Compile time code generation for C#

Edit

> **This documentation and the underlying platform code is a work in progress.**

In previous versions of React Native for Windows, code generation for C# modules was performed using reflection. Since 0.63 we improved this by adding a compile time code generation.

There are several benefits to this approach:

- The cost of analyzing and reflecting over your code doesn't have to happen at each application startup, it only happens once at build time.
- Improved error reporting. Because the generator operates on the source code it can provide file and line information when an error is detected.
- You will discover all errors at compile time, rather than when the code is exercised.

How to use it

New projects

If you create a new project from template (the recommended way), there is nothing you need to do to take advantage of this new feature, it is enabled by default.

by editing the MSBuild project (the `.vcxproj` or `.csproj` file) and add:

```
    <PropertyGroup>
      ...
+     <ReactNativeCodeGenEnabled>true</ReactNativeCodeGenEnabled>
    </PropertyGourp>
```

Copy

Next you will have to add the reflection-based generation. Open `ReactPackageProvider.cs` and make the following changes:

1. Add a using statement at the top of the file

```
    using Microsoft.ReactNative;
+ using Microsoft.ReactNative.Managed;
```

Copy

2. Replace the call to the generated implementation with a reflection-based call.

```
        public void CreatePackage(IReactPackageBuilder packageBuilde
        {
-           CreatePackageImplementation(packageBuilder);
+           packageBuilder.AddReflectionReactPackageProvider<ReactPackagePrc
        }
```

Copy

> Disclaimer: The reflection logic may be removed in the near future.

Upgrading old projects

When you upgrade an existing 0.62 project to a later version you will already have imports for the shared files so you will get the codegen by default. It might be the case that your C#

```
using Microsoft.ReactNative;

namespace ...
{
    public partial class ReactPackageProvider : IReactPackageProvider
    {
        public void CreatePackage(IReactPackageBuilder packageBuilder)
        {
            CreatePackageImplementation(packageBuilder);
        }

        /// <summary>
        /// This method is implemented by the C# code generator
        /// </summary>
        partial void CreatePackageImplementation(IReactPackageBuilder package
    }
}
```

2. Hook up the registration in the constructor in `App.xaml.cs`:

```
        public App()
        {
            ...

            PackageProviders.Add(new Microsoft.ReactNative.Managed.ReactPackag
+           PackageProviders.Add(new ReactPackageProvider());
```

How it works

`windows\MyProject\obj\x64\Debug\MyProject.ReactNativeCodeGen\MyProject.ReactNativeCodeGen.rsp`. The tool takes the same arguments that the C# compiler receives, the source files, the resolved package and project references and the defines. It then uses those to create a workspace using the C# compiler APIs (Roslyn) which analyze the code and emit the registration calls to a generated C# file next to the `.rsp` file. For example `windows\MyProject\obj\x64\Debug\MyProject.ReactNativeCodeGen\MyProject.ReactNativeCodeGen.g.cs`. This auto-generated C# file is then included into your application.

Your application needs to have two pieces of code to ensure it is hooked up. If you use the templates to create the project, everything is taken care of. This paragraph just intends to explain how it works. First it needs to have a partial class called `ReactPackageProvider` in the default namespace of your project. This class must have a partial method declared with the following signature:

```
partial void CreatePackageImplementation(IReactPackageBuilder packageBuilder);
```

Copy

which the generated file implements. This class should implement `IReactPackageProvider` and call the partial method from the `CreatePackage` function. You then need to register this class in your App's XAML startup code by adding the following call to the constructor:

```
PackageProviders.Add(new ReactPackageProvider());
```

Copy

You can find the full code if you create a new template, or in the instructions for upgrading old projects above.

‹ Using JSValue

Developing Windows apps on a non-Windows PC ›