XAM151

Cross Platform SOAP-based Web Services

Download class materials from
university.xamarin.com

Microsoft

Xamarin University

Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user.

Microsoft or Xamarin may have patents, patent applications, trademarked, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any license agreement from Microsoft or Xamarin, the furnishing of this document does not give you any license to these patents, trademarks, or other intellectual property.

Xamarin, MonoTouch, MonoDroid, Xamarin.iOS, Xamarin.Android, Xamarin Studio, and Visual Studio are either registered trademarks or trademarks of Microsoft in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.

Objectives

1. Consume SOAP services

Working with SOAP services

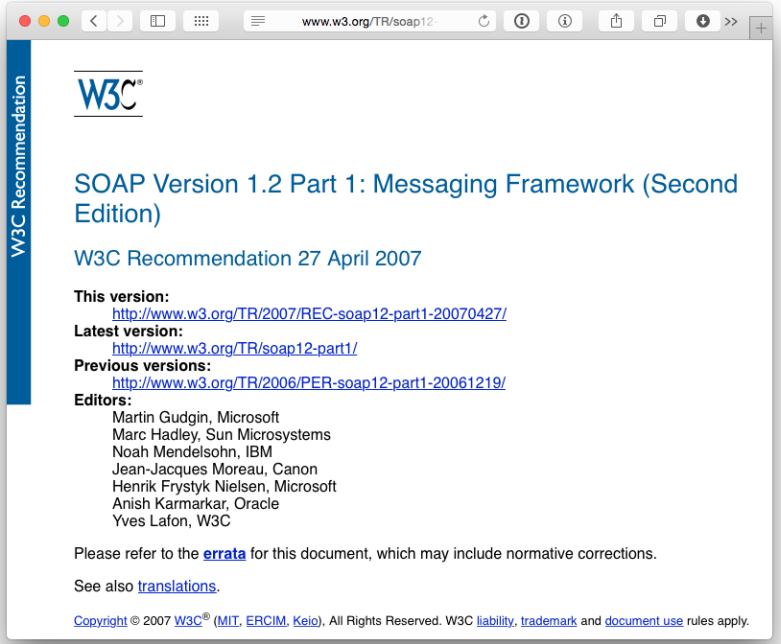
Xamarin University

Tasks

1. Create a WCF Client
2. Connect to a SOAP service
3. Execute SOAP service methods

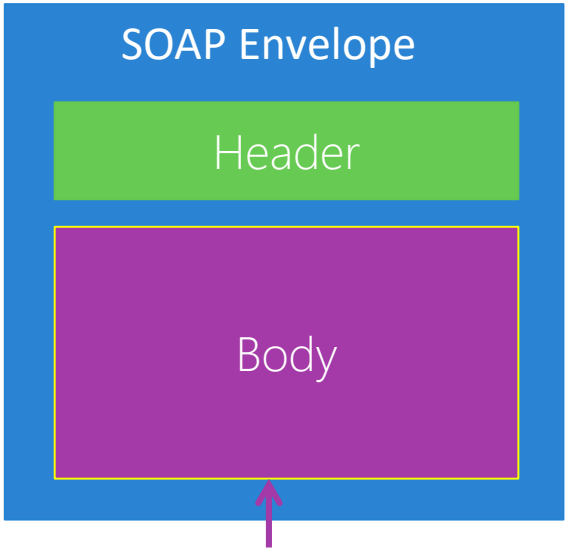
What is SOAP?

- SOAP is a RPC-based specification standard for web services which defines:
  - discovery
  - data format + transfer
  - security
  - data reliability
  - …



SOAP specification

- SOAP has a defined XML structure, referred to as the SOAP Envelope
- Variety of additional specifications available to define various aspects of communication which plug into this structure
  - referred to as WS-*
  - not widely supported on all platforms



SOAP uses HTTP **POST** to call services; the API and parameters are passed in the message body

Comparing REST and SOAP

	REST	SOAP
Standardized	No	Yes (WS-*)
Transport	HTTP	HTTP, FTP, Pipes, MSMQ etc.
Data Format	any, prefer JSON	XML
Reach	Excellent	Good
Development tools	Basic	Good
Flexibility	Flexible	Rigid
Scalability	Excellent	Reasonable
Security	HTTPS	WS-Security
Operations	Stateless	Often statefull

What is WCF?

- Windows Communication Foundation is a highly configurable and flexible .NET library for building distributed, service-oriented applications

- Original vision was to be the "kitchen sink" of communication, but it's flexibility has made it moderately complicated to use

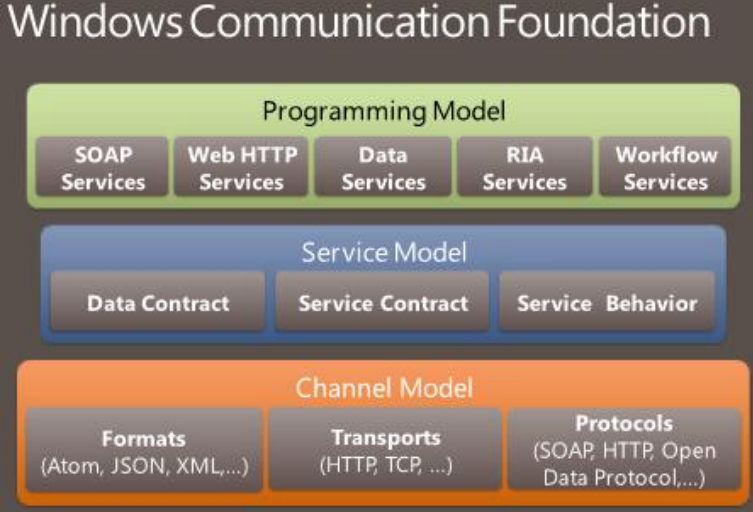


image from msdn.microsoft.com

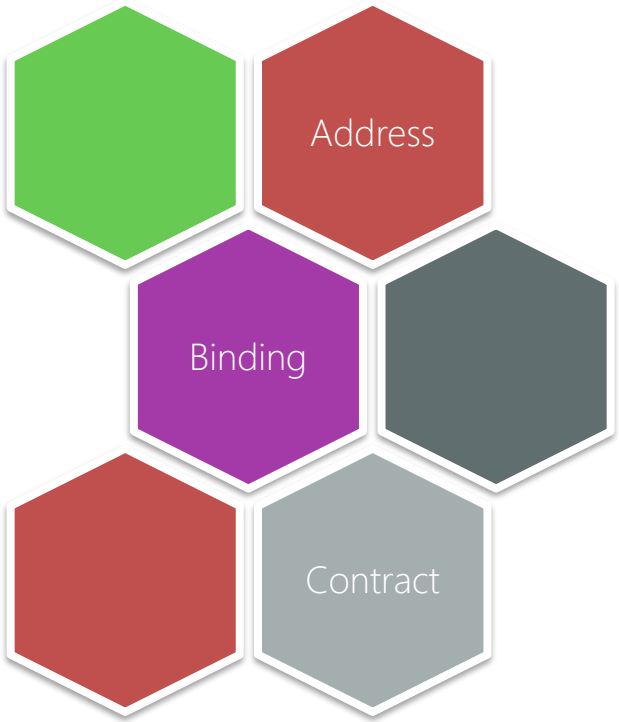
Using WCF with Xamarin

- Adding WCF client-side support to a mobile application today requires a Windows machine with Visual Studio or the **Silverlight SDK**

WCF Endpoints

- Communication with WCF Services is performed over *endpoints* which define the **address**, **binding**, and **contract** that is exposed

- Client and service must agree on these three pieces of information, sometimes called the "ABCs" of WCF



Address

- Service address defines the *location* of the service with two pieces of information

```
http://itunes.apple.com/movies
```

```
net.tcp://23.75.234.144:5544
```

```
https://services.xamarin.com/account
```

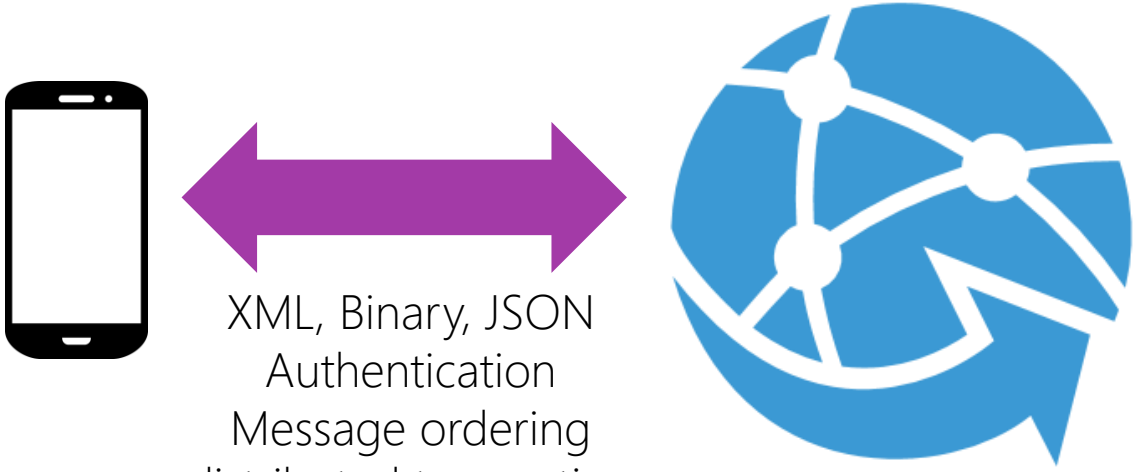
Protocol used to access it

Address where it is hosted

Mobile applications can only utilize HTTP(s)-based addresses

Binding

- Selected **binding** determines how the client and service will communicate and what parts of the SOAP specifications are used



WCF includes pre-defined bindings for various scenarios but mobile apps only have access to a small subset of the defined bindings – that may change in the future

Contract

- The WCF **contract** defines what the service *can do* and is most often modeled as an interface which is shared by the client and service

```
[ServiceContract(Namespace = "http://services.xamarin.com/xamu")]
public interface ICourseCatalog
{

    ...


}
```

`ServiceContract` indicates that this is a contract definition; namespace must match service expectation as it is coded into the SOAP envelope

Contract

- The WCF **contract** defines what the service *can do* and is most often modeled as an interface which is shared by the client and service

```
[ServiceContract(Namespace = "http://services.xamarin.com/xamu")]
public interface ICourseCatalog
{
    [OperationContract]
    Course GetCourseById(int id);
    [OperationContract]
    IList<Course> GetAll();
    [OperationContract]
    void ScheduleCourse(int id, DateTime scheduledDate);
}
```

OperationContract indicates that this is a specific API that can be called

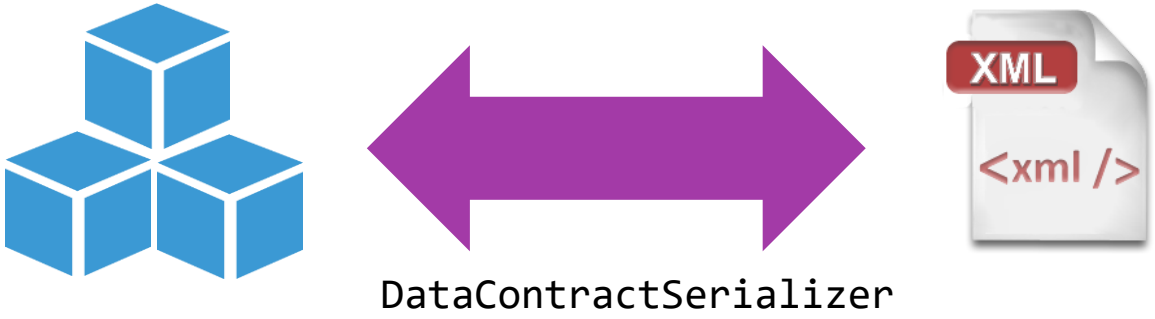
Serialization with SOAP + WCF

- SOAP must serialize the data passed to and from each API
- Serialization format is tied to the *binding*, preferred format for most bindings is **XML + XSD**

```
<?xml version="1.0"
    encoding="utf-16"?>
<Course xmlns="http://...">
  <Description>
     ...
  </Description>
  <Id>
     ...
  </Id>
  <Name>
     ...
  </Name>
</Course>
```

What is DataContractSerializer?

- WCF uses **`DataContractSerializer`** to convert objects into XML and vice-versa



This is the *default serializer*; however WCF is quite extensible and one of those extensibility points is in changing how data is serialized by supplying a different serializer

Controlling the serialization

- Attributes used to map an object and its members to an XSD schema

Objects to be serialized must be marked with the **DataContract** attribute

Fields to be serialized must be marked with the **DataMember** attribute

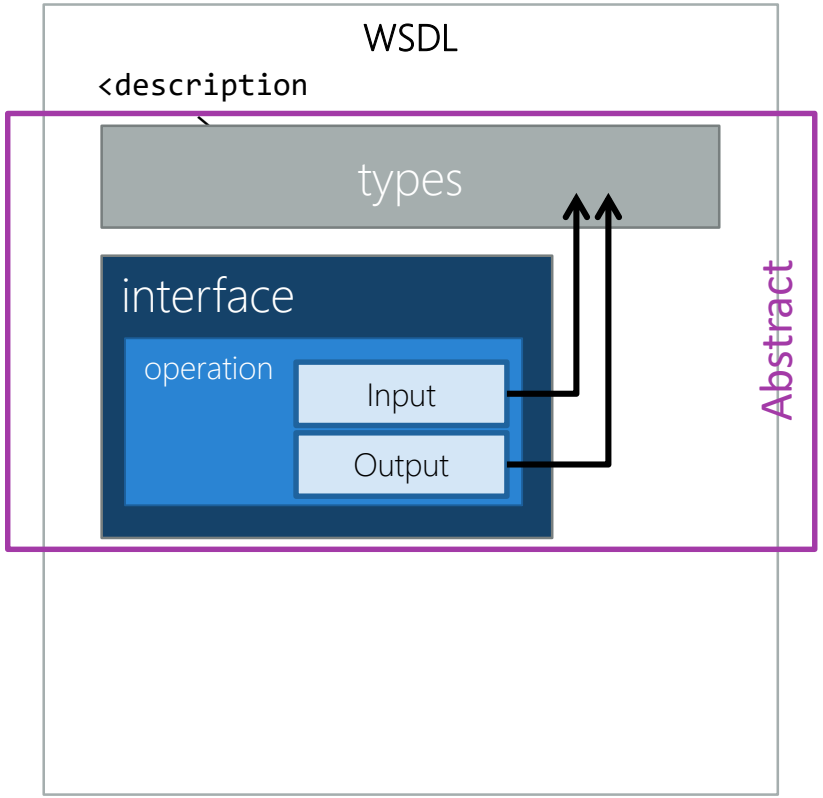
```
[DataContract(Namespace="urn:xamarin.com")]
public class Course
{
    [DataMember(Name="key")]
    public int Id { get; set; }

    [DataMember]
    public string Name { get; set; }

    [DataMember(IsRequired = false)]
    public string Description { get; set; }
}
```


Modern versions of **DataContractSerializer** do not require these attributes and will instead serialize all public properties, but being explicit is a good practice

What is WSDL?

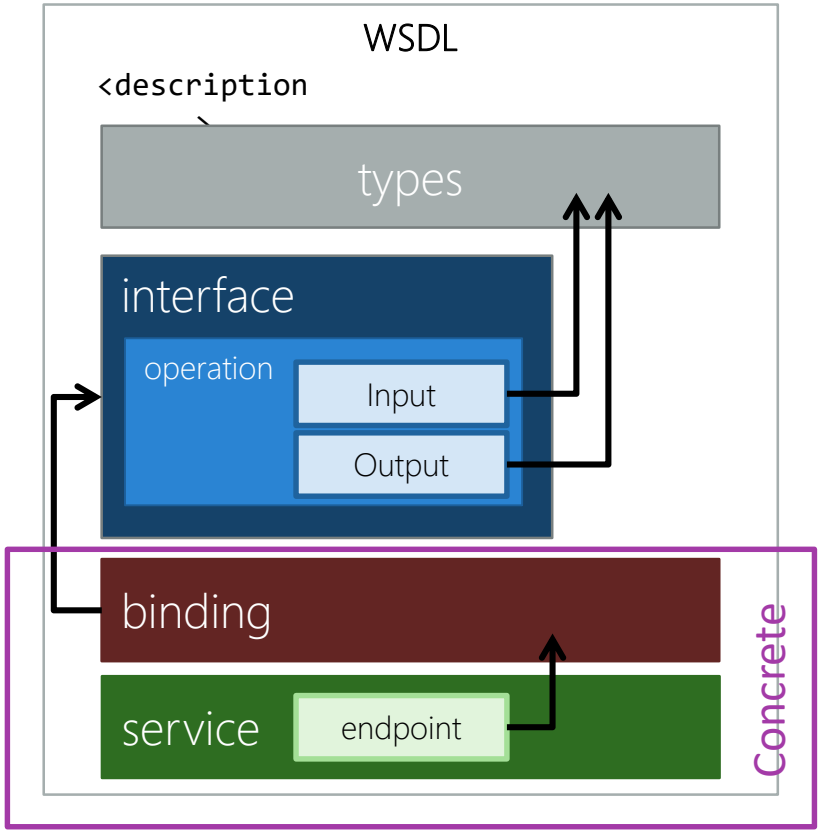
- Web Service Description Language (WSDL) is an XML document that provides the technical description for interacting with a specific a web service

- Tools generate the contracts and necessary WCF binding code directly from WSDL documents



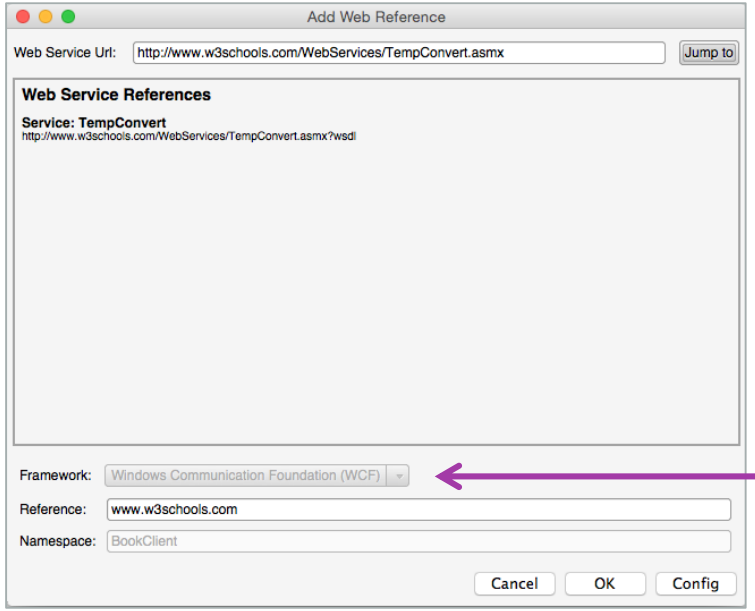
What is WSDL?

- Web Service Description Language (WSDL) is an XML document that provides the documentation for a web service
- Tools generate the contracts and necessary WCF binding code directly from WSDL documents



Generating the contracts

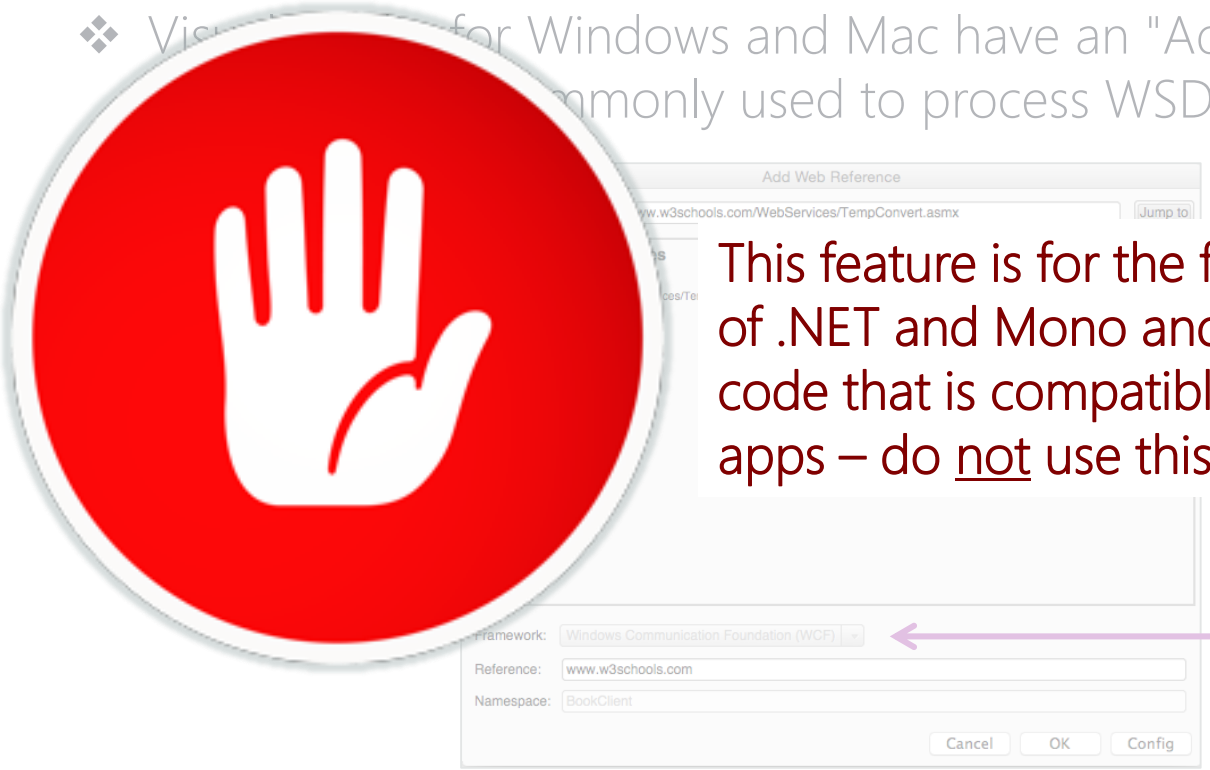
- Visual Studio for Windows and Mac have an "Add Web Reference" dialog which is commonly used to process WSDL



It even supports WCF ...

Generating the contracts

- Vis[illegible] for Windows and Mac have an "Add Web Reference" [illegible]mmonly used to process WSDL

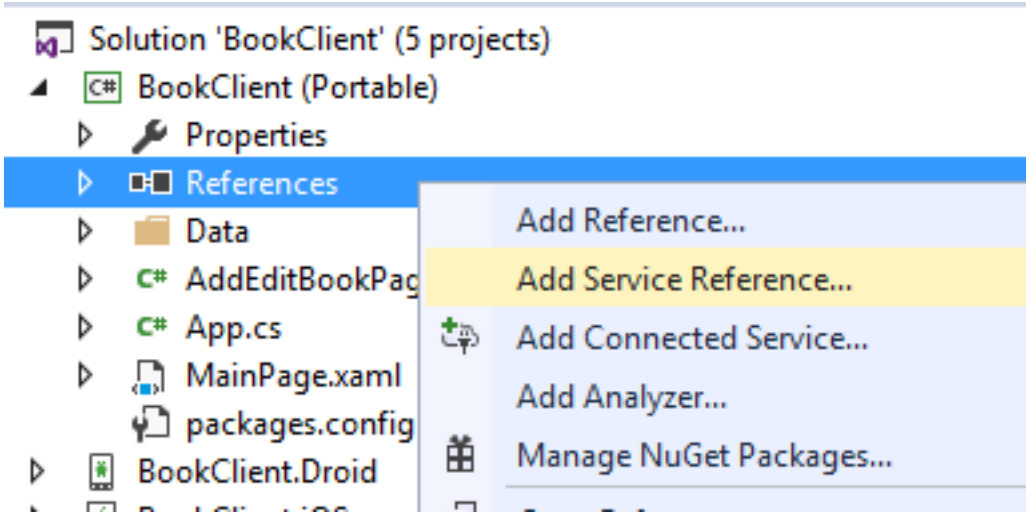


This feature is for the full desktop version of .NET and Mono and will not generate code that is compatible with your mobile apps – do not use this approach

It even supports WCF …

Generating the contracts

- Can use the **Add Service Reference** on the **References** folder in a PCL, as long as the PCL does *not* support Windows Phone 8.1 (WinRT)



Generating the contracts

- Silverlight SDK ships with a command line tool which is used to generate all the client access code from a **WSDL file**

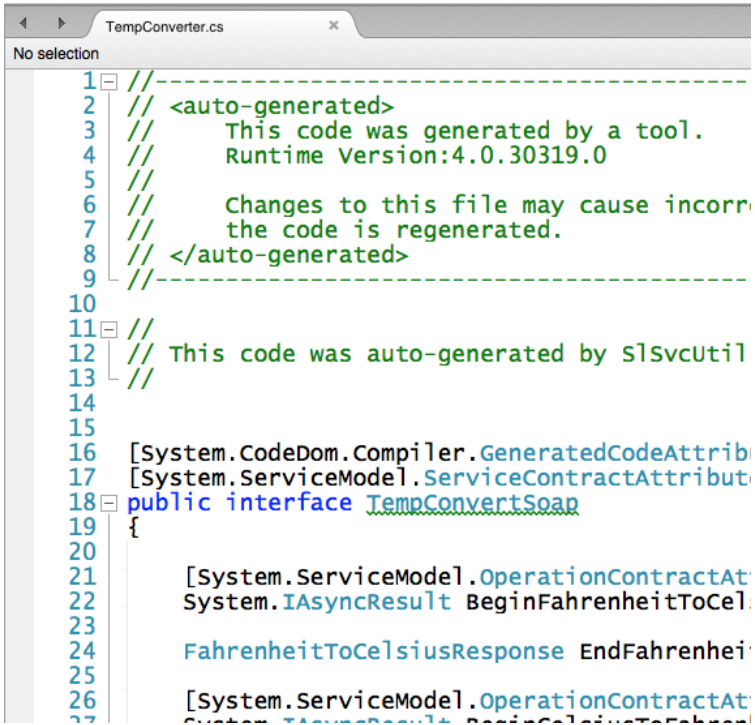
```
C:\Program Files (x86)\Microsoft SDKs\Silverlight\v5.0\Tools\SLSvcUtil.exe
```

```
C:\> SLSvcUtil /noConfig http://www.w3schools.com/WebServices/TempConvert.asmx

Microsoft (R) Silverlight Service Model Proxy Generation Tool
[Microsoft (R) Silverlight SDK, Version 5.0.61118.0]
Copyright (c) Microsoft Corporation.  All rights reserved.

Attempting to download metadata from 'http://www.w3schools.com/WebServices/TempConvert.asmx' using W
S-Metadata Exchange or DISCO.

Generating files...
C:\Users\Mark\Desktop\TempConvert.cs
```


Caution: There is also a desktop utility **`SVCUtil.exe`** but it generates desktop based code which will not run properly on the mobile platform today

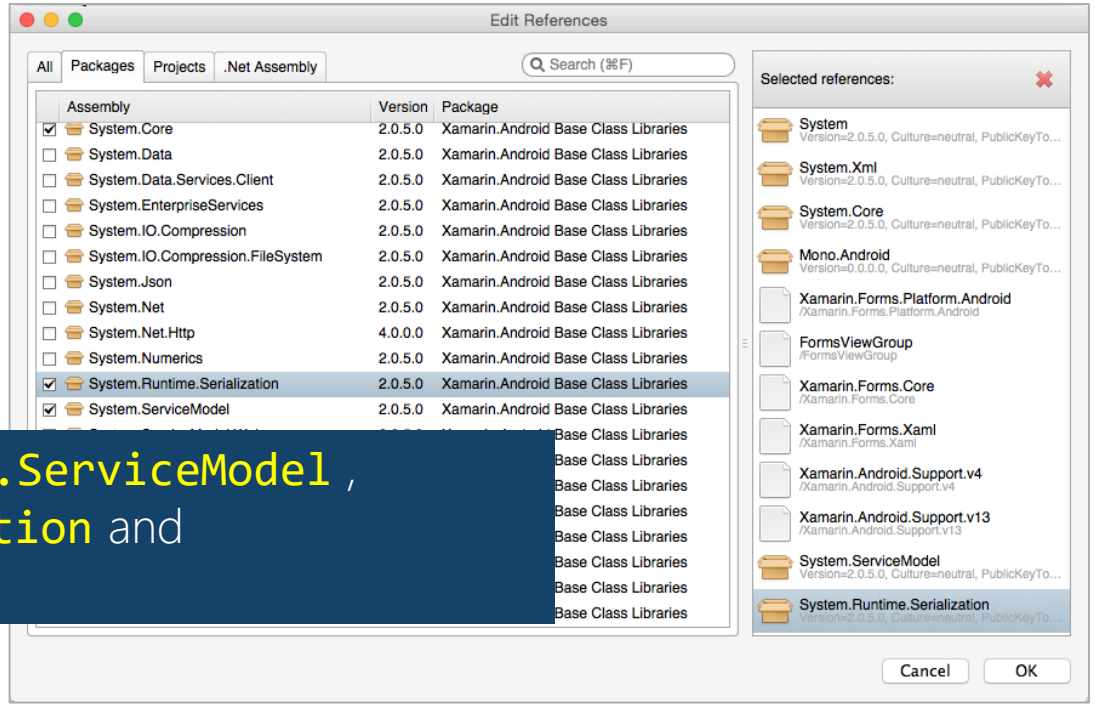
What gets generated?

- Tool processes the WSDL and generates a single C# source file with all the necessary contracts and a proxy class used to invoke the operations on the web service



Reminder: add WCF references

- When using **`SLSVCUtil`** to generate the proxy source file, you must also manually add the DLL references to support WCF



Must add references to `System.ServiceModel`, `System.Runtime.Serialization` and `System.ServiceModel.Web`

Calling the web service

- Tools generate a web *proxy* which exposes asynchronous RPC-style methods representing the operations on the web service

```
var client = new TempConvertSoapClient(
          new BasicHttpBinding(BasicHttpSecurityMode.None),
          new EndpointAddress(
              "http://www.w3schools.com/WebServices/TempConvert.asmx"));

client.FahrenheCelsiusCompleted += (sender, e) => {
     Devi                                  () => label.Text = e.Result);
};

client.Fa                                  );
```

Must provide binding and address to proxy constructor

Calling the web service

- Tools generate a web *proxy* which exposes asynchronous RPC-style methods representing the operations on the web service

```
var client = new TempConvertSoapCli
         new BasicHttpBinding(Basic
         new EndpointAddress(
            "http://www.w3schools.c                    asmx"));

client.FahrenheitToCelsiusCompleted += (sender, e) => {
     Device.BeginInvokeOnMainThread(() => label.Text = e.Result);
};

client.FahrenheitToCelsiusAsync("75");
```

Async methods use event model, wire up to **Completed** event

Calling the web service

- Tools generate a web *proxy* which exposes asynchronous RPC-style methods representing the operations on the web service

```
var client = new TempConvertSoapClient(
          new BasicHttpBinding(BasicHttpSecurityMode.None),
          new EndpointAddress(
              "http://www.w3schools.com/WebServices/TempConvert.asmx"));

client.FahrenheitToCelsiusCompleted += (sender, e) => {
     Device.BeginInvokeOnMainThread(() => label.Text = e.Result);
};

client.FahrenheitToCelsiusAsync("75");
```

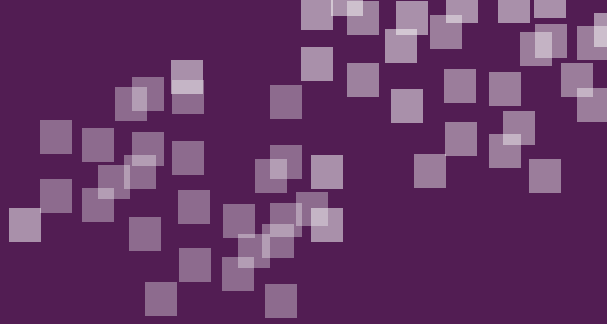
.. and then call method

Calling the web service

- Tools generate a web *proxy* which exposes asynchronous RPC-style methods representing the operations on the web service

```
var client = new TempConvertSoapClient(
          new BasicHttpBinding(BasicHttpSe
          new EndpointAddress(
              "http://www.w3schools.com/Web                    );

client.FahrenheitToCelsiusCompleted += (sender, e) => {
     Device.BeginInvokeOnMainThread(() => label.Text = e.Result);
};

client.FahrenheitToCelsiusAsync("75");
```

Completed callback occurs with status + response on background thread



Individual Exercise

Calling a SOAP Book Service

Xamarin University

Summary

1. What is SOAP and WCF?
2. Creating a WCF Client
3. Connecting to a SOAP service
4. Execute service methods

Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile