1. C#和.NET的关系
2. .NET：通用开发平台，包含.NET Framework（有CLR、FCL等）、Mono、.NET Core等技术框架，.NET拥有CLI（公共语言基础结构），其中包括与语言无关的运行时和语言互操作性
3. 面向.NET的编程语言

C#：高级、强大、类型安全且面向对象，运行在.NET FW/Core CLR上

F#：跨平台、函数优先

VB.NET：简单易上手

1. 总结：.NET是一个支持多语言开发的平台，C#是编程语言，C#可运行在.NET平台上。

二、.NET平台

1. .NET Framework

2000年 .NET 1.0发布，.NET Framework需要借助 Mono才可以跨平台

2. Mono

Xamarin公司开发的开源项目，支持Windows、Linux、maxOS、iOS、Android

1. .NET Standard

.NET API规范，.NET标准，同时也是一个目标框架（只有类库可用）。作用是提高.NET生态的一致性，.NET FW和.NET Core都是基于.NET Standard的实现

1. .NET Core

特点：核心是创新、开源（MIT、Apache2协议）、跨平台，具有统一性和兼容性，使用命令行工具，现代化，.NET基金会支持

1. .NET Core和.NET Framework

.NET Core实现部分.NET Framework的功能（大多是不含用户界面的部分），如JIT（.NET Core是RyuJIT）、GC、类型

1. .NET Core和ASP.NET Core

主从关系，ASP.NET的运行是.NET Core

7. .NET 和 Java

相同点：即时编译（JIT）的动态语言

不同点：

|  |  |  |
| --- | --- | --- |
| **平台** | **中间语言** | **运行时** |
| .NET | IL | JRE |
| Java | 字节码 | CLR |

三、.NET5

1. 开源，和GitHub的社区向定位。

2. 跨平台的实现。

3. 支持举足轻重（原词leveraging直译杠杆作用）的特定平台功能，例如winform和wpf，以及xamarin在每个原生平台上的原生适配。

4. 高性能。

5. 并行安装（在这里指的应该是可以和旧版本共存）。

6. 小的项目文件（SDK风格）。

7. 强悍的命令行接口（CLI）。

8. VS，VS for Mac，VSCode集成

四、ASP.NET Core概述

1. ASP.NET Core是一个跨平台、高性能、模块化、可扩展、轻量级的开源框架。是基于ASP.NET 4.x系列重新设计的，所以起初称为ASP.NET 5，后来被重命名为ASP.NET Core

2.ASP.NET Core的特性：

a) 跨平台。可托管在IIS、Apache、Docker、HTTP.sys、Nginx，另外使用自带的Kestrel可以托管在进程中（自托管）；可以部署在Linux，maxOS等平台或使用对应的IDE开发

b) Web API和MVC统一的技术栈：Controller从相同的基类继承，并返回IActionResult，IActionResult的常用实现如ViewResult（MVC）、JsonResult（Web API）

c) 原生依赖注入支持（DI）

d) 更强的可测试性。单元测试、集成测试

e) 轻量、高性能的模块中间件

f) 开源、社区支持

g) 原生GRPC支持

五、ASP.NET Core项目启动流程

1. ASP.NET Core项目文件 .csproj

以前：包含文件或文件夹的引用；需要卸载项目后才可以编辑项目文件

现在：不包含文件和文件夹的引用，由文件系统确定哪些文件文件夹属于项目，即项目根目录下的所有文件和文件夹都属于项目；不需要卸载项目文件即可编辑，且可以双击项目打开项目文件

2. ASP.NET Core项目入口 Program.Main()

ASP.NET Core应用程序最初作为控制台应用程序启动，调用CreateHostBuilder方法，返回一个实现IHostBuilder的对象；然后调用Build方法，将ASP.NET Core应用程序生成并托管到服务器上；然后调用Run方法，ASP.NET Core应用程序开始侦听传入的HTTP请求。

CreateDefaultBuilder方法：将Kestrel作为服务器并启用IIS集成；从各种配置源中加载

配置信息；配置日志记录。

文本

描述已自动生成

Startup类，配置应用程序所需的服务、配置应用程序的请求处理管道

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

3.进程内托管模型

ASP.NET Core应用程序可托管在进程内（InProcess）或进程外（OutOfProcess）。通过AspNetCoreHostingModel节点配置，默认是进程内。

a) 进程内托管。即在IIS（w3wp.exe / iisexpress）进程中托管

b) 进程外托管。将Web请求转发到运行Kestrel服务器的后端ASP.NET Core应用程序

什么是Kestrel服务器？它是ASP.NET Core的跨平台服务器，默认作为内部服务器包含在ASP.NET Core中，也可以用作面向互联网的Web服务器。用于托管应用程序的进程是dotnet.exe，使用.NET Core CLI运行.NET Core应用程序时，使用Kestrel作为Web服务器（即便AspNetCoreHostingModel配置为InProcess也一样）。

什么是CLI？它是.NET Core跨平台的命令行工具。CLI命令可以实现：

1. 创建新项目、配置文件、解决方案等
2. 还原依赖项和工具包
3. 生成项目、运行项目

4.进程外托管模型

AspNetCoreHostingModel配置为OutOfProcess，或者CLI运行程序（启动Kestrel服务器，进程外托管）。

内部Web服务器是Kestrel时，外部Web服务器可以是IIS、Nginx、Apache。

当应用程序配置为进程外托管时，IIS Express充当反向代理服务器，即IIS Express接收Http请求并转发给Kestrel处理，Kestrel将响应转发给IIS Express 再转发到浏览器。

六、ASP.NET Core中的配置文件

1. 启动配置信息：launchsetting.json。

a) 当使用VS或CLI运行项目时，使用此配置文件。

b) 仅用于本地，不需要发布到生产环境

launchsetting.json中默认有两个配置信息IIS Express、MockSchoolManagement（项目名称）。对于commandName，vs默认使用IIS Express、CLI默认使用Project。commandName与AspNetCoreHostingModel的配置组合，结果如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| commandName | AspNetCoreHostingModel | Internal Web Server | External Web Server |
| Project | 忽略 | Kestrel | Kestrel |
| IIS Express | InProcess | IIS Express | IIS Express |
| IIS Express | OutOfProcess | Kestrel | IIS Express |
| IIS | InProcess | IIS | IIS |
| IIS | OutOfProcess | Kestrel | IIS |

2.通过图形界面设置launchsetting.json

a) 如何打开界面？右击项目文件->属性->调试。

b) 关于环境变量ASPNETCORE\_ENVIRONMENT，默认值时Development，在Startup类中，可以通过IHostingEnvironment获取环境变量的配置，从而对测试环境或生产环境做不同的处理。

3. ASP.NET Core中的应用程序配置

应用程序配置有：

1. Appsettings.json或appsettings.{Environment}.json
2. UserSecrets（用户机密）
3. Environment Variable（环境变量）
4. Command-line arguments（命令行参数）

访问应用程序配置。

在Startup类中访问应用程序配置，需要注入IConfiguration服务（由ASP.NET框架提供）。注意：key不区分大小写

1. appsettings.json。 如果访问得到的配置value为乱码，则需要将appsettings.json的文件格式修改为urf-8。appsettings.{Environment}.json的优先级高于appsettings.json
2. 用户机密。右击项目->管理用户机密->在secrets.json中添加需要加密的配置信息。secrets.json配置信息后，会在csproj文件中增加一个UserSecretsId的节点，标识这个secrets.json文件的ID，secrets.json不在项目中
3. 环境变量。在launchsettings.json文件中，environmentVariables节点下配置的keyValue。
4. 命令行参数。在CLI窗口，输入dotnet run MyKey=”value from command line”

以上四种方式配置的相同的Key，优先级从上往下递增。原因是Host的CreateDefaultBuilder方法在应用程序启动时按照特定顺序读取配置源。

七、ASP.NET Core中间件及其原理

1. 什么是中间件

在ASP.NET Core中，中间件是一个可以处理HTTP请求或响应的软件管道，比如：验证、处理错误、提供静态文件。

2. 中间件的工作原理

① 中间件处理请求后，可以传递给下一个中间件。

② 中间件在Startup类中的Configure方法中配置。

③ 终端中间件：处理请求而不调用下一个中间件，使管道短路。

④ 中间件是按照添加到管道中的顺序执行的。

⑤ 大型团队中，中间件一般以NuGet包提供。

3. 配置ASP.NET Core请求处理管道

① Run方法是将一个终端中间件添加到管道。

② 注册User()中间件，可以调用下一个中间件。

八、