# Getting Acquainted with ASP.NET Core

## 2.7. Inspecting the ASPNET Core API Project Code Structure

* How to bypass requests

|  |  |
| --- | --- |
| Template pipeline | By pass pipeline |
|  |  |

## 2.8. Demo: Working with Different Environments

In ASP.NET Core, the terms:

* "**build**" refers to the process of **compiling** source code, dependencies, and resources into an executable.
  + DEBUG Build: Used for development. Includes debugging symbols and is not optimized for performance.
  + RELEASE Build: Used for production. Excludes debugging symbols and optimizes performance.
* **“environment”** defines how the application should **behave** based on the runtime setting. Common environments include:
  + Development → Used for local development (detailed error messages, hot reload).
  + Staging → Intermediate stage before production (for testing).
  + Production → Live environment with optimized performance and security.

|  |
| --- |
| launchSettings.json |
|  |

# Creating the API and Returning Resources

## 3.2. Clarifying the MVC Pattern

Model-View-Controller:

* Improves separation of concerns
* Improves testability
* Promotes reuse

Parts

* Model: application data, business rules.
* View: Representation of data, html
* Controller: receive input from user and interact with view.

## Demo: Registering API Services on the Container

Other Adds related with MVC

A screenshot of a computer

AI-generated content may be incorrect.

## Returning Resources

|  |  |
| --- | --- |
| Adding new controllers | ApiController Attribute |
| ControllerBase: just enough to MC  Controller: generic, with view things | Include routing, return 400 bad request, problems details |

## Leaning About Routing

|  |  |
| --- | --- |
| Old way | New way |
| UseRouting(): maks the position in the middleware pipeline where routing decision must be made.  UseEndpoints():maks the position in the middleware pipeline where endpoint must be executed.  Map Endpoints: convetion ou attribute base   * API uses attribute based | New way |
| Attribute-based Routing | [Route] template |
| Use attributes at controller and action level:  [Routel, [HttpGet]  Combined with a URI template, requests are matched to controller actions |  |

## Test config

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | **1. Abra o Visual Studio** e carregue sua solução (MyApi).  **2. No Solution Explorer**, clique com o botão direito na **Solução (MyApi.sln)** → **Adicionar** → **Novo Projeto**.  3. Na janela **Criar um Novo Projeto**:  3.1. Selecione **xUnit Test Project** (ou **Projeto de Testes xUnit** em português).  3.2. Ajuste o local da pasta   * 1. Clique em **Avançar**.   3.4. Nomeie o projeto como **MyApi.Tests**.  3.5. Clique em **Criar**.  4. Agora o projeto de testes foi criado e aparecerá no **Solution Explorer**. | Para ver o Test Explorer:  Barra de tarefas, View > Test Explorer |

# d