Anatomy of ASP.NET Core

Building Blocks

Application Entry Point

STARTUP.CS

- Load configuration (from files, environment variables, etc)
- Populate SERVICES (Direct Injection container: including MVC, Entity Framework, Identity, Logging, Session, Caching, etc)
- Define PIPELINE (order of middleware to process requests)

Primary Components

MODELS
(standalone objects and database tables mapped to objects using EF)

VIEWS
(csHTML using Razor to define pages sent to response)

CONTROLLERS
(manage which views to deliver and which models to use)

VIEW MODELS
(combinations of multiple DB and/or standalone objects for populating specific views)

LOGIC
(interfaces and business logic implementations place in SERVICES, pass to CONTROLLERS by DI)

Assistive Compenents

LAYOUTS
(site template views, as Master Pages in Web Forms)

PARTIAL VIEWS
(useful for creating simple reusable view markup)

VIEW COMPONENTS
(useful for creating fully standalone components such as shopping carts, dynamic navs, login panels, etc)

AREAS
(folder structures for organising MVC into discrete chunks, e.g. client and admin)

TAG HELPERS
(HTML attribute elements that map to server-side functionality, many built-in, can create own)

FORMATTERS
(SERVICES that format data for response output, e.g. JSON (built-in), plain-text (built-in), XML, etc)

FILTERS
(runs code before/after particular stage of pipeline: authorization, resource, action, exception, and result)

Request Lifecycle

REQUEST
received by Web Server (IIS)

↓

REQUEST
passed to ASP.NET Core Web Server (Kestrel)

↓

PIPELINE
selects which middleware processes request

↓

CONTROLLER
processes all components including (where required):

- Logic (from DI services)
- Model
- View

↓

RESPONSE
sent to client

J .Madle - 16 August 2016

Naming Conventions in ASP.NET Core

MVC Naming Conventions

- **URL to Controller Routing** — Where URL is /{Path} then Controller Class name is {Path}Controller
- **Areas** — Partitions large applications into smaller MVC groupings within the root “Areas” folder, for example Areas/Client and Areas/Admin. Views are located in the following order:
  /Areas/<Area-Name>/Views/<Controller-Name>/<Action-Name>.cshtml
  /Areas/<Area-Name>/Views/Shared/<Action-Name>.cshtml
  /Views/Shared/<Action-Name>.cshtml

General Programming Conventions

- **Interfaces** — Defines required implementation of classes deriving from the interface; camel-case naming with “I” prefix, for example IProductInterface
- **Repositories** — A class representing a collection of data objects, abstracting the data context away from the actual objects; camel-case naming with “Repository” suffix, for example “ProductsRepository”
- **Factories** — A class that returns new instances of objects, abstracting the creation process away from the primary program flow; camel-case naming with “Factory” suffix, for example “ProductFactory”
- **ContextAdapter** — A class representing a fragment of a larger object, avoids passing the entire larger object into other classes; camel-case naming with “ContextAdapter” suffix, for example “RequestContextAdaptery”