## Lab: XAM427 - Enterprise WCF

### Prerequisites

You will need a development environment, either a Windows PC with the Android SDK, Visual Studio and either Xamarin.iOS or Xamarin.Android installed. We will be using the Android emulator to test the code we are building, so make sure to have a virtual device already configured and ready to run. See the **Visual Studio with Xamarin** setup documentation if you need help getting your environment setup: <http://docs.xamarin.com/guides/cross-platform/getting_started/visual_studio_with_xamarin/>

### Downloads

<https://university.xamarin.com/classes/xam427-enterprise-wcf-integration>

### Lab Goals

The goal of this lab will be to build a simple WCF service, build a client for that service and then consume it from within our app

## Steps

### Open the Starting Solution

1. Launch Visual Studio 2012/3.
2. Open up the project **Xamarin\_EnterpriseWCF\_Begin**
3. Right-Click and set the **WcfServiceHost** project as the startup project  
   

### Creating the WCF Service

1. Locate the comment TODO: Step 1 - Define our service contract and uncomment the code

[OperationContract]

string GetRandomMonkeyName();

[OperationContract]

IEnumerable<MonkeyInformation> GetMonkeyMatch(MonkeyQuery query);

1. Locate the comment TODO: Step 2 - Create an implementation our service implementation and uncomment the code below

public string GetRandomMonkeyName()

{

    var random = new Random();

    var monkeyInformation = MonkeyData.GetMonkeyInformation();

    return

        monkeyInformation.Any()

            ? monkeyInformation.ElementAt(random.Next(monkeyInformation.Count)).CommonName

            : string.Empty;

}

public IEnumerable<MonkeyInformation> GetMonkeyMatch(MonkeyQuery query)

{

    var monkeyInformation = MonkeyData.GetMonkeyInformation();

    return

        query == null || (String.IsNullOrEmpty(query.Family) && String.IsNullOrEmpty(query.Subfamily) && String.IsNullOrEmpty(query.Genus))

        ? Enumerable.Empty<MonkeyInformation>()

        : monkeyInformation

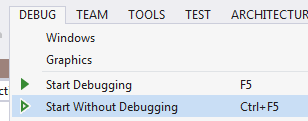
            .Where(mi => String.IsNullOrEmpty(query.Family) || mi.Family.IndexOf(query.Family, StringComparison.OrdinalIgnoreCase) >= 0)

            .Where(mi => String.IsNullOrEmpty(query.Subfamily) || mi.Subfamily.IndexOf(query.Subfamily, StringComparison.OrdinalIgnoreCase) >= 0)

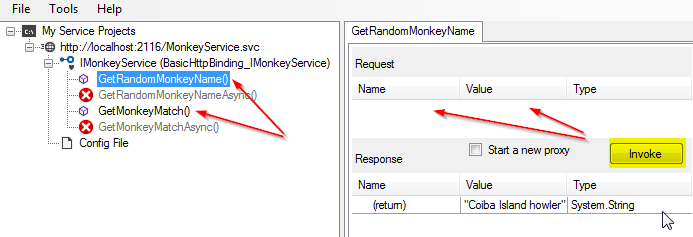
            .Where(mi => String.IsNullOrEmpty(query.Genus) || mi.Genus.IndexOf(query.Genus, StringComparison.OrdinalIgnoreCase) >= 0);

}

1. Highlight the MonkeyService.svc file and select BUILD > Start Without Debugging



1. You should get a WCF Test Client displayed. Click a service such as GetRandomMonkeyName and select **Invoke**



1. You should get a result based on the data that you provided

**TIP:** This project is running under IIS Express and will only be available locally. If you would like to configure IIS Express to work remotely, follow the steps located here: **<http://www.microsoft.com/en-us/download/details.aspx?id=28359>**

### Create the WCF Service Client

1. Locate the **ServiceGenerator.bat** file as part of the solution



1. Right-Click on this file and select **Open Command Prompt**



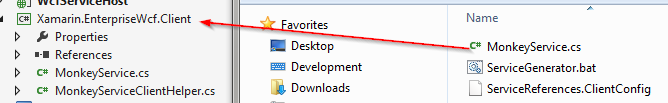
1. Run the ServiceGenerator.bat file



1. Return to Visual Studio
2. Right-Click the ServiceGenerator.bat file and Select Open Containing Folder



1. Drag-and-Drop the MonkeyService.cs file into the Xamarin.EnterpriseWcf.Client Project



1. Locate TODO: Step 3 - Define the service endpoint and uncomment the code below

#if DEBUG

private static readonly EndpointAddress ServiceEndPoint = new EndpointAddress("http://192.168.28.5:2116/MonkeyService.svc");

#else

private static readonly EndpointAddress ServiceEndPoint = new EndpointAddress("http://prodaddress/MonkeyService.svc");

#endif

1. Locate TODO: Step 4 - Create our binding and uncomment the code below

private static BasicHttpBinding CreateBasicHttpBinding()

{

    var binding = new BasicHttpBinding

    {

        Name = "basicHttpBinding",

        MaxBufferSize = 2147483647,

        MaxReceivedMessageSize = 2147483647

    };

    var timeout = new TimeSpan(0, 0, 30);

    binding.SendTimeout = timeout;

    binding.OpenTimeout = timeout;

    binding.ReceiveTimeout = timeout;

    return binding;

}

1. Locate Step 5 - Create a method that provides a configured service client and uncomment the code below

public static MonkeyServiceClient CreateMonkeyServiceClient()

{

    return new MonkeyServiceClient(CreateBasicHttpBinding(), ServiceEndPoint);

}

1. Build the solution

### Add the Client to Android or iOS

1. Locate Step 6 – Android/iOS - Initialize our Service and uncomment the code below

private void InitializeHelloWorldServiceClient()

{

    monkeyServiceClient = MonkeyServiceClientHelper.CreateMonkeyServiceClient();

    monkeyServiceClient.GetMonkeyMatchCompleted += GetMonkeyMatchCompleted;

    monkeyServiceClient.GetRandomMonkeyNameCompleted += GetRandomMonkeyNameCompleted;

}

1. Build and run the application on your preferred platform

## Summary

In this lab, we learned how a mobile application built with Xamarin can consume a WCF web service using the BasicHttpBinding.