## ASP.NET Core 3 学习

**一．StarUp 类 配置**

1. StarUp 普通的类 约束实现
2. **ConfiggutrServices** 用于依赖注入配置

DI 依赖注入 （Dependency Injection）

DI优点： 解耦，没有强依赖

-利于单元测试

-不需要了解具体的服务类

-也不需要管理服务类的生命周期

IOC 容器 （Inversion ofControl）控制反转

类型在IOC容器中注册 services 服务

请求实例 实例的生命周期

**Transient**：注册服务在每一次请求生成一个新的实例

**Scoped**：web请求，根据每一次web请求生成一个新的实例，web请求结束了，生命周期结束.

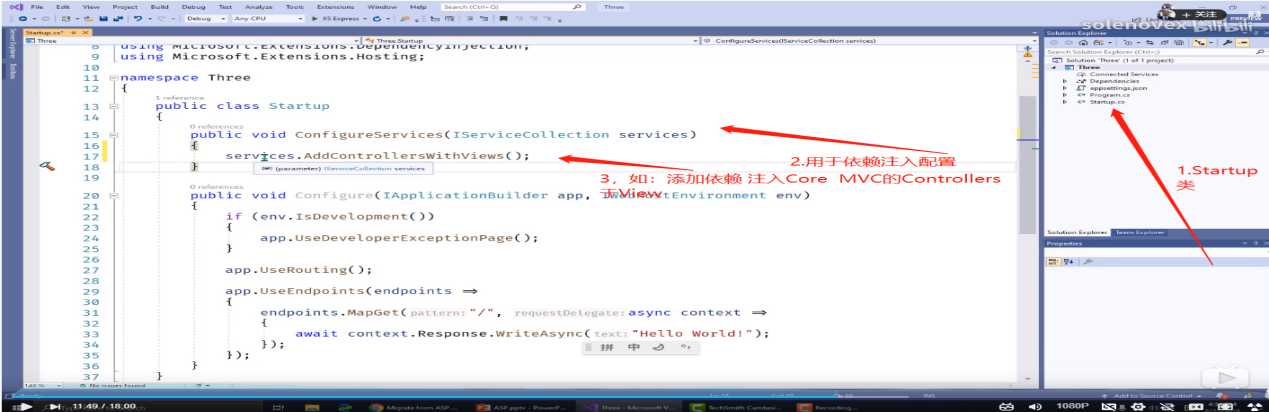
**Singleton**：服务实例创建，以后请求这个服务的时候会一直用这个实例，只有等这个应用程序停止的时候生命周期才结束.

语法：Services.Add**Singleton<I定义接口类,具体实现类>();**

全部功能 Services.AddMvc()

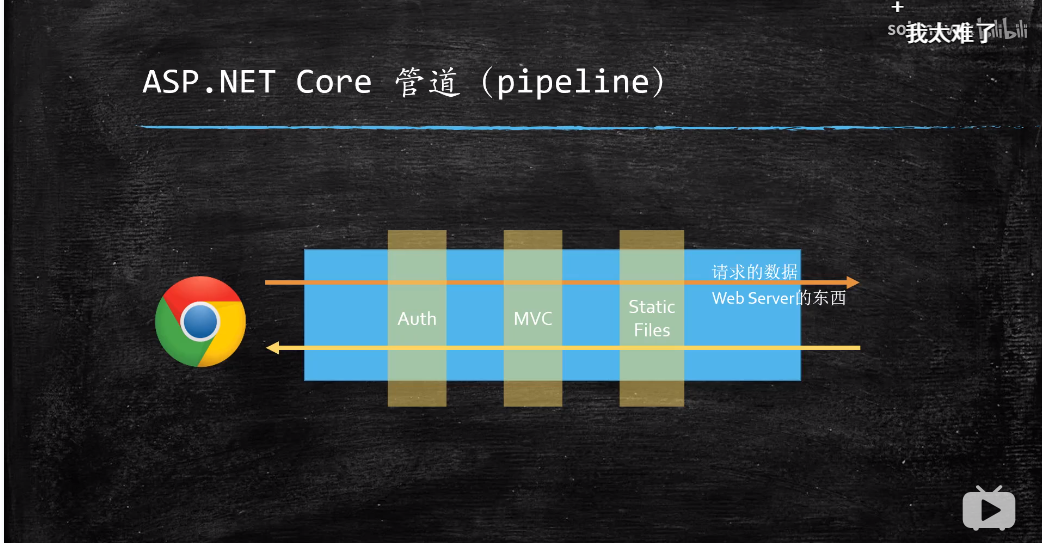
MVC 注入 Services.AddControllersWithviews()

WEB API Services.AddControllers()



