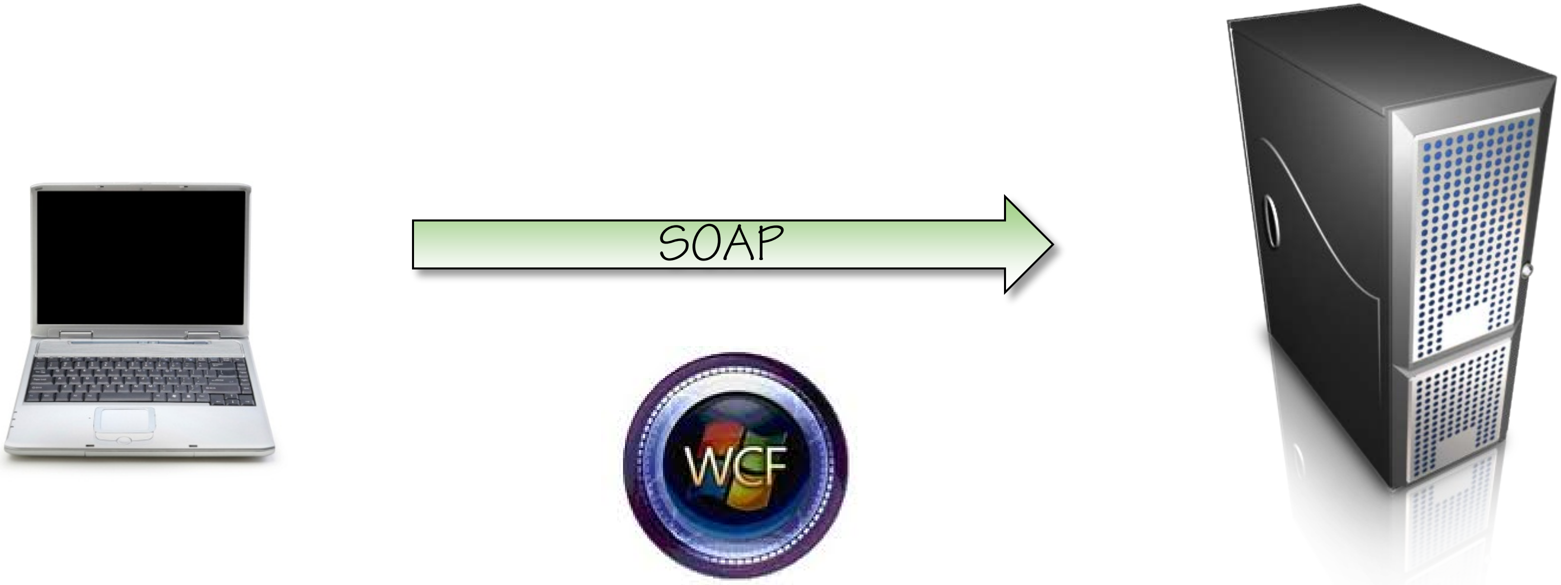
MVC 4 – WebAPI

GET Resources

Scott Allen

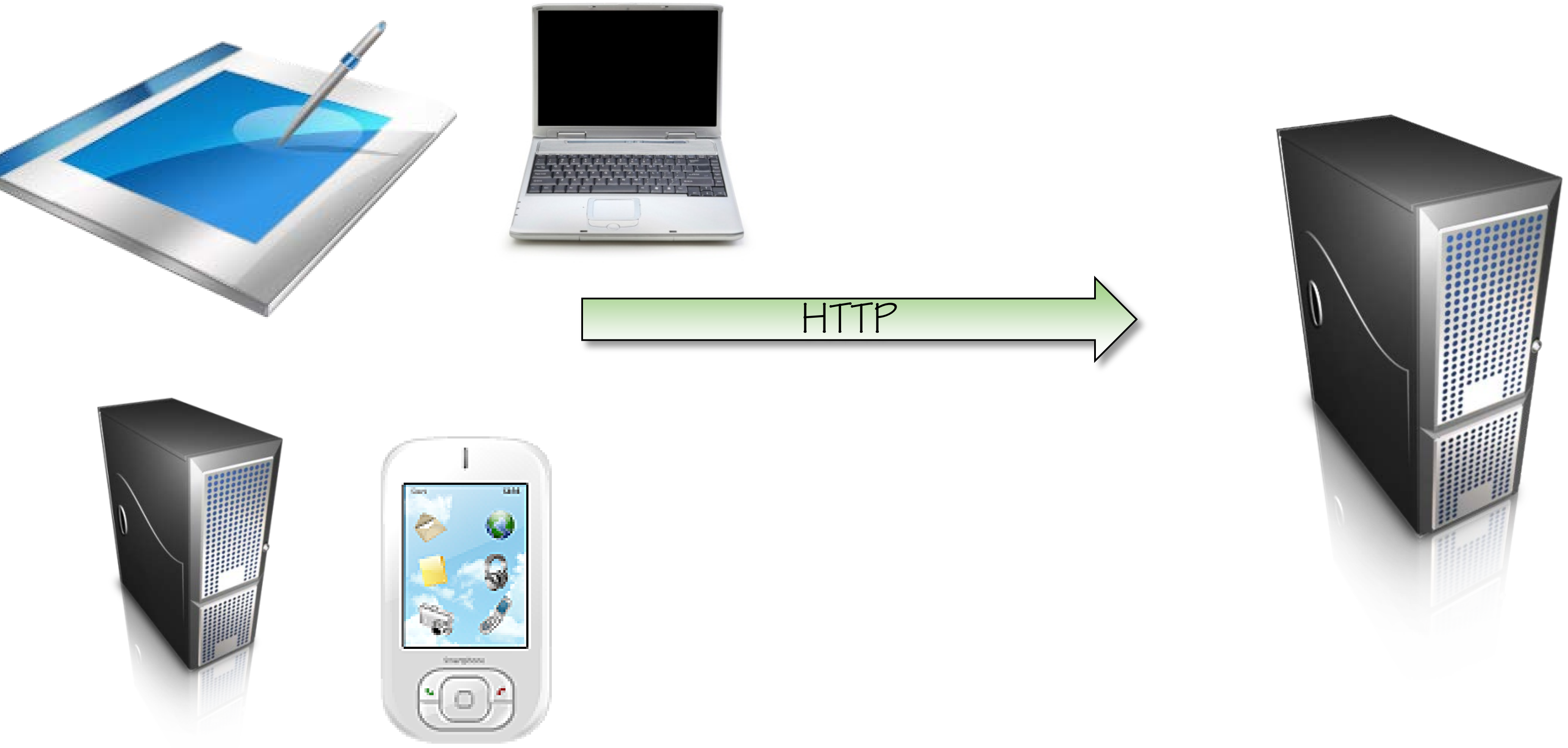
pluralsight
hardcore developer training

Web Services



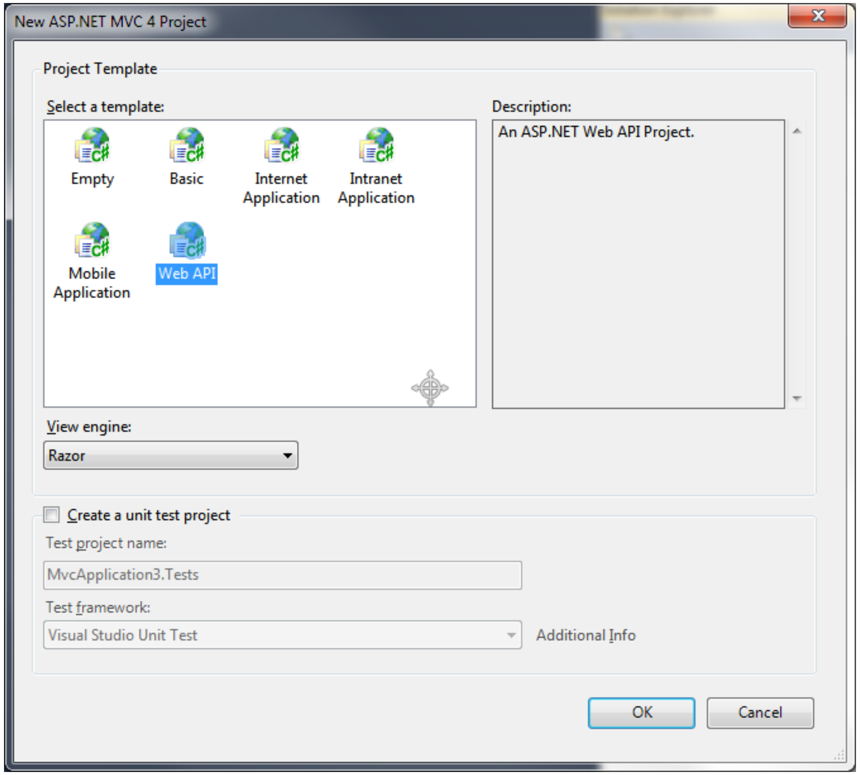
SOAP

HTTP Web Services



Getting Started

- Create a WebAPI project
- Create an ASP.NET project
- Create any project!
  - install-package Microsoft.AspNet.WebApi.SelfHost



Select a template: Web API

Description:
An ASP.NET Web API Project.

