Atitit 融合战略的艺术

目录

[1.1. Os方向 uwp ms。。 1](#_Toc330)

[1.2. Sdk 方向 。ms netcore 1](#_Toc20048)

[2. Netcore优点 1](#_Toc16622)

[2.1. .NET应用程序可以是跨平台的 1](#_Toc14518)

[2.2. 扩大您类库的覆盖面 2](#_Toc3175)

[2.3. 在任何平台上简单部署 2](#_Toc8805)

[2.4. 云和容器 3](#_Toc9899)

[2.5. java run everythong。 3](#_Toc6854)

[2.6. 军民融合战略 4](#_Toc161)

[3. Ref 4](#_Toc31168)

art 融合战略

## Os方向 uwp ms。。

## Sdk 方向 。ms netcore

# Netcore优点

## ****.NET应用程序可以是跨平台的****

您现有的.NET Framework应用程序可以在其他操作系统上工作。对于希望扩大类库的受众平台，或者希望在分布式应用程序的其他领域使用相同代码的开发人员来说，这是非常好的选择。即使你想用你亮瞎眼的MacBook上开发.NET，而无需双启动到Windows。

并不是所有的框架都被移植到.NET Core，但主要的部分。存在一些API差异。例如，如果您使用了大量的反射，则可能需要重构代码才能使用Core。有关更多信息，请看文章末尾的的两者差异部分。

## ****扩大您类库的覆盖面****

使用.NET Core，您可以使用.NET标准库编写应用程序或库。

然后它可以在许多平台上共享。

同一个库可以在后台服务“本地”或云中运行，也可以在手机，平板电脑或桌面上运行的客户端应用程序中运行。与其为iOS，Android和Windows构建单独的应用程序，您可以构建一个适用于所有平台的应用程序。.NET Core是小型和完美的容器，可以轻松扩展并缩短开发时间。

.NET Core和.NET Standard Library建立了一个通用平台。过去，当新版本的操作系统或新设备出现时，开发人员有责任重新构建新平台的应用程序或库，并分发更新。使用.NET Core，不需要重建和重新分配。只要新的平台支持你所有的依赖库，它就支持你的应用程序。

## ****在任何平台上简单部署****

Microsoft产品往往具有复杂的安装过程。COM组件，注册表项，特殊文件夹，GAC - 都是为了利用Windows的特性而设计的。.NET框架依赖于这些构造，这使得它不适合其他操作系统。

在发布依赖于.NET Framework的应用程序时，安装程 序必须足够聪明才能检测是否安装了错误的.NET Framework版本，并为用户提供正确的方法。大多数现代Windows版本都包含.NET Framework。这使得某些应用程序更容易安装，但是如果应用程序使用默认情况下未安装的功能（如ASP.NET与IIS或WCF组件的集成），则可能会导致复杂的问题。

另一个复杂的问题来自补丁。包括错误修复或安全更新的修补程序可以通过Windows更新或通过Microsoft下载中心分发给客户。您测试应用程序的.NET Framework可能与客户使用的补丁程序不同。当您假定.NET Framework对于所有客户都是一样的时候，通常很难确定在应用程序中导致奇怪行为的原因。

.NET Core的模块化设计意味着您只包含所需的依赖关系。所有这些依赖关系与您的应用程序进入相同的文件夹。部署应用程序与复制文件夹一样简单。这种方法的另一个优点是可以有多个版本并行运行。这个策略对于使所有平台的部署体验保持一致至关重要。

## ****云和容器****

在云系统中，用更少的硬件为更高密度的用户提供服务是非常重要的。应用程序的占位面积越小，密度越高。虚拟机已经在云端普遍存在多年，但是它们有几个问题：

容器通过虚拟化操作系统来解决虚拟机的问题。容器只包含应用程序及其依赖项。文件大小要小很多倍，启动时间以秒为单位，只有应用程序加载到内存中，容器保证在任何主机上工作。

内置于Windows的.NET Framework不能在容器上运行。鉴于容器的明显优势，**.NET Core的设计决定之一就是使其成为模块化**。这意味着你的.NETCore应用程序可以被“发布”，**使得它和它的所有依赖关系在一个地方，这很容易放入容器**。

## java run everythong。

## 军民融合战略

# Ref

为什么选择.NETCore？ - Chaunce - 博客园.html