MOBILE DEVELOPMENT

CONSUMING WEB SERVICES USING SOAP AND REST APPS

CONTENTS

OVERVIEW

A WebService is a Consumer_Machine-to-Provider_Machine collaboration schema that operates over a computer network.

The data exchanges occur independently of the OS, browser, platform, and programming languages used by the provider and the consumers.

A provider may expose multiple EndPoints (sets of WebServices), each offering any number of typically related functions.

WebServices expect the computer network to support standard Web protocols such as XML, HTTP, HTTPS, FTP, and SMTP.

Example: Weather information, money exchange rates, world news, stock market quotation are examples of applications that can be modeled around the notion of a remote data-services provider surrounded by countless consumers tapping on the server's resources.

OVERVIEW
(Advantages of using the webService architecture)

Under the WebService strategy the invoked functions are implemented once (in the server) and called many times (by the remote users).

Some advantages of this organization are:

- Elimination of redundant code,
- Ability to transparently make changes on the server to update a particular service function without clients having to be informed.
- Reduced maintenance and production costs.

OVERVIEW (Why should Android developer learn how to create a WebService?)

Simple apps are usually self-contained and do not need to collaborate with other parties to obtain additional data or services (for instance, think of a scientific calculator)

However, there are many cases in which the data needed to work with is very extensive, or changes very often and cannot (should not) be hard-coded into the app. Instead, this kind of data should be requested from a reliable external source (for instance, what is the Euro-to-Dollar rate of change right now?)

Another class of apps requires a very high computing power perhaps not available in the mobile device (think of problems such as finding the shortest/fastest route between to mapped locations, or best air-fare & route selection for a traveler)

It is wise for an Android developer to learn how to solve typical problems that exceed the capacities of the handheld devices. Understanding the possibilities offered by the client-server computing model will make the developer be a more complete and better professional.

OVERVIEW (WEBSERVICE ARCHITECTURE)

An ideal Webservice provider is designed around four logical layers which define the ways in which data is to be transported, encoded, exposed and discovered by the users.

Layers	**Responsibility**
Transport	Move messages through the network, using HTTP, SMTP, FTP, …
Messaging	Encoding of data to be exchanged (XML)
Description	WSDL (Web Service Desc. Lang) used for describing public methods available from the endpoint
Discovery	UDDI (Universal Description & Discovery Integration) facilitates location and publishing of services through a common registry

CLIENT SIDE – CONSUMING WEBSERVICES

There are two widely used forms of invoking and consuming WebServices:

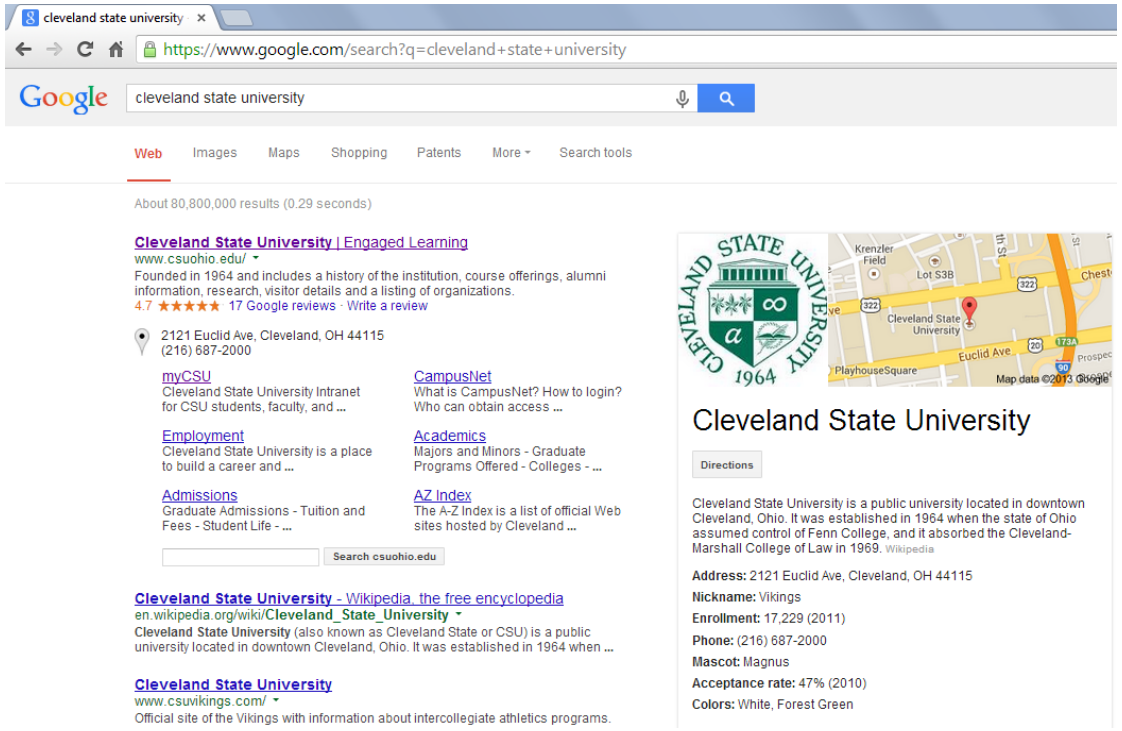
- Representational State Transfer (REST): Closely tie to the HTTP protocol by associating its operation to the common methods: GET, POST, PUT, DELETE for HTTP/HTTPS. This model has a simple invocation mode and little overhead. Service calls rely on a URL which may also carry arguments. Sender & receiver must have an understanding of how they pass data items from one another. As an example: Google Maps API uses the REST model.
- Remote Procedure Call (RPC): Remote services are seen as coherent collections of discoverable functions (or method calls) stored and exposed by EndPoint providers. Some implementations of this category include: Simple Object Access Protocol (SOAP), Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) and Sun Microsystems's Java/Remote Method Invocation (RMI).

CLIENT SIDE – CONSUMING WEBSERVICES

Example: Using REST. The following URL is used to make a call to the Google Search service asking to provide links to the subject “Cleveland State University”

https://www.google.com/search?q=cleveland+state+university

Transport Provider Action Arguments

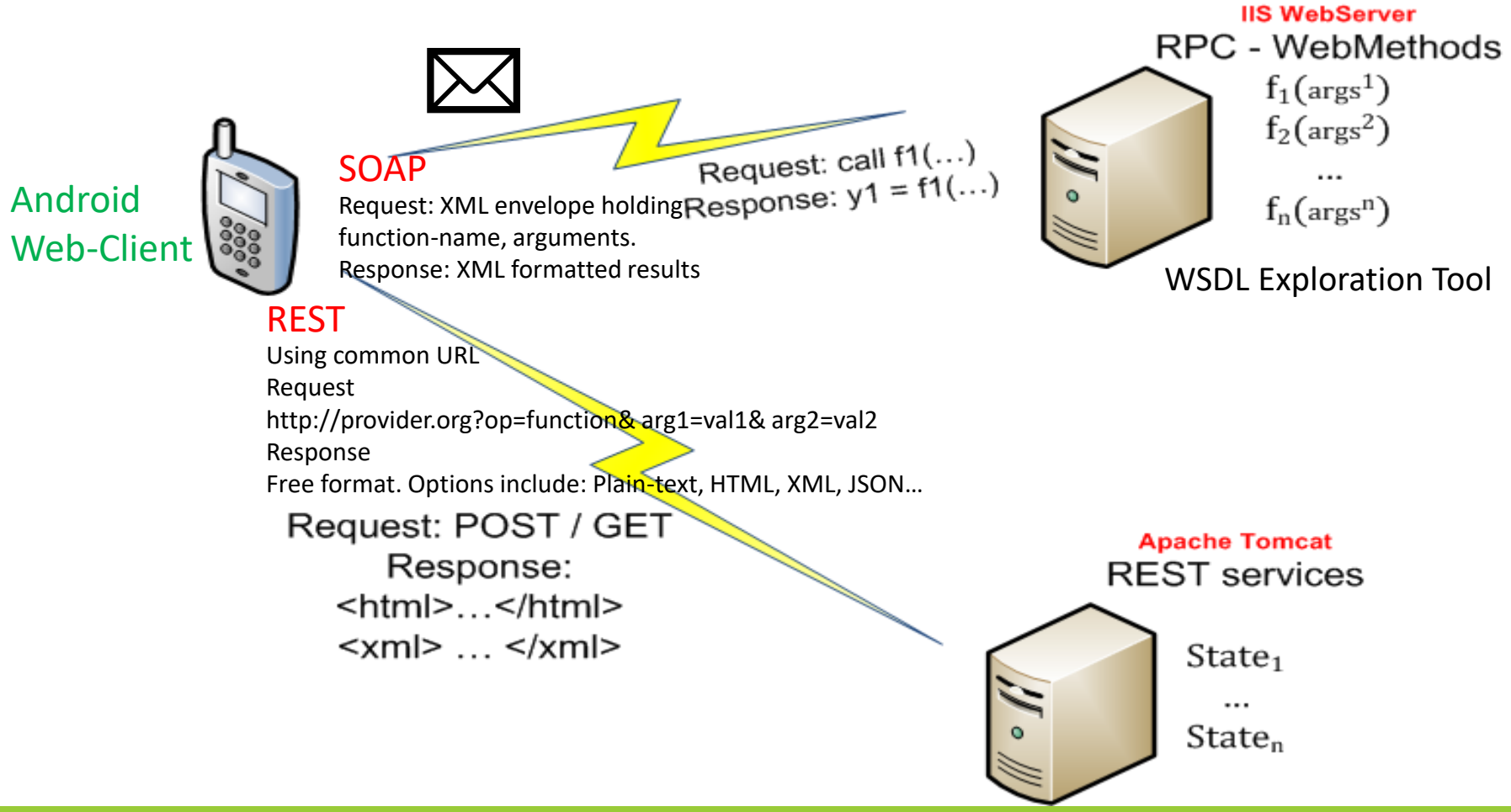


CLIENT SIDE – CONSUMING WEBSERVICES (REST vs. SOAP)

Although SOAP and REST technologies accomplish the same final goal, that is request and receive a service, there are various differences between them.

- REST users refer to their remote services through a conventional URL that commonly includes the location of the (stateless) server, the service name, the function to be executed and the parameters needed by the function to operate (if any). Data is transported using HTTP/HTTPS.
- SOAP requires some scripting effort to create an XML envelop in which data travels. An additional overhead on the receiving end is needed to extract data from the XML envelope. SOAP accepts a variety of transport mechanisms, among them HTTP, HTTPS, FTP, SMTP, etc.
  - SOAP uses WSDL (WebService Description Language) for exposing the format and operation of the services. REST lacks an equivalent exploratory tool.

CLIENT SIDE – CONSUMING WEBSERVICES (WebClient consuming services using REST & SOAP)



EXAMPLES OF ANDROID APPS USING REST & SOAP

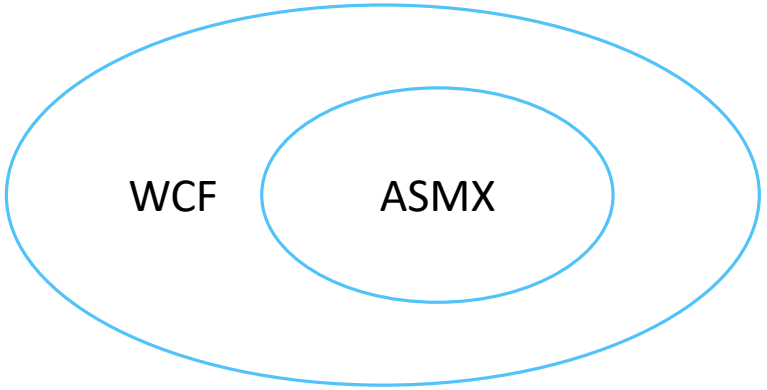
In the next sections we will present three examples showing how an Android web-client typically interacts with a remote server requesting and consuming WebServices.

- Example 1. SOAP client / .NET provider: An Android app uses a XML KSOAP envelope to call a Windows IIS server. WebServices are implemented as a set of C#.NET functions.
- Example 2. REST client / PHP provider: A REST Android client invokes remote PHP services which consult a database on behalf of the client. The response is formatted using JSON.
- Example 3. REST client / Servlet provider: Our Android app communicates with an Tomcat Server in which its WebServices are implemented as Java Servlets. As in the previous example, the results of a database query are returned as a JSON string.

WINDOWS COMMUNICATION FOUNDATION (WCF)

WCF is a Microsoft technology that provides a framework for writing code to communicate across heterogeneous platforms [1, 2].

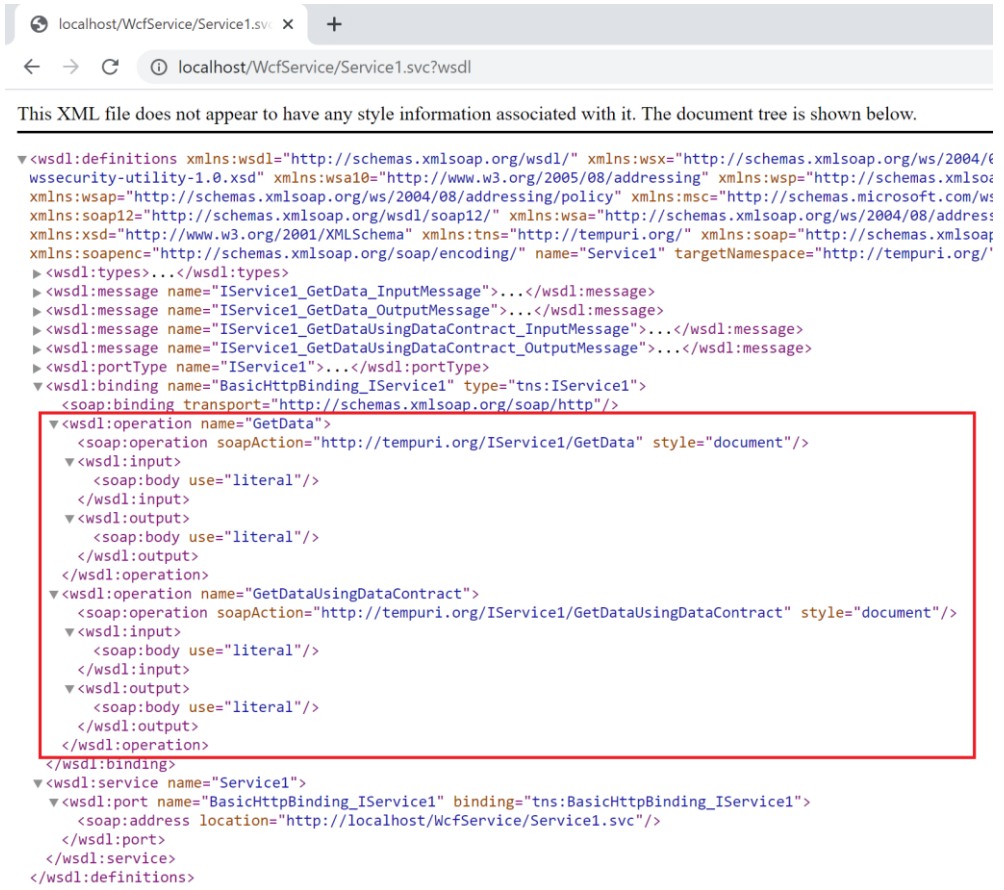
- 1. An IIS WebServer may host various EndPoints (WebServices).
- 2. Each of those EndPoints uses WSDL to provide a way of exposing its composition and behavior to clients wishing to find and communicate with the services.
- 3. Each endpoint includes:
  - address (URL - where to send messages),
  - binding (how to send messages), and a
  - contract (an explanation of what messages contain)



https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/february/learn-the-abcs-of-programming-windows-communication-foundation
https://docs.microsoft.com/en-us/dotnet/framework/wcf/getting-started-tutorial?redirectedfrom=MSDN

WINDOWS COMMUNICATION FOUNDATION (WSDL service contracts)

Example:

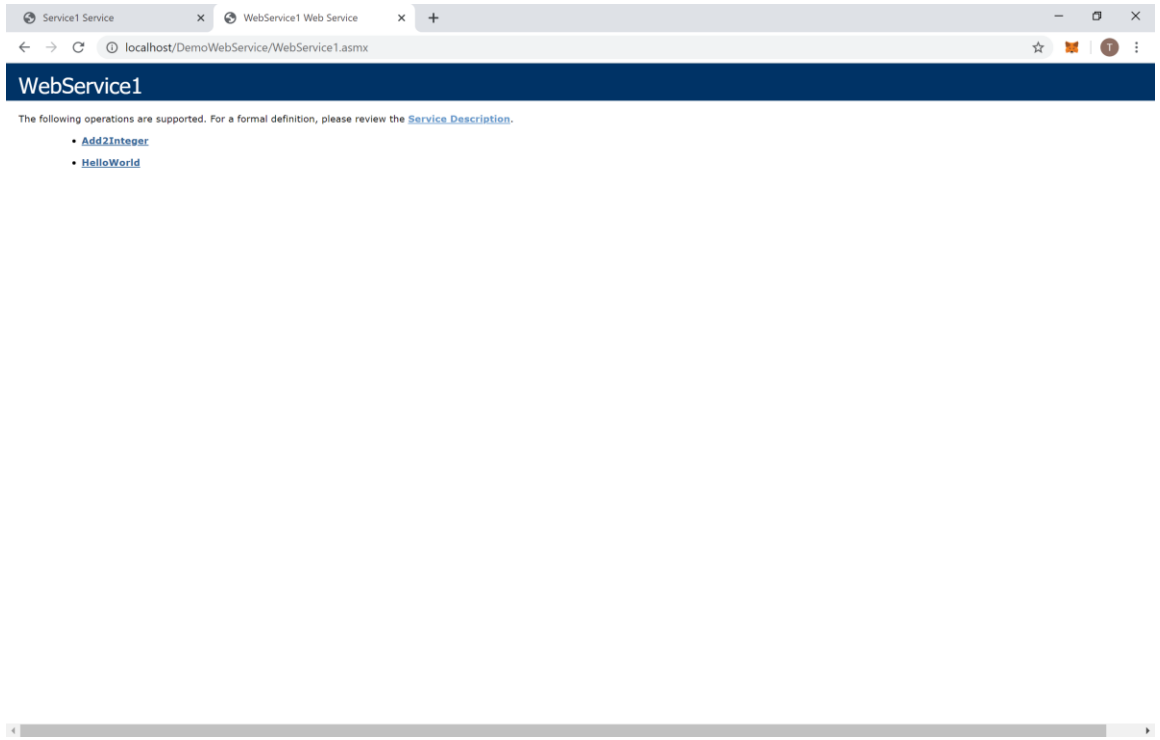


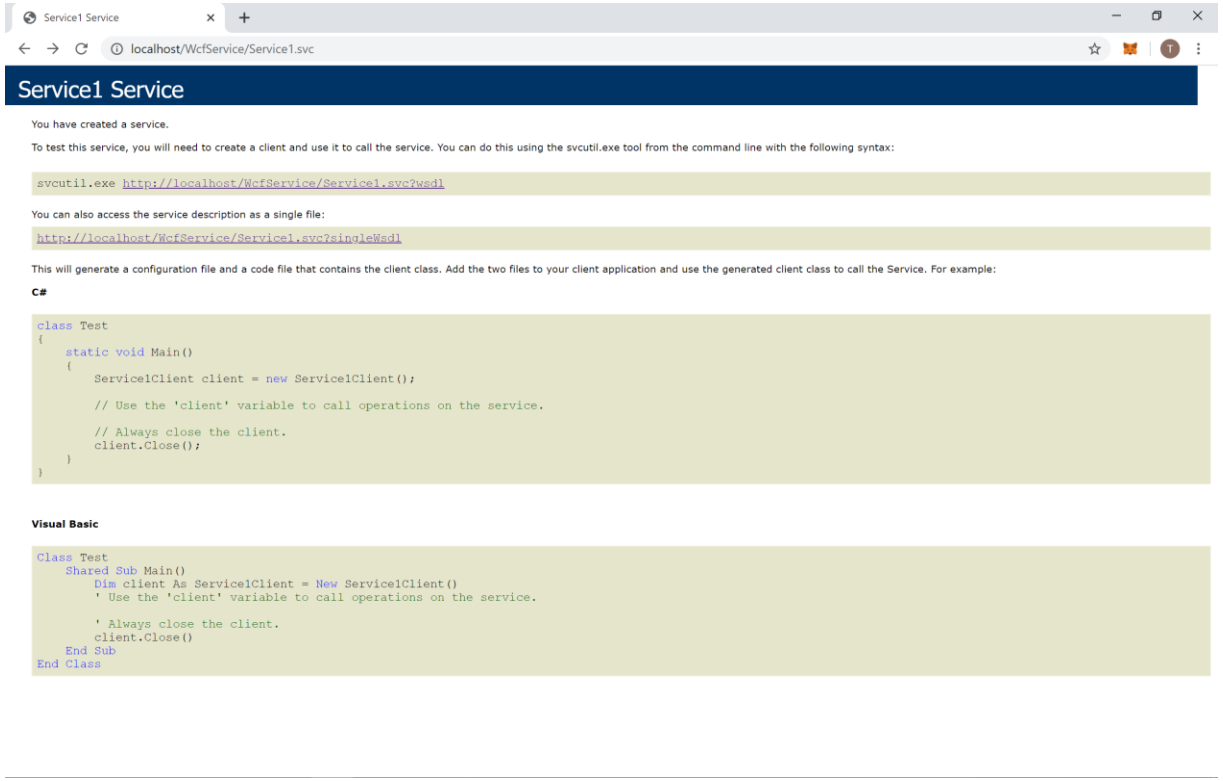
ASMX

WCF

WINDOWS COMMUNICATION FOUNDATION (WCF)

Removing “?WSDL” from the links will exposes endpoint service functions