View engine: Razor

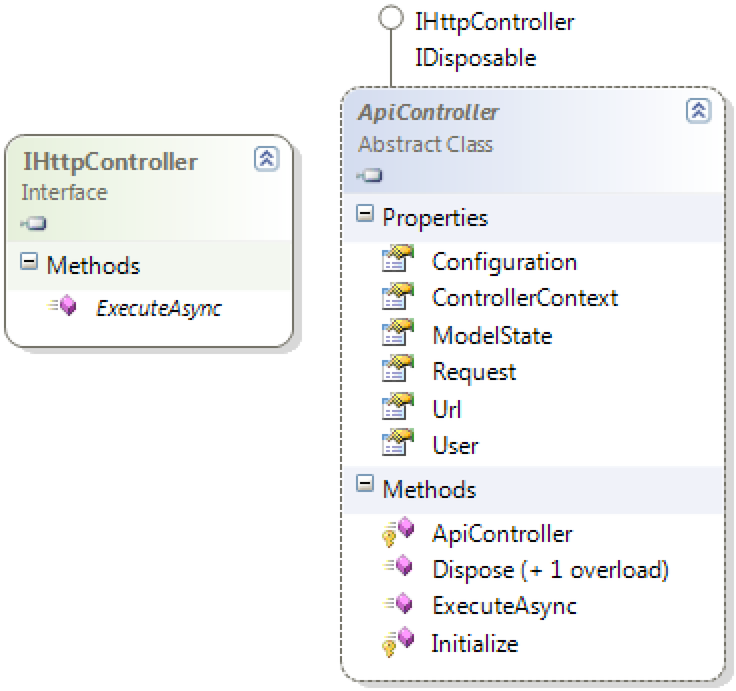
□ Create a unit test project

Test project name: MvcApplication3.Tests

Test framework: Visual Studio Unit Test

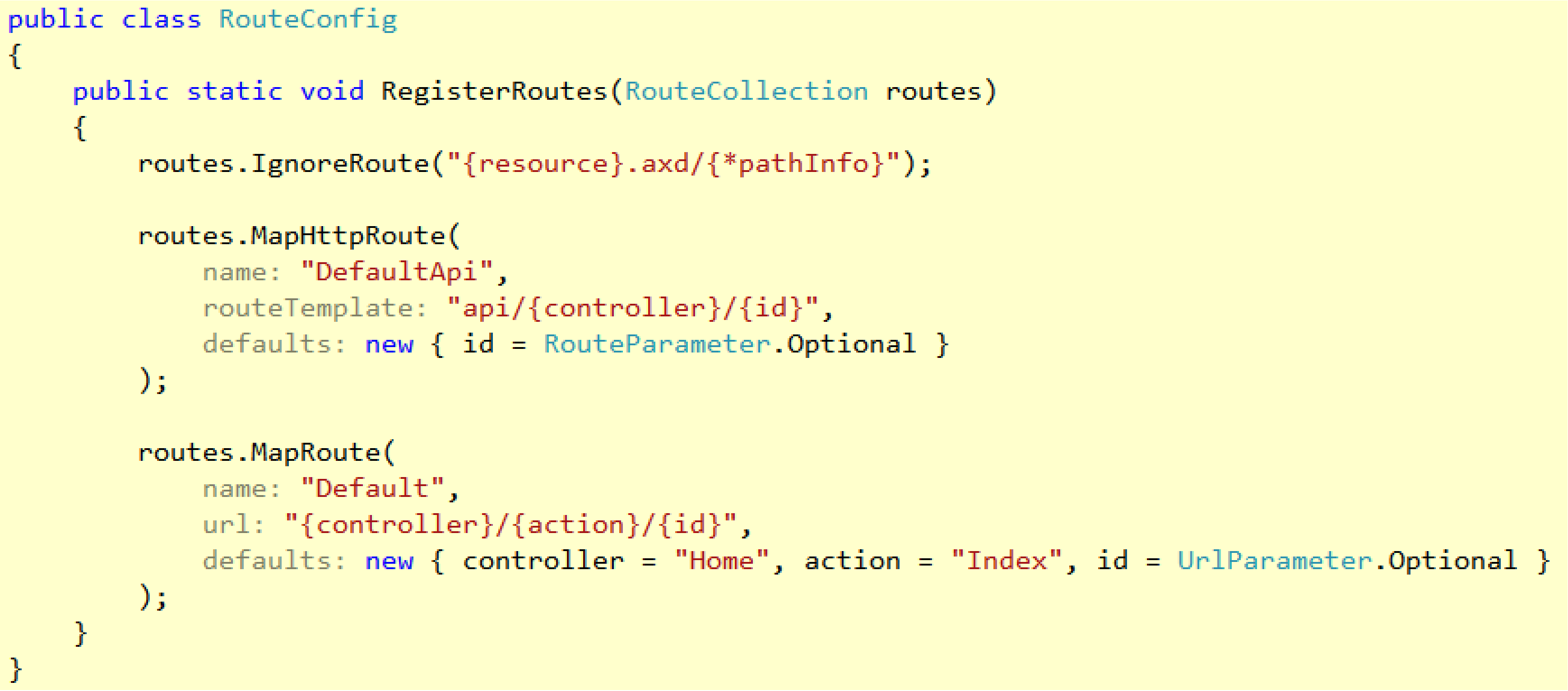
Controllers

- WebAPI uses an ApiController base class



Routing

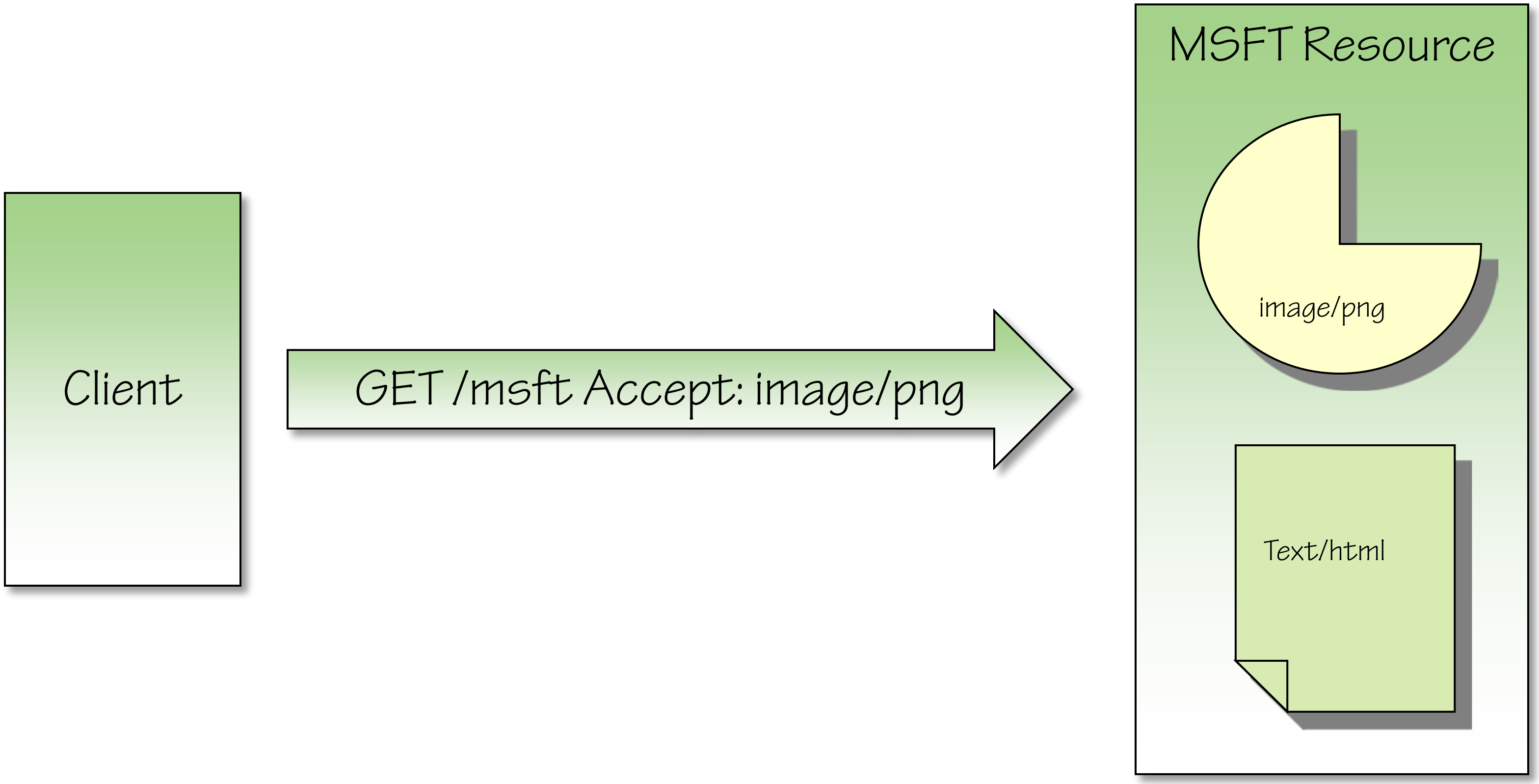
- Routes by HTTP method (GET, POST, PUT, DELETE)
- Other methods available via [AcceptVerbs]
- Looks at the start of an action name



