بسم الله الرحمن الرحیم

تکلیف اول درس برنامه نویسی موبایل

محمد حسین اسناوندی ۹۹۲۰۲۳۰۰۱

۱)معماری MAUI و طریقه اجرای یک برنامه به چه صورت است به طور کامل به زبان خود توضیح دهید.

در معماری(Model-View-ViewModel) MVVM ، برنامه به سه قسمت اصلی تقسیم می‌شودModel :، View، و .ViewModel این معماری به تفکیک مسئولیت‌ها و افزایش قابلیت تست و نگهداری کد کمک می‌کند .

در زمینه‌ی(Multi-platform App UI) .NET MAUI ، معماری MVVM نیز به طور گسترده استفاده می‌شود .NET MAUI .یک فریمورک توسعه اپلیکیشن‌های چندسکویی است که توسط Microsoftایجاد شده است.

در MVVM برایMAUI ، مفاهیم اصلی به شرح زیر هستند:

1)Model

- این قسمت تصمیمات مربوط به چگونگی دسترسی به داده‌ها و پردازش آن‌ها را انجام می‌دهد.

View (2

- مسئول نمایش داده‌ها به کاربر است.

- درMAUI ، ممکن است تعدادی از طراحی‌ها و متنوعیت‌ها بر اساس پلتفرم مقصد) مانند iOS یا (Android وجود داشته باشد.

ViewModel(3

- واسط بین Model و View است.

- مسئول تبدیل داده‌ها از فرمت مناسب برای Model به فرمت مناسب برای View است.

نحوه اجرای برنامه :

1) اجرای فایل mainprogram.cs

2) اجرای App.cs

3) اجرای AppShell

4) اجرای mainpage.xaml که ظاهر برنامه ما در آنجا قرار دارد.

۲)آیا می توانیم برنامه ای بدون App Shell داشته باشیم. آیا می توانیم صفحه ای بدون XAML داشته باشیم، به طور کامل توضیح دهید.

بله، در .NET MAUI می‌توان برنامه‌هایی بدون استفاده از AppShell داشته باشیم و همچنین صفحات بدون استفاده از XAML نیز ایجاد کنیم. دیگر نیازی به استفاده از این

مفاهیم نیست و می‌توانید به جای آن‌ها از ساختارها و زبان‌های دیگری برای تعریف UI استفاده کنید مثل خود سی شارپ.

۳) با ذکر مثال Attached Property و XAML MarkUpExtension را با توضیح دهید.

1. Attached Property:

Attached Property یک ویژگی است که به شیء دیگری متصل شده است، به عبارت دیگر، این ویژگی‌ها به شیءها افزوده می‌شوند. این امکان به توسعه‌دهندگان این قابلیت را می‌دهد که ویژگی‌های خاص خود را به شیءها اضافه کنند بدون اینکه از کلاس اصلی آن شیءها به ارث ببرند.

```
<Grid>
   <Button Content="Click me" Grid.Row="1"/>
</Grid>
```

در اینجا، Grid.Row یک Attached Property است که به Button افزوده شده و مشخص می‌کند که Button در سطر ۱ از Grid قرار گیرد.

2. XAML MarkupExtension:

MarkupExtension در XAML یک مکانیزم است که این امکان را فراهم می‌کند تا مقادیر را در زمان اجرا تولید کنید و به XAML اجازه دهید با مقادیر پیچیده‌تر و پویا برخورد کند. MarkupExtensionها از دستورهای خاصی برای تولید مقادیر در زمان اجرا استفاده می‌کنند.

```
<TextBlock Text="{Binding Name}" />
```

در اینجا، Binding یک MarkupExtension است که در زمان اجرا مقدار Name از ViewModel را به TextBlock منتقل می‌کند. MarkupExtension به توسعه‌دهندگان این امکان را می‌دهد تا مقادیر پویا و پیچیده‌تر را در XAML استفاده کنند.

۴)کامپایل فایل های XAML به طور پیش فرض چه زمانی انجام می شود؟ آیا روشی برای تغییر آن وجود دارد.

کامپایل فایل‌های XAML به طور پیش‌فرض در زمان اجرا (Runtime) انجام می‌شود، به این معنا که فایل‌های XAML به صورت متنی در زمان اجرا تجزیه و تحلیل شده و به شیءهای .NET تبدیل می‌شوند. این اجرای پویا به توسعه‌دهندگان این امکان را می‌دهد که تغییرات در رابط کاربری (UI) خود را بدون نیاز به کامپایل مجدد برنامه اعمال کنند.

اما در برخی موارد، ممکن است تمایل داشته باشید که کامپایل فایل‌های XAML به صورت اولیه در زمان طراحی (Design Time) صورت گیرد تا در حین توسعه و طراحی، مشکلات و اشکالات UI را سریع‌تر شناسایی و اصلاح کنید. برخی ابزارهای توسعه محیطی (IDE) مانند Visual Studioاین امکان را فراهم کرده‌اند.

برای اجرای کامپایل در زمان طراحی، معمولاً از روش‌هایی مانند استفاده از ابزارها و افزونه‌های IDE یا تنظیمات خاص در فایل‌های پروژه (مانند فایل csproj در پروژه‌های .NET) استفاده می‌شود. به عنوان مثال، ممکن است یک خط کد مشابه زیر به فایل csproj اضافه شود:

```
<Project ...>
  <PropertyGroup>
    <IsWpfProject>true</IsWpfProject>
<AlwaysCompileMarkupFilesInSeparateDomain>true</AlwaysCompileMarkupFilesInSeparateDomain>
    <XamlDiagnostics>true</XamlDiagnostics>
  </PropertyGroup>
<Project/>
```

مقادیر مختلف می‌توانند با توجه به نیازهای پروژه تغییر یابند. همچنین، افزونه‌ها و ابزارهای طراحی ممکن است قابلیت اجرای کامپایل در زمان طراحی را اضافه کنند.