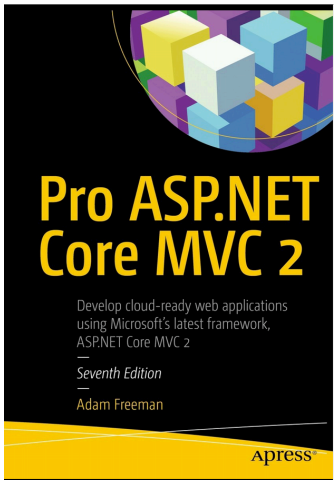
ASP.NET Core, Microservices and Azure Service Fabric

Dominic Duggan
Stevens Institute of Technology

1

Reading



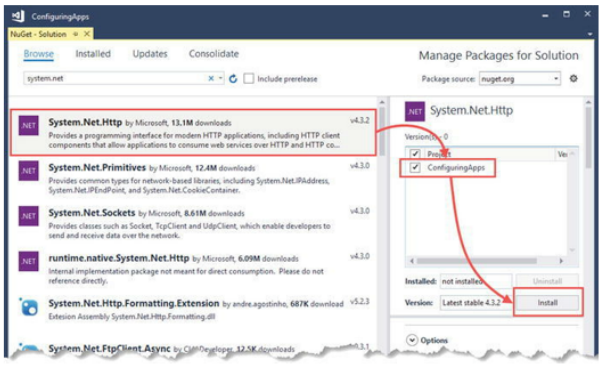
2

CONFIGURING MVC APPLICATIONS

3

Add Functionality

- Configuration file: *project-name*`.csproj`
- Add packages with NuGet package manager



4

Add Functionality

- Configuration file: *project-name*.csproj
- Add packages from command line:

```
dotnet add package System.Net.Http --version
4.3.2
```

5

Important Packages

- Metapackage for ASP.NET Core MVC & EF
  `Microsoft.AspNetCore.All`
- Support for making HTTP requests:
  `System.Net.Http`

6

Main Program: Program.cs

```
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
namespace HelloWorld{
public class Program {
  public static void Main(string[] args) {
    BuildWebHost(args).Run();
  }
  public static IWebHost BuildWebHost(string[] args) {
    WebHost.CreateDefaultBuilder(args)
                      .UseStartup<Startup>()
                      .Build();
  }
}
```

7

Main Program: Program.cs

```
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
namespace HelloWorld{
public class Program {
  public static void Main(string[] args) {
    BuildWebHost(args).Run();
  }
  public static IWebHost BuildWebHost(string[] args) {
    WebHost.CreateDefaultBuilder(args)
                      .UseStartup<Startup>()
                      .Build();
  }
}
```

8

Main Program: Program.cs

```
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
namespace HelloWorld{
public class Program {
  public static void Main(string[] args) {
    BuildWebHost(args).Run();
  }
  public static IWebHost BuildWebHost(string[] args) {
    return new WebHostBuilder() ...
                      .UseStartup<Startup>()
                      .Build();
  }
}
```

9

Main Program: Program.cs

Web server

```
return new WebHostBuilder()
  .UseKestrel()
  .UseContentRoot(Directory.GetCurrentDirectory())
  .ConfigureAppConfiguration((context, config) => ...)
        ... E.g. call config.addJsonFile(...) ...

  .ConfigureLogging((hostingContext,logging) => ...)

  .UseDefaultServiceProvider((context,options) => ...



  .UseStartup<Startup>()
  .Build();
```

10

Main Program: Program.cs

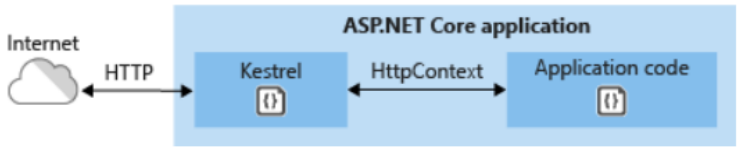
```
return new WebHostBuilder()
  .UseKestrel()
  .UseContentRoot(Directory.GetCurrentDirectory())
  .ConfigureAppConfiguration((context, config) => ...)
        ... E.g. call config.addJsonFile(...) ...

  .ConfigureLogging((hostingContext,logging) => ...)

  .UseDefaultServiceProvider((context,options) => ...


  .UseStartup<Startup>()
  .Build();
```

Dependency injection framework

11

Main Program: Program.cs

```
return new WebHostBuilder()
  .UseKestrel()
  .UseContentRoot(Directory.GetCurrentDirectory())
  .ConfigureAppConfiguration((context, config) => ...)
```

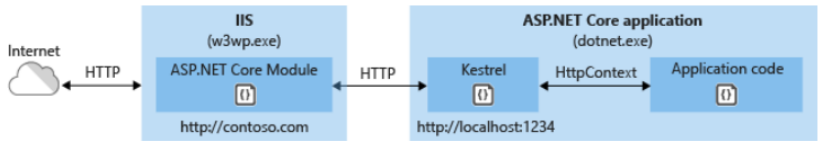



```
.)
...

  .UseStartup<Startup>()
  .Build();
```

12

Main Program: Program.cs

```
return new WebHostBuilder()
  .UseKestrel()
  .UseContentRoot(Directory.GetCurrentDirectory())
  .ConfigureAppConfiguration((context, config) => ...)
```




```
  .UseIISIntegration()

  .UseStartup<Startup>()
  .Build();
```

13

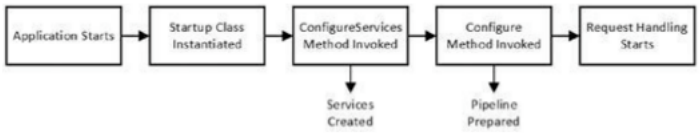
Basic Main Program

```
namespace HelloWorld{
public class Program {
  ...
  public static IWebHost BuildWebHost(string[] args) {
    return new WebHostBuilder()
                 .UseKestrel()
                 .UseContentRoot(
                      Directory.GetCurrentDirectory())
                 .UseIISIntegration()
                 .UseStartup<Startup>()
                 .Build();
  }
}
```

14

Startup Class

- Set up services (ConfigureServices)
- Set up request pipeline (Configure)
  - Handle incoming HTTP requests
  - Produce responses



15

Startup Class

```
namespace HelloWorld {
  public class Startup {
     public void ConfigureServices
             (IServiceCollection services) {

     }
     public void Configure
             (IApplicationBuilder app,
              IHostingEnvironment env) {
       app.Run(async (context) => {
          await context.Response
            .WriteAsync("Hello World!");
          });
     }
  }
}
```

16

Startup Class

```
namespace HelloWorld {
  public class Startup {
     public void ConfigureServices
             (IServiceCollection services) {
       services.AddMvc();
     }
     public void Configure
             (IApplicationBuilder app,
              IHostingEnvironment env) {
       app.UseMvcWithDefaultRoute();
     }
  }
}
```

17

Startup Class

```
namespace HelloWorld {
  public class Startup {
     public void ConfigureServices
             (IServiceCollection services) {
       services.AddSingleton<UptimeService>();
       services.AddMvc();
     }
     public void Configure
             (IApplicationBuilder app,
              IHostingEnvironment env) {
       app.UseMvcWithDefaultRoute();
     }
  }
}
```

18

Example Service

```
public class UptimeService {

  private Stopwatch timer;

  public UptimeService() {
    timer = Stopwatch.StartNew();
  }

  public long Uptime =>
    timer.ElapsedMilliseconds;
  }
}
```

19

Using Service

```
public class HomeController : Controller {

  private UptimeService uptime;

  public HomeController(UptimeService up) =>
    uptime = up;

  public ViewResult Index() =>
    View(...uptime.Uptime...);

}
```

Dependency injection

20

DEFINING MIDDLEWARE

21

Middleware Component

- Constructor takes `RequestDelegate` as argument
  - Dependency injection
- Defines `Invoke(HttpContext ctxt)` method

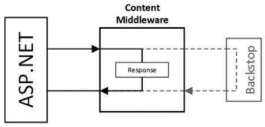
22

Example Middleware Component

```
using Microsoft.AspNetCore.Http;
public class ContentMiddleware {
  private RequestDelegate nextDelegate;
  public ContentMiddleware(RequestDelegate next) =>
    nextDelegate = next;
  public async Task Invoke(HttpContext httpContext) {
    if (httpContext.Request.Path.ToString().ToLower() ==
              "/middleware") {
        await httpContext.Response.WriteAsync(
                "This is from the content middleware",
                Encoding.UTF8);
      } else {
        await nextDelegate.Invoke(httpContext);
      }
    }
  }
}
```

23

Example Middleware Component

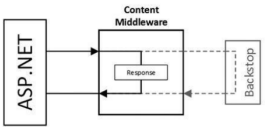



```
using Microsoft.AspNetCore.Http;
public class ContentMiddleware {
  private RequestDelegate nextDelegate;
  public ContentMiddleware(RequestDelegate next) =>
    nextDelegate = next;
  public async Task Invoke(HttpContext httpContext) {
    if (httpContext.Request.Path.ToString().ToLower() ==
              "/middleware") {
        await httpContext.Response.WriteAsync(
                "This is from the content middleware",
                Encoding.UTF8);
      } else {
        await nextDelegate.Invoke(httpContext);
      }
    }
  }
}
```

24

Configuring Middleware

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    services.AddSingleton<UptimeService>();
    services.AddMvc();
  }

  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    app.UseMiddleware<ContentMiddleware>();
  }
}
```



25

Middleware and Services

```
using Microsoft.AspNetCore.Http;
public class ContentMiddleware {
  private RequestDelegate nextDelegate;
  private UptimeService uptime;
  public ContentMiddleware(RequestDelegate next, UptimeService up){
    nextDelegate = next;
    uptime = up;
  }
  public async Task Invoke(HttpContext httpContext) {
    if (httpContext.Request.Path.ToString() == "/middleware") {
        await httpContext.Response.WriteAsync(
                "This is from the content middleware”
                + uptime.Uptime, Encoding.UTF8);
      } else {
        await nextDelegate.Invoke(httpContext);
      }
    }
  }
}
```

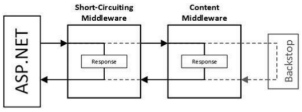
26

Short-Circuiting Middleware

- Intercept requests before content generation
- Example: Return 403 if User-Agent header contains “edge”

Short-Circuiting Middleware

```
public class SCMiddleware {

  private RequestDelegate nextDelegate;

  public SCMiddleware(RequestDelegate next) =>
    nextDelegate = next;

  public async Task Invoke(HttpContext httpContext) {
    if (httpContext.Request.Headers["User-Agent”]
           .Any(h => h.ToLower().Contains("edge"))) {
      httpContext.Response.StatusCode = 403;
    } else {
      await nextDelegate.Invoke(httpContext);
    }
  }
}
```



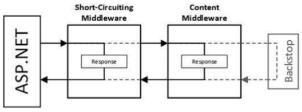
Request-Editing Middleware

- Doesn't generate a response
- Change requests before they reach other components
- Example: Add browser type property ("`EdgeBrowser`") to request based on `User-Agent` header

29

Request-Editing Middleware

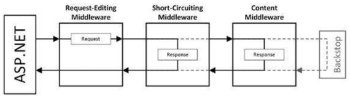
```
public class BTypeMiddleware {

  private RequestDelegate nextDelegate;

  public BTypeMiddleware(RequestDelegate next) =>
    nextDelegate = next;

  public async Task Invoke(HttpContext httpContext) {
    httpContext.Items["EdgeBrowser"] =
       httpContext.Request.Headers["User-Agent"]
           .Any(h => h.ToLower().Contains("edge"));
    await nextDelegate.Invoke(httpContext);
  }
}
```



30

Combining Middlewares

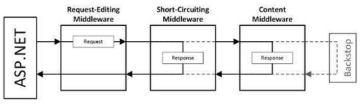
```
public class SCMiddleware {

  private RequestDelegate nextDelegate;

  public SCMiddleware(RequestDelegate next) =>
    nextDelegate = next;

  public async Task Invoke(HttpContext httpContext) {
    if (httpContext.Items["EdgeBrowser"] as bool? == true) {
      httpContext.Response.StatusCode = 403;
    } else {
      await nextDelegate.Invoke(httpContext);
    }
  }
}
```



31

Configuring Middleware

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

  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    app.UseMiddleware<BTypeMiddleware>();
    app.UseMiddleware<SCMiddleware>();
    app.UseMiddleware<ContentMiddleware>();
  }
}
```



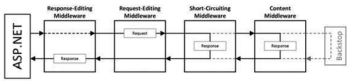
32

Response-Editing Middleware

- Change responses from other components
- Example: Log responses

33

Request-Editing Middleware

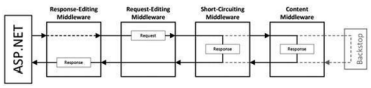



```
public class ErrorMiddleware {

  private RequestDelegate nextDelegate;

  public ErrorMiddleware(RequestDelegate next) =>
    nextDelegate = next;

  public async Task Invoke(HttpContext httpContext) {
    await nextDelegate.Invoke(httpContext);
    if (httpContext.Response.StatusCode == 403) {
      await httpContext.Response
            .WriteAsync("Edge not supported", Encoding.UTF8);
    } else if (httpContext.Response.StatusCode == 404) {
      await httpContext.Response
            .WriteAsync("No content", Encoding.UTF8);
    }
  }
}
```

34

Configuring Middleware

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

  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    app.UseMiddleware<ErrorMiddleware>();
    app.UseMiddleware<BTypeMiddleware>();
    app.UseMiddleware<SCMiddleware>();
    app.UseMiddleware<ContentMiddleware>();
  }
}
```



35

Add MVC With Default Routing

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    ... services.AddMvc();
  }
  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    app.UseMiddleware<ErrorMiddleware>();
    app.UseMiddleware<BTypeMiddleware>();
    app.UseMiddleware<SCMiddleware>();
    app.UseMiddleware<ContentMiddleware>();
    app.UseMvcWithDefaultRoute();
  }
}
```

36

Add MVC With Explicit Routing

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    ... services.AddMvc();
  }
  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    ...
    app.UseMvc(routes => {
      routes.MapRoute(
        name: “default”,
        template: “{controller=Home}/{action=Index}/{id?}”);
    });
  }
}
```

37

Hosting Environment Properties

- ApplicationName
- EnvironmentName
  - Development, staging, production
- ContentRootPath, WebRootPath
  - Static content can be provided by middleware
- ContentRootFileProvider, WebRootFileProvider

38

Environment-Specific Middleware

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    ... services.AddMvc();
  }
  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    if (env.IsDevelopment()) {
      app.UseMiddleware<ErrorMiddleware>();
      ...
      app.UseMiddleware<ContentMiddleware>();
    }
    app.UseMvc(routes => ...);
  }
}
```

39

Exception-Handling

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    ... services.AddMvc();
  }
  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    if (env.IsDevelopment()) {
      app.UseDeveloperExceptionPage();
      app.UseStatusCodePages();
    } else {
      app.UseExceptionhandler("/Home/Error");
    }
    app.UseMvc(routes => ...);
  }
}
```

40

Providing Static Content

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) { ...
  }
  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    if (env.IsDevelopment()) {
      app.UseDeveloperExceptionPage();
      app.UseStatusCodePages();
    } else {
      app.UseExceptionhandler("/Home/Error");
    }
    app.UseStaticFiles();
    app.UseMvc(routes => ...);
  }
}
```

41

APPLICATION CONFIGURATION

42

Configuring Application

```
public class Program {
public static IWebHost BuildWebHost(string[] args) {
  return new WebHostBuilder()
    .UseKestrel()
    .UseContentRoot(Directory.GetCurrentDirectory())
    .ConfigureAppConfiguration((context, config) => {
      config.AddJsonFile("appsettings.json",
                 optional: true, reloadOnChange: true);
      config.AddEnvironmentVariables();
      if (args != null) {
        config.AddCommandLine(args);
      }
    })
    ...
    .Build();
}
```

43

Configuring Logging

```
public class Program {
public static IWebHost BuildWebHost(string[] args) {
  return new WebHostBuilder()
    .UseKestrel()
    .UseContentRoot(Directory.GetCurrentDirectory())
    .ConfigureAppConfiguration((context, config) => )
    .ConfigureLogging((hostingContext, logging) => {
      logging.AddConfiguration(
                 hostingContext.Configuration
                            .GetSection("Logging"));
      logging.AddConsole();
      logging.AddDebug();
    })
    ...
    .Build();
}
```

44

Example Configuration File (appSeeings.json)

```
{ ...
  "Logging": {
      "LogLevel": {
           "Default": "Debug",
           "System": "Information",
           "Microsoft": "Information”
      }
  ...
}
```

45

Logging Levels

- Trace
  - Development only
- Debug
- Information
- Warning
- Error
- Critical
  - Catastrophic failure
- None
  - Disable logging messages

46

Configuring MVC Services

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    services.AddSingleton<UptimeService>();
    services.AddMvc()
            .AddMvcOptions(options => {
      options.RespectBrowserAcceptHeader = true;
    });
  }
  public void Configure
         (IApplicationBuilder app,
          IHostingEnvironment env) {
    ...
  }
}
```

47

MVC Builder Extension Methods

- AddMvcOptions
- AddFormatterMappings
- AddJsonOptions
- AddRazorOptions
- AddViewOptions

48

Selected MvcOptions Properties

- Conventions
  - For controllers and actions
- Filters
- FormatterMappings
- InputFormatters
- OutputFormatters
- ModelValidatorProviders
- RespectBrowserAcceptHeader

49

Environment-Specific Configuration

```
public class Startup {
  public void ConfigureServices
         (IServiceCollection services) {
    services.AddSingleton<UptimeService>();
    services.AddMvc()
            .AddMvcOptions(...);
  }
  public void ConfigureDevelopmentServices
         (IServiceCollection services) {
    services.AddSingleton<UptimeService>();
    services.AddMvc();
  }
  public void Configure(...) { ...
  }
  public void ConfigureDevelopment(...) { ...
  }
}
```

50

Environment-Specific Configuration

```
public class Startup {
  public void ConfigureServices(...) { ...
  }
  public void ConfigureDevelopmentServices(...) { ...
  }
  public void Configure(IApplicationBuilder app,
                        IHostingEnvironment env) { ...
    app.UseExceptionHandler("/Home/Error");
    app.UseStaticFiles();
    app.UseMvc(routes => {
      routes.MapRoute(name: "default”,
        template: "{controller=Home}/{action=Index}/{id?}");
    });
  }
  public void ConfigureDevelopment(...){ ...
  }
}
```

51

Environment-Specific Configuration

```
public class Startup {
  public void ConfigureServices(...) { ...
  }
  public void ConfigureDevelopmentServices(...) { ...
  }
  public void Configure(IApplicationBuilder app,
                        IHostingEnvironment env) { ...
  }
  public void ConfigureDevelopment(IApplicationBuilder app,
                                   IHostingEnvironment env){
    app.UseDeveloperExceptionPage();
    app.UseStatusCodePages();
    app.UseBrowserLink();
    app.UseStaticFiles();
    app.UseMvcWithDefaultRoute(); }
}
```

52

CONTROLLERS

53

Startup Class

```
namespace HelloWorld {
  public class Startup {
     public void ConfigureServices (IServiceCollection services) {
       services.AddMvc();
       services.AddMemoryCache();
       services.AddSession();
     }
     public void Configure
             (IApplicationBuilder app, IHostingEnvironment env) {
       app.UseStatusCodePages();
       app.UseDeveloperExceptionPage();
       app.UseStaticFiles();
       app.UseSession();
       app.UseMvcWithDefaultRoute();
     }
  }
}
```

Required for session management

54

Startup Class

```
namespace HelloWorld {
  public class Startup {
     public void ConfigureServices (IServiceCollection services) {
       services.AddMvc();
       services.AddMemoryCache();
       services.AddSession();
     }
     public void Configure
             (IApplicationBuilder app, IHostingEnvironment env) {
       app.UseStatusCodePages();
       app.UseDeveloperExceptionPage();
       app.UseStaticFiles();
       app.UseSession();
       app.UseMvcWithDefaultRoute();
     }
  }
}
```

Adding session data to requests and cookies to responses

55

Getting Input Parameters

- Context Objects
- Action Method Parameters
- Model Binding

56

Getting Input Parameters

- Context Objects
  - Request
  - Response
  - HttpContext
  - RouteData
  - ModelState
  - User
- Action Method Parameters
- Model Binding

57

Common HttpRequest Properties

- Path
- QueryString
- Headers
- Body
- Form
- Cookies

58

API CONTROLLERS

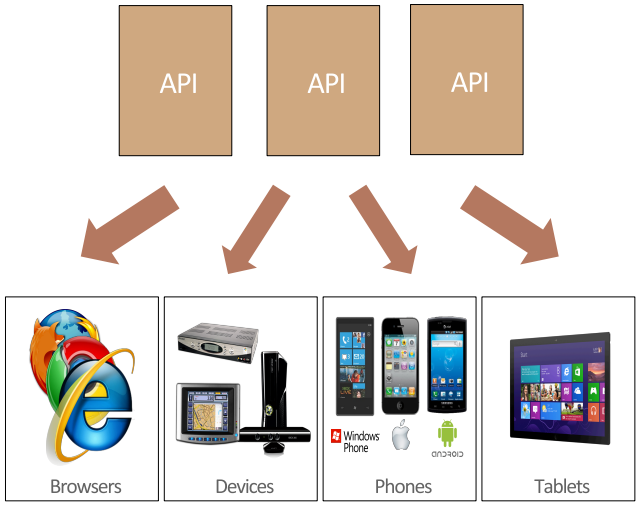
59

API Controllers

- Replacement for Web API
- MVC Controllers that return data

60

API Controllers connect to all HTTP aware clients



Differences from MVC

- Dispatch to action based on HTTP verb
  - MVC: based on action name
- Raw data objects in response
  - Converted to e.g. JSON, XML
- Content type negotiation

62

Modeling Binding vs Formatters

- **Model Binding** (MVC)
  - Value provider extracts name-value pairs from request
  - Model binder builds input (view) model
  - MVC model binder reads from URI *& body*
  - Web API model binder reads from URI *only*
- **Formatters** (API Controllers)
  - Serializers with metadata (content type)
  - Decodes request body, writes request response
  - XML, JSON supported out of the box

63

Action Parameter Binding

- Posted automatically from input data
- Simple type: using model binding
- Complex type (model): using formatters
- Override using [FromUrl], [FromBody], [ModelBinder]

64

Attributes for Parameter Binding

- `[ModelBinder]`
  - Use model binding, specify custom binder
  - Model binders defined for collections
  - Implied for simple types
- `[FromUrl]`
  - Model binding only from URI
- `[FromBody]`
  - Read body, use *formatter*
  - Only one argument
  - Implied for complex type

65

Validation

- Run on data from every request
- Errors accumulated in the ModelState
- Check ModelState.IsValid
- Uses DataAnnotations or custom validation logic

66

Example

- Room Reservation System

67

Models for Example

```
namespace ApiControllers.Models {
  public class Reservation {
    public int ReservationId { get; set; }
    public string ClientName { get; set; }
    public string Location { get; set; }
  }
}
```

68

Models for Example

```
public interface IRepository {

  IEnumerable<Reservation> Reservations { get; }

  Reservation this[int id] { get; }

  Reservation AddReservation
                  (Reservation reservation);

  Reservation UpdateReservation
                  (Reservation reservation);

  void DeleteReservation(int id);
}
```

69

Models for Example

```
public class MemoryRepository : IRepository {
  private Dictionary<int, Reservation> items;

  public MemoryRepository() {
    items = new Dictionary<int, Reservation>();
    new List<Reservation> {
      new Reservation { ClientName = "Alice",
                        Location = "Board Room" },
      new Reservation { ClientName = "Bob",
                        Location = "Lecture Hall" },
      new Reservation { ClientName = "Joe",
                        Location = "Meeting Room 1" }
    }.ForEach(r => AddReservation(r));
  }
```

70

Models for Example

```
public class MemoryRepository : IRepository {
  private Dictionary<int, Reservation> items;

  public Reservation this[int id] =>
    items.ContainsKey(id) ? items[id] : null;

  public IEnumerable<Reservation> Reservations =>
    items.Values;

  public Reservation(AddReservation reservation) {...}

  public void DeleteReservation(int id) =>
    items.Remove(id);

}
```

71

Controller

```
public class HomeController : Controller {
  private IRepository repository { get; set; }
  public HomeController(IRepository repo) =>
    repository = repo;

  public ViewResult Index() =>
    View(repository.Reservations);

  [HttpPost] public IActionResult
  AddReservation(Reservation reservation) {
    repository.AddReservation(reservation);
    return RedirectToAction("Index");
  }
}
```

72

View

```
@model IEnumerable<Reservation>
@{  Layout = "_Layout"; }
...
<table>
  <thead><tr><th>ID</th>
       <th>Client</th><th>Location</th></tr></thead>
  <tbody>
    @foreach (var r in Model) {
      <tr> <td>@r.ReservationId</td>
           <td>@r.ClientName</td>
           <td>@r.Location</td>
      </tr>
    }
  </tbody>
</table>
```

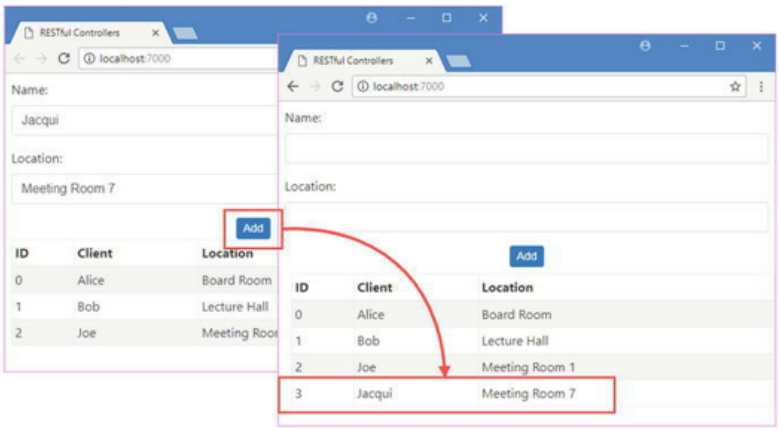
73

Startup Class

```
public class Startup {
   public void ConfigureServices
               (IServiceCollection services) {
     services.AddSingleton<IRepostory, MemoryRepository>();
     services.AddMvc();
   }
   public void Configure
               (IApplicationBuilder app,
                IHostingEnvironment env) {
     app.UseStatusCodePages();
     app.UseDeveloperExceptionPage();
     app.UseStaticFiles();
     app.UseMvcWithDefaultRoute();
   }
}
```

74

Example Interaction



75

RESTFUL EXAMPLE

76

Verb	URL	Description	Payloads
GET	/api/reservations	This combination retrieves all the objects.	This response contains the complete collection of Reservation objects.
GET	/api/reservations/1	This combination retrieves the reservation object whose ReservationId is 1.	The response contains the specified Reservation object.
POST	/api/reservation	This combination creates a new Reservation.	The request contains the values for the other properties required to create a Reservation object. The response contains the object that was stored, ensuring that the client receives the saved data.
PUT	/api/reservation	This combination replaces an existing Reservation.	The request contains the values required to change the properties of the specified Reservation. The response contains the object that was stored, ensuring that the client receives the saved data.
PATCH	/api/reservation/1	This combination modifies the existing Reservation object whose ReservationId is 1.	This request contains a set of modifications that should be applied to the specified Reservation object. This response is a confirmation that the changes have been applied.
DELETE	/api/reservation/1	This combination deletes the Reservation object whose ReservationId is 1.	There is no payload in the request or response.

77

API Controller

```
[Route("api/[controller]")]
public class ReservationController : Controller {

  private IRepository repository;

  public ReservationController(IRepository repo) =>
    repository = repo;

  [HttpGet]
  public IEnumerable<Reservation> Get() =>
    repository.Reservations;

  [HttpGet("{id}")]
  public Reservation Get(int id) =>
    repository[id];
```

78

API Controller

```
[Route("api/[controller]")]
public class ReservationController : Controller {

  [HttpPost]
  public Reservation Post([FromBody] Reservation res) =>
    repository.AddReservation(
      new Reservation { ClientName = res.ClientName,
                        Location = res.Location
                      });

  [HttpPut]
  public Reservation Put([FromBody] Reservation res) =>
    repository.UpdateReservation(res);

 [HttpDelete("{id}")] public void Delete(int id) =>
    repository.DeleteReservation(id);
```

79

API Controller

```
[Route("api/[controller]")]
public class ReservationController : Controller {

  [HttpPatch("{id}")]
  public StatusCodeResult Patch(int id,
    [FromBody] JsonPatchDocument<Reservation> patch) {
    Reservation res = Get(id);
    if (res != null) {
      patch.ApplyTo(res);
      return Ok();
    }
    return NotFound();
  }
```

80

Defining The Controller

- Context path:

```
[Route("api/[controller]")]
public class ReservationController : Controller {
```

- HTTP method:

```
[HttpGet]
public IEnumerable<Reservation> Get() =>
  repository.Reservations;
```

- Extending the route:

```
[HttpGet("{id}")]
public Reservation Get(int id) =>
  repository[id];
```

81

Customizing The Result

```
[HttpGet("{id}")]
public IActionResult Get(int id) {
  Reservation result = repository[id];
  if (result == null) {
    return NotFound();
  } else {
    return Ok(result);
  }
}
```

82

Testing with PowerShell

```
Invoke-RestMethod
  http://localhost:7000/api/reservation
  -Method GET

Invoke-RestMethod
  http://localhost:7000/api/reservation/1
  -Method GET

Invoke-RestMethod
  http://localhost:7000/api/reservation
  -Method POST
  -Body (@{clientName="Anne"; location="Meeting Room
4"} | ConvertTo Json)
  -ContentType "application/json"
```

83

Content Negotiation

- Default: application/json
  - Special case for strings: text/plain
- Requesting XML:

```
Invoke-WebRequest
  http://localhost:7000/api/…
  -Headers @{Accept="application/xml"}
 | select @{n='Content-Type';e={ $_.Headers."Content-Type" }},
          Content
```

84

Content Negotiation

- Enabling XML:

```
namespace HelloWorld {
  public class Startup {
    public void ConfigureServices
                    (IServiceCollection services) {
      services.AddSingleton<Irepostory, MemoryRepository>();
      services.AddMvc()
              .AddXmlDataContractSerializerFormatters();
    }
    ...
  }
}
```

85

Content Negotiation

- Specifying the content type:

```
[HttpGet("{id}")]
[Produces("application/json")]
public Reservation Get(int id) =>
  repository[id];
```

86

Content Negotiation

- Specifying the content type:

```
[HttpGet("{id}/{format?}")]
[FormatFilter]
[Produces("application/json",
          "application/xml")]
public Reservation Get(int id) =>
  repository[id];
```

87

Content Negotiation

- Enabling XML:

```
namespace HelloWorld {
  public class Startup {
    public void ConfigureServices
                   (IServiceCollection services) {
      ...
      services.AddMvc()
              .AddXmlDataContractSerializerFormatters();
              .AddMvcOptions(opts => {
                opts.FormatterMappings.
                   .setMediaTypeMappingForFormat("xml",
                      new MediaTypeHeaderValue("application/xml"));
                });
    }
    ...
  }
}
```

88

MICROSERVICES

89

Microservices

- Architect server application as set of small services
  - Each service in its own process
  - Communicating via HTTP and Websockets
- Bounded Context per service
- Each service deployed independently
- Per service domain data model and domain logic

90

Examples of Microservices

- Protocol gateways
- User profiles
- Shopping carts
- Inventory processing
- Purchase subsystem
- Payment processing
- Queues
- Caches

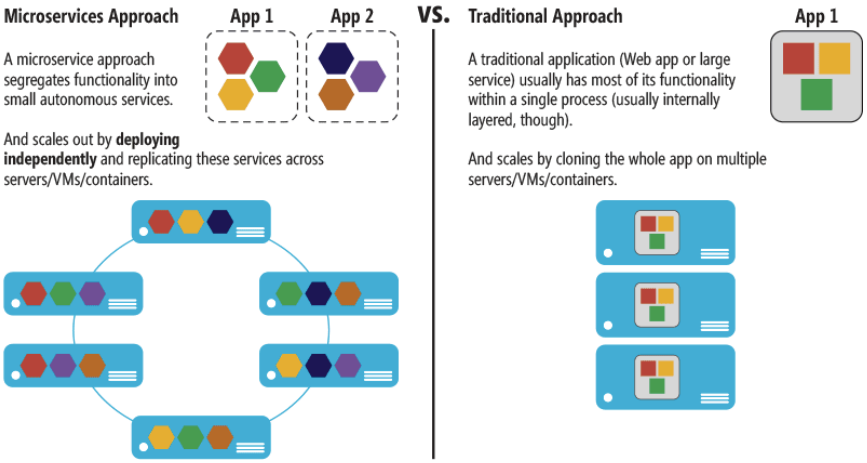
91

Advantages of Microservices

- Agility
  - Superior maintainability
  - Granular release planning
- Independent scaling out
- Continuous integration and development
  - Accelerates delivery of new functionality

92

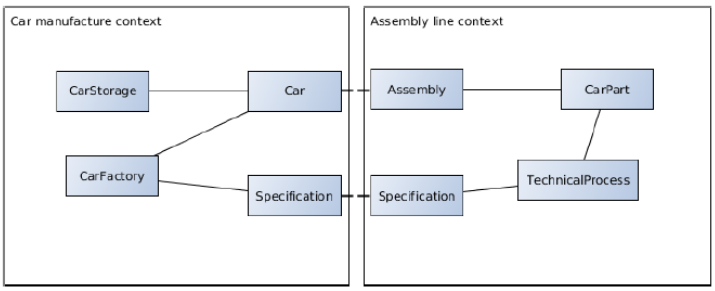
Microservices vs Traditional Server



93

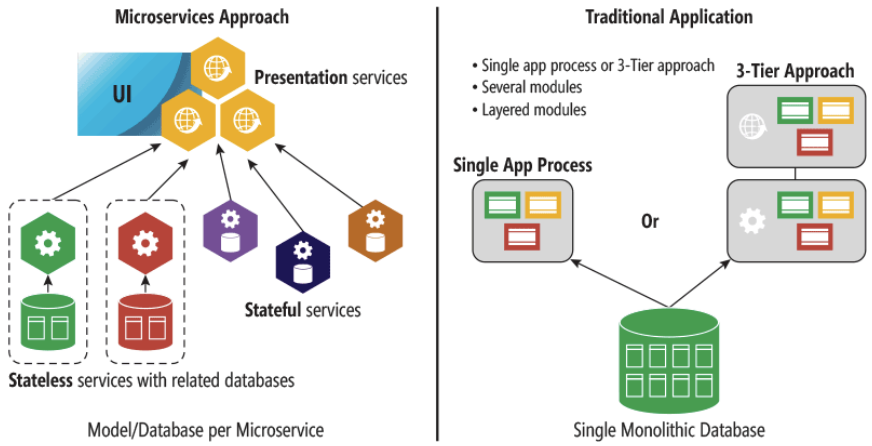
Data Sovereignity

- Each microservice owns its own domain data and logic
- Cf bounded context (domain-driven design)



94

Data Sovereignity



95

Stateless vs Stateful Microservices

- Stateless
  - External DB (SQL Database, Document DB)
  - Problem: Latency
  - Problem: Complexity (caching etc)
- Stateful
  - Local state for faster access
  - Problem: scaling out
  - Data replication
  - Data partitioning

  } Service Fabric

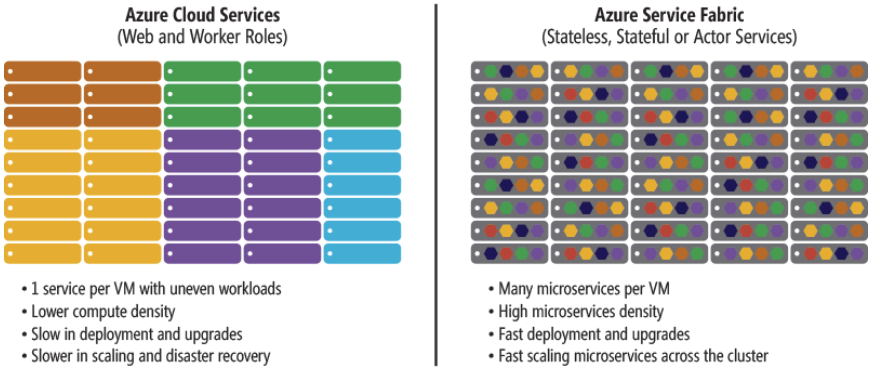
96

Microservices in Azure

- Cloud Services
  - Web Role & Worker Role
  - One instance per VM
- Webjobs
  - One instance per VM
- Azure Service Fabric
  - Self-host HTTP Listener
  - Hundreds of instances per VM

97

Cloud Services vs Service Fabric



98

Azure Service Fabric

- Infrastructure for scaling out microservices
- Originally developed for building out SQL Server to Azure SQL Database

- Reliable Actor APIs
- Reliable Service APIs

99

Reliable Actor API vs Reliable Service API

Reliable Actor API

- Your scenario involves many small independent units/objects of state and logic (live Internet of Things objects or gaming back-end scenarios are great examples)

Reliable Service API

- You need to maintain logic and queries across multiple entity types and components

100

Reliable Actor API vs Reliable Service API

Reliable Actor API

- You work with a massive amount of single-threaded objects while still being able to scale and maintain consistency
- You want the framework to manage the concurrency and granularity of state

Reliable Service API

- You use reliable collections (like .NET reliable Dictionary and Queue) to store and manage your state/entities
- You want to control the granularity and concurrency of your state

101

Reliable Actor API vs Reliable Service API

Reliable Actor API

- You want Service Fabric to manage the communication implementation for you
- You want the framework to manage the concurrency and granularity of state

Reliable Service API

- You want to decide on, manage and implement the communication protocols (Web API, WebSockets, Windows Communication Foundation and so on)
- You want to control the granularity and concurrency of your state

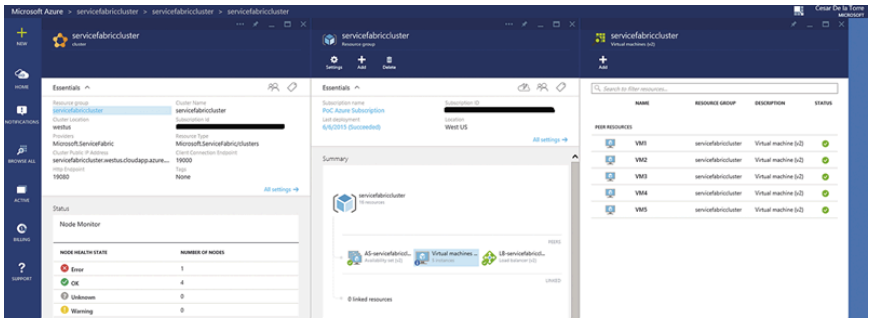
102

Structure of a Stateless Service

```
using Microsoft.ServiceFabric.Services;
namespace MyApp.MyStatelessService {
  public class MyStatelessService : StatelessService {
    protected override async Task RunAsync
                 (CancellationToken cancellationToken) {
      int iterations = 0;
      while (!cancellationToken.IsCancellationRequested) {
         ServiceEventSource.Current.ServiceMessage(this,
                                              "Working-{0}",
                                              iterations++);
         // Service logic here
         await Task.Delay(TimeSpan.FromSeconds(1),
                          cancellationToken);
      }
    }
  }
}
```

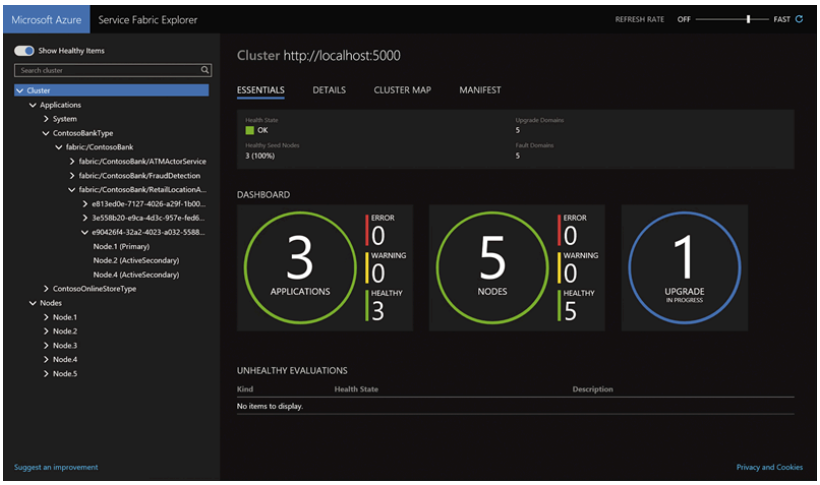
103

Service Fabric Cluster in Azure Portal



104

Service Fabric Explorer



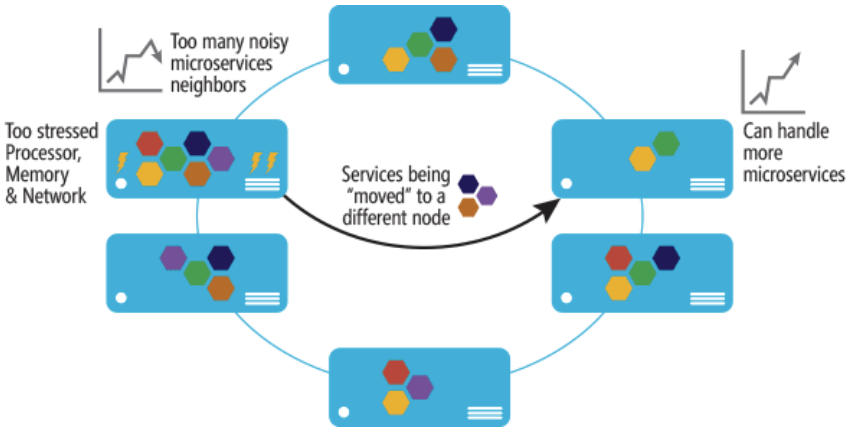
105

Service Fabric Support

- Scaling out Stateless Services
- Automatic Resource Balancing
  - Redeployment of microservices
- Built-in Failover and Replication
  - Because of multiple instances
- Placement Constraints
  - E.g. Web tier and business tier separate
- Load Balancing of Requests
- Health

106

Automatic Resource Balancing



107

Stateful Microservice in Service Fabric

- Colocate microservice and data
- Replicate data in each partition
  - Replicated microservices with data
  - Hot secondary backups for failover
  - Reliable Dictionary & Reliable Queue
  - Similar to Azure SQL Database

108

Reliable Actors

- Asynchronous, single-threaded programming model
- Actor = state + computation
- Example: live IoT objects (e.g. vehicles)
- Treated as in-memory objects
  - Persisted on local disk
  - Replicated throughout cluster

109

Reliable Actor

```
// Object instances of this Actor class will be running
// transparently in the service back end.

public class VehicleActor : Actor<Vehicle>,
                            IVehicleActor {

  public void UpdateGpsPosition(GpsCoordinates coord) {
    // Update coordinates and trigger any data
    // processing through the State property
    // on the base class Actor<TState>.
    this.State.Position = coord;
  }

}
```

110

Reliable Actor Client

```
// Client .NET code
ActorId actorId = ActorId.NewId();
string applicationName =
   "fabric:/IoTVehiclesActorApp";

IVehicleActor vehicleActorProxy =
  ActorProxy.Create<IVehicleActor>
            (actorId, applicationName);

vehicleActorProxy
  .UpdateGpsPosition(
       new GpsCoordinates(40.748440,
                          -73.984559))
```

111

Conclusions

- ASP.NET Core
  - Small footprint
  - Self-hosting
  - Many microservices on a single node
- Service Fabric
  - Infrastructure for managing distributed microservices
  - Reliable Services
  - Reliable Actors for IoT apps

112