```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

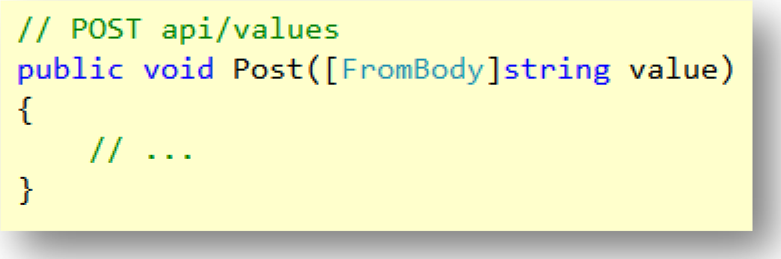
Content Negotiation

- Core mechanism of HTTP
- Client specifies desired formats using Accept header



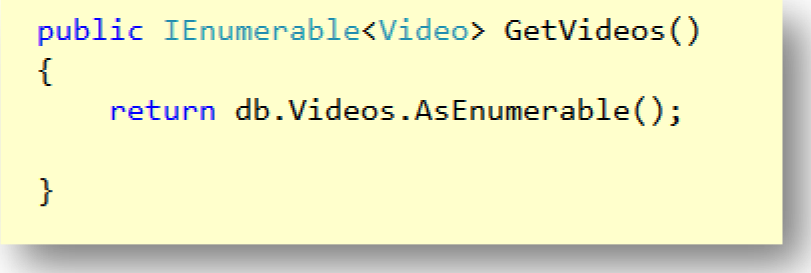
Action Parameters

- Primitive types (assumed to not be in message body)
- Complex types (assumed to be in the message body)
- Only a single model allowed from the message body

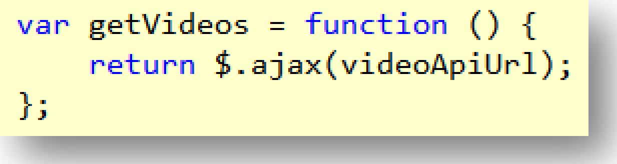


```
// POST api/values
public void Post([FromBody]string value)
{
    // ...
}
```

GET Pattern

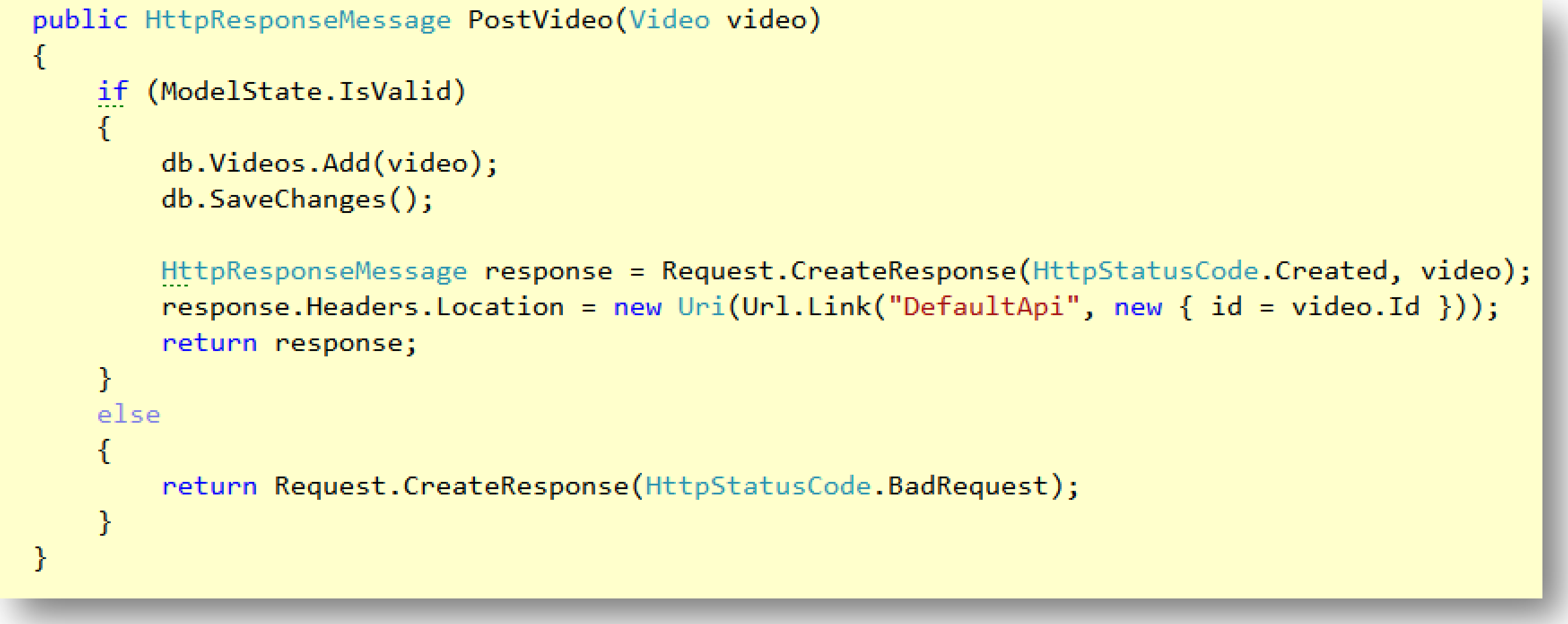


```
public IEnumerable<Video> GetVideos()
{
    return db.Videos.AsEnumerable();

}
```

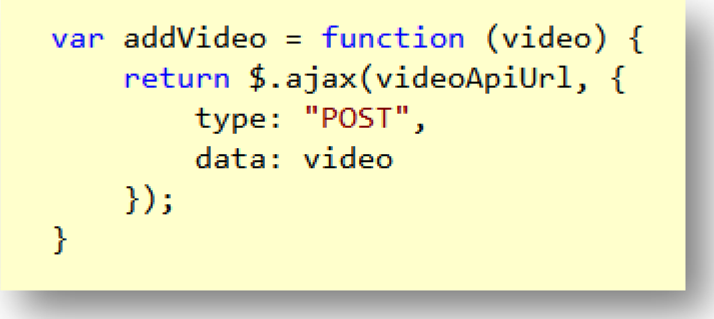


```
var getVideos = function () {
    return $.ajax(videoApiUrl);
};
```

POST Pattern

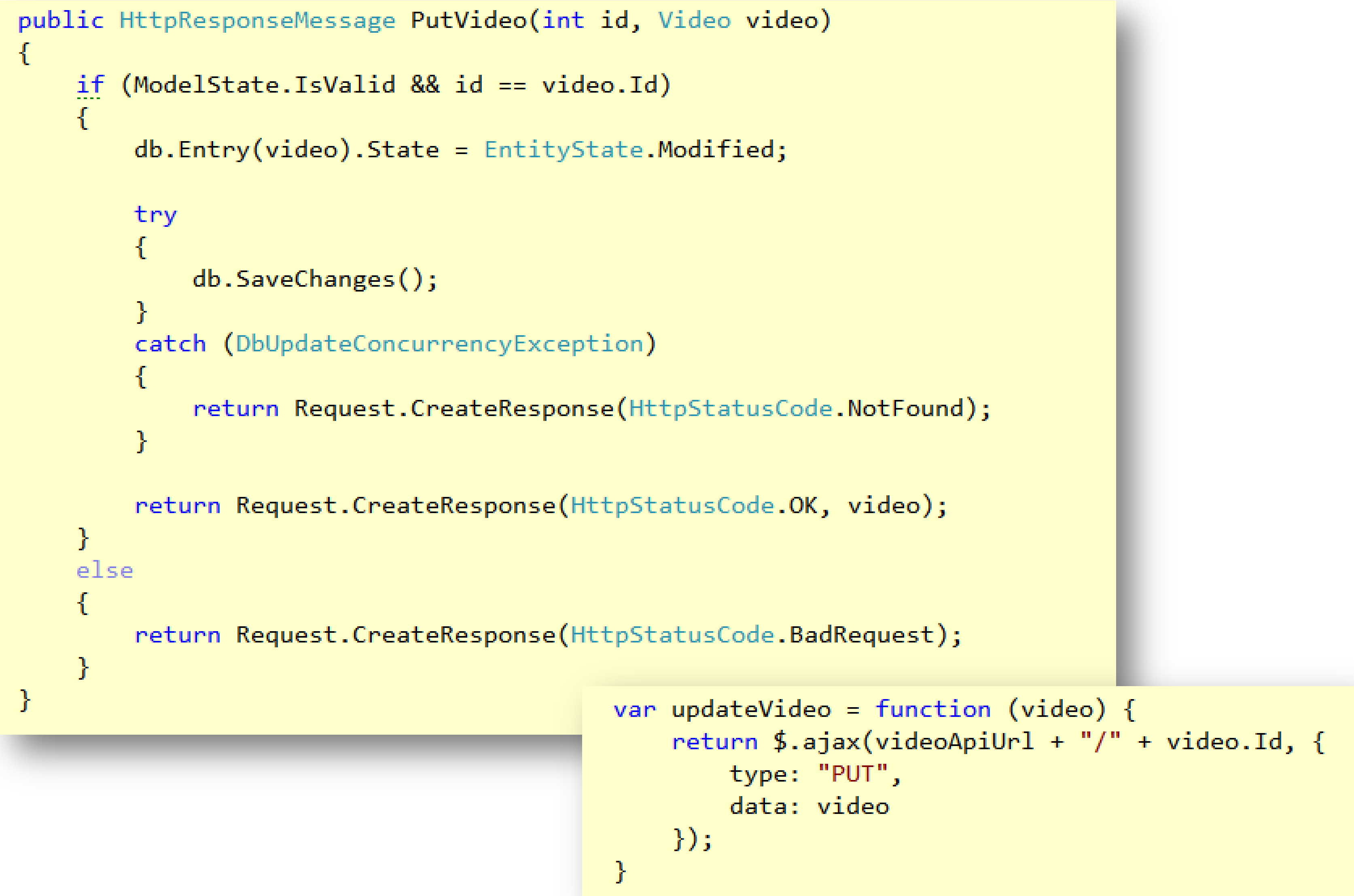


```
public HttpResponseMessage PostVideo(Video video)
{
    if (ModelState.IsValid)
    {
        db.Videos.Add(video);
        db.SaveChanges();

        HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created, video);
        response.Headers.Location = new Uri(Url.Link("DefaultApi", new { id = video.Id }));
        return response;
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.BadRequest);
    }
}
```



```
var addVideo = function (video) {
    return $.ajax(videoApiUrl, {
        type: "POST",
        data: video
    });
}
```

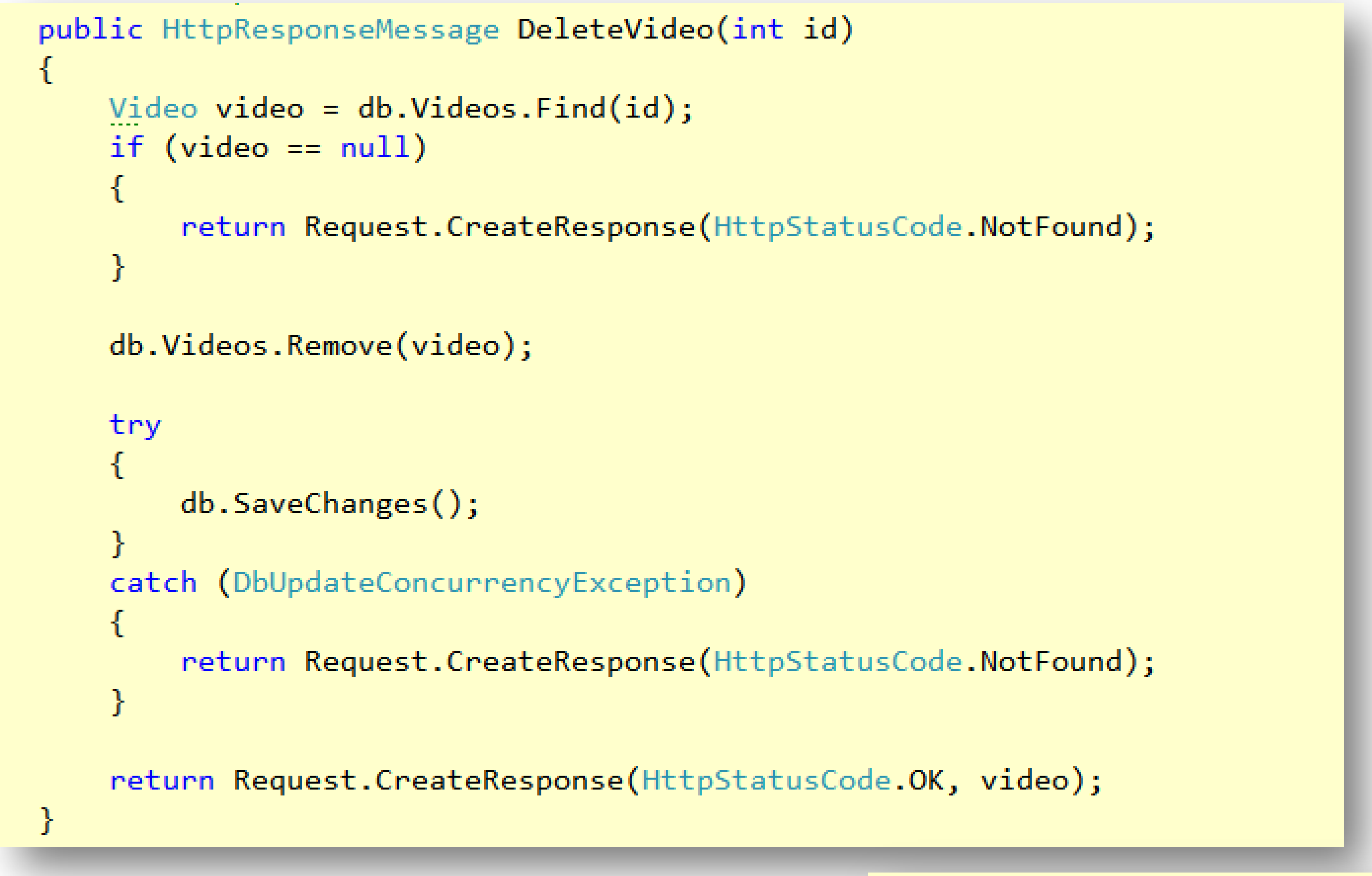
PUT Pattern



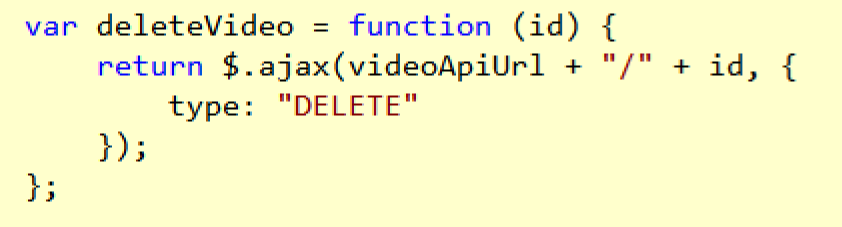
```
public HttpResponseMessage PutVideo(int id, Video video)
{
    if (ModelState.IsValid && id == video.Id)
    {
        db.Entry(video).State = EntityState.Modified;

        try
        {
            db.SaveChanges();
        }
        catch (DbUpdateConcurrencyException)
        {
            return Request.CreateResponse(HttpStatusCode.NotFound);
        }

        return Request.CreateResponse(HttpStatusCode.OK, video);
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.BadRequest);
    }
}
```

```
var updateVideo = function (video) {
    return $.ajax(videoApiUrl + "/" + video.Id, {
        type: "PUT",
        data: video
    });
}
```

DELETE Pattern

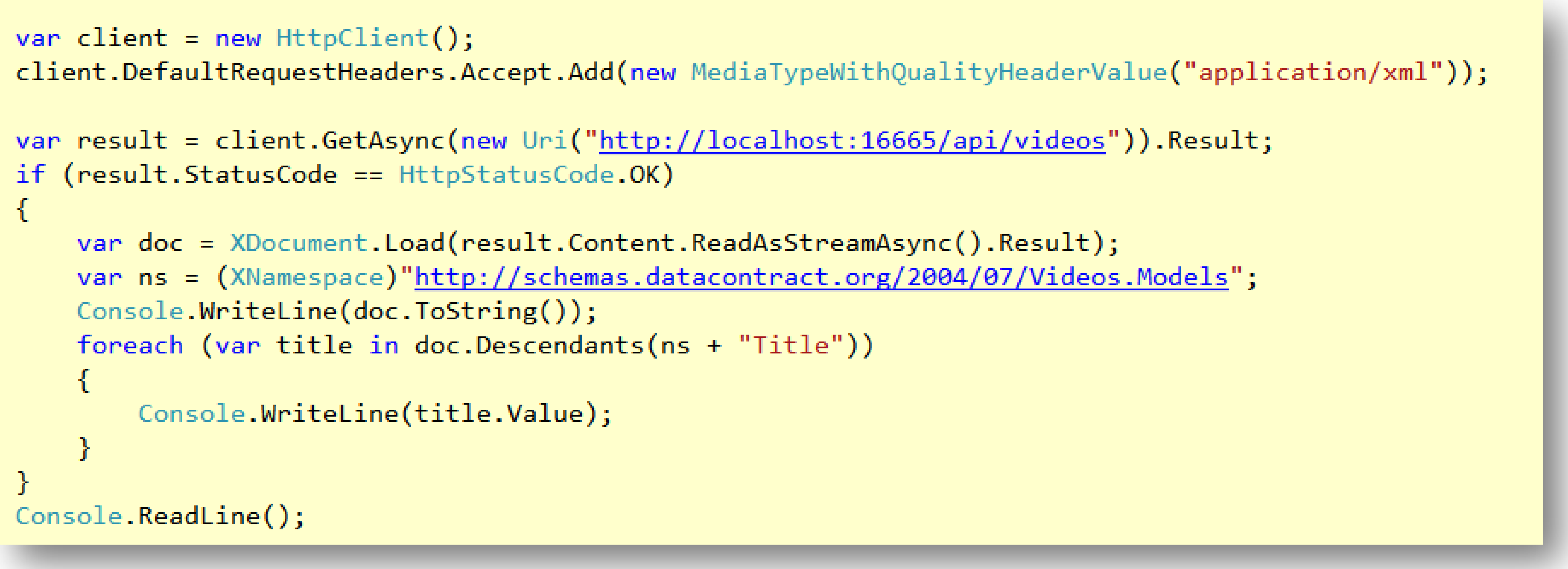


```
public HttpResponseMessage DeleteVideo(int id)
{
    Video video = db.Videos.Find(id);
    if (video == null)
    {
        return Request.CreateResponse(HttpStatusCode.NotFound);
    }

    db.Videos.Remove(video);

    try
    {
        db.SaveChanges();
    }
    catch (DbUpdateConcurrencyException)
    {
        return Request.CreateResponse(HttpStatusCode.NotFound);
    }

    return Request.CreateResponse(HttpStatusCode.OK, video);
}
```



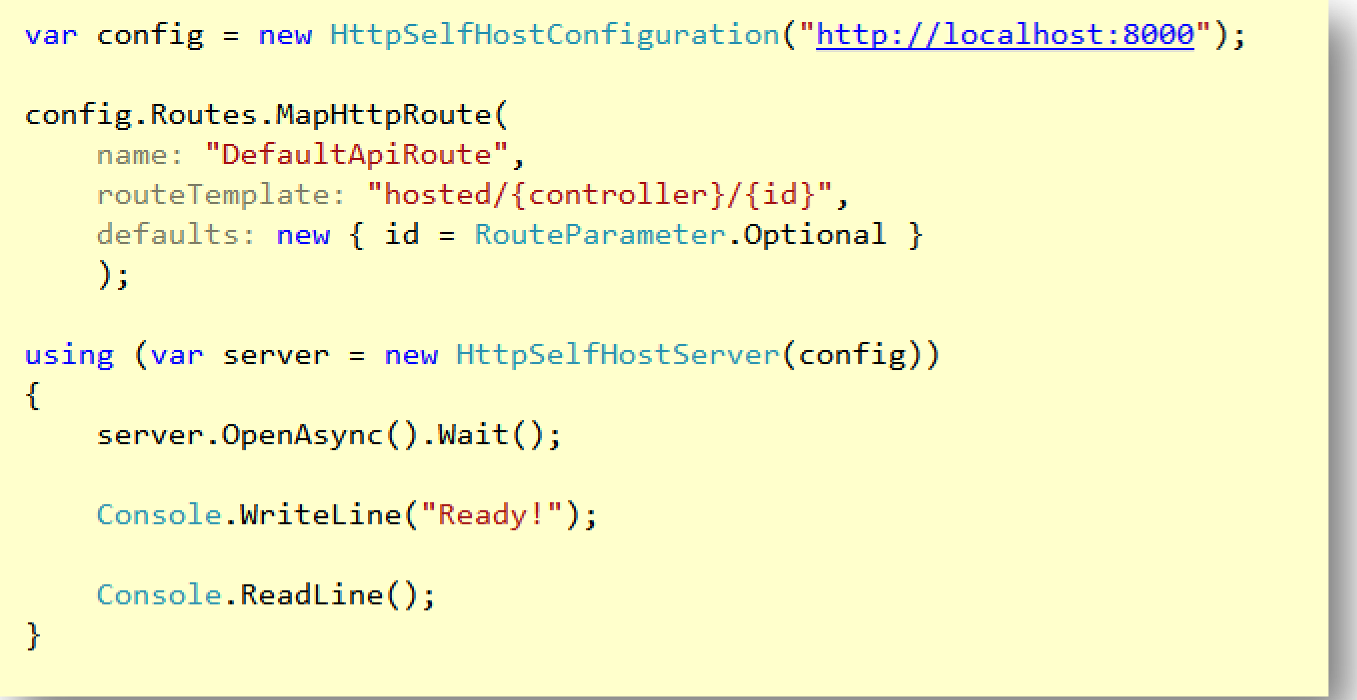
```
var deleteVideo = function (id) {
    return $.ajax(videoApiUrl + "/" + id, {
        type: "DELETE"
    });
};
```

HttpClient



```
var client = new HttpClient();
client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/xml"));

var result = client.GetAsync(new Uri("http://localhost:16665/api/videos")).Result;
if (result.StatusCode == HttpStatusCode.OK)
{
    var doc = XDocument.Load(result.Content.ReadAsStreamAsync().Result);
    var ns = (XNamespace)"http://schemas.datacontract.org/2004/07/Videos.Models";
    Console.WriteLine(doc.ToString());
    foreach (var title in doc.Descendants(ns + "Title"))
    {
        Console.WriteLine(title.Value);
    }
}
Console.ReadLine();
```

Self Hosting



```
var config = new HttpSelfHostConfiguration("http://localhost:8000");

config.Routes.MapHttpRoute(
    name: "DefaultApiRoute",
    routeTemplate: "hosted/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
    );

using (var server = new HttpSelfHostServer(config))
{
    server.OpenAsync().Wait();

    Console.WriteLine("Ready!");

    Console.ReadLine();
}
```

Summary

- WebAPI embraces HTTP
- Scalable, interoperable, flexible
- Includes client and self hosting capabilities