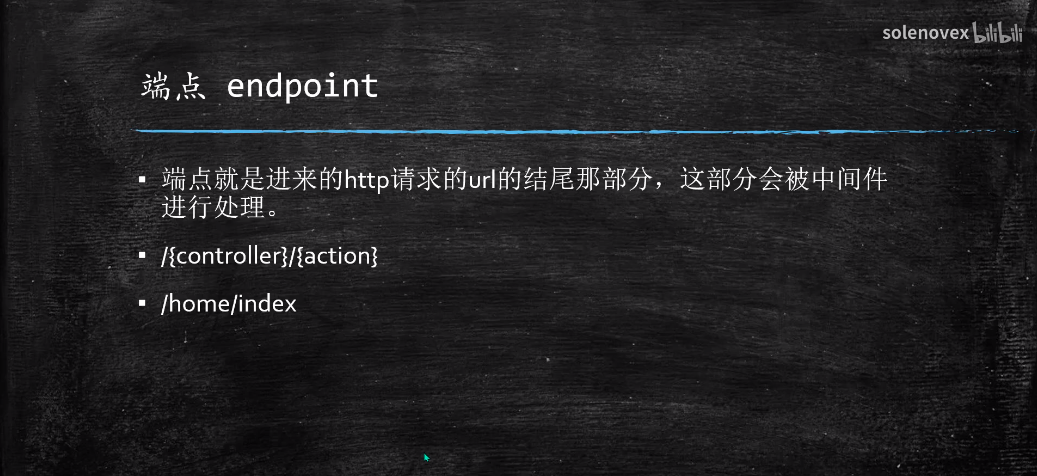
1. **ASP.NET Core 管道 （pipeline） Configure**

中间件





端点 endpoint UseEndpoints



例如：路由中间件 UseRouting



静态文件使用中间件 App.UseStaticFiles() 文件夹 wwwroot下存放静态文件（css 图片 图标 等）

1. 包管理

服务端 Nuget

前端 NPM

添加配置文件 npn

1. **使用 MVC 相关技术**



Controlle 控制器

[Http Get ] 默认请求方式

[Http Post]

View 视图

Tag Helper 在view文件下添加 \_ViewImports 文件 在文件中 添加Tag Helper

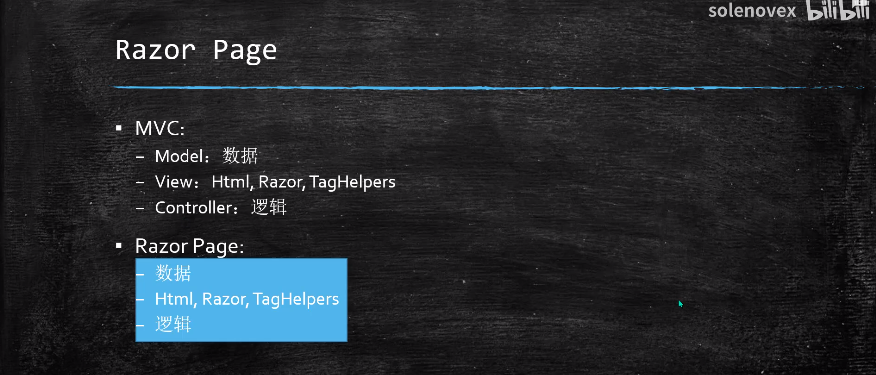
语法：@addTagHelper \* , Microsoft.AspNetCore.Mvc.TagHelpers”

Model 实体

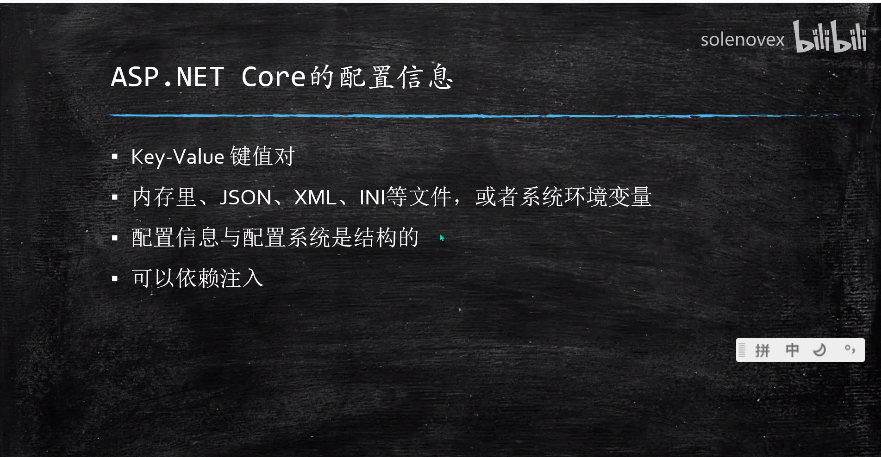
View Component



Razor Page



1. **ASP.NET Core 的配置信息 Settings**



四．SIgnalR 原理

