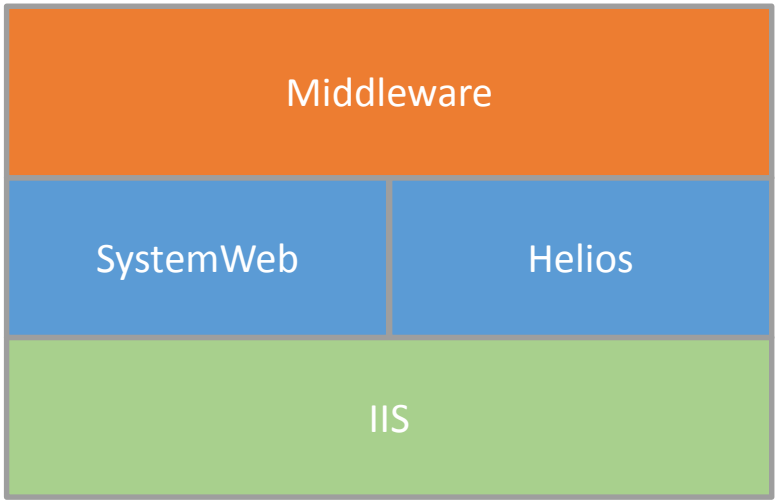
Configuring middleware

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

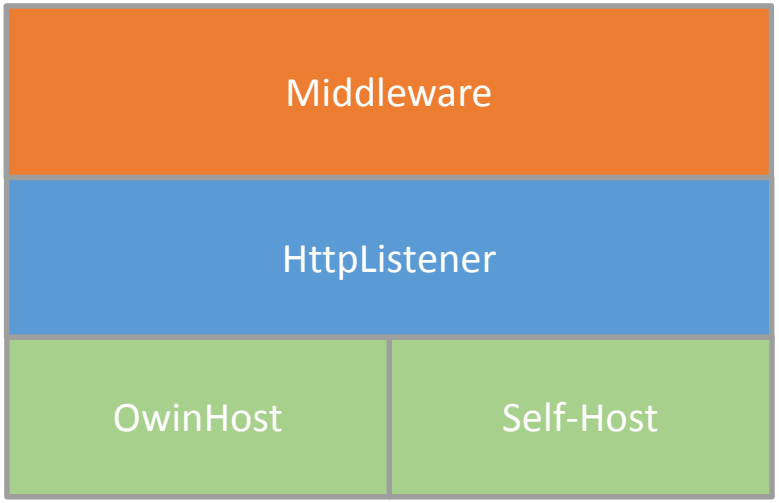
```
public class Startup
{
    public void Configuration(IAppBuilder app)
    {
        app.Use(typeof(SomeMiddleware));
        app.Use(typeof(SomeOtherMiddleware));
    }
}
```

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>");
    }
}
```

AppBuilderUseExtensions

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

```
public void Configuration(IAppBuilder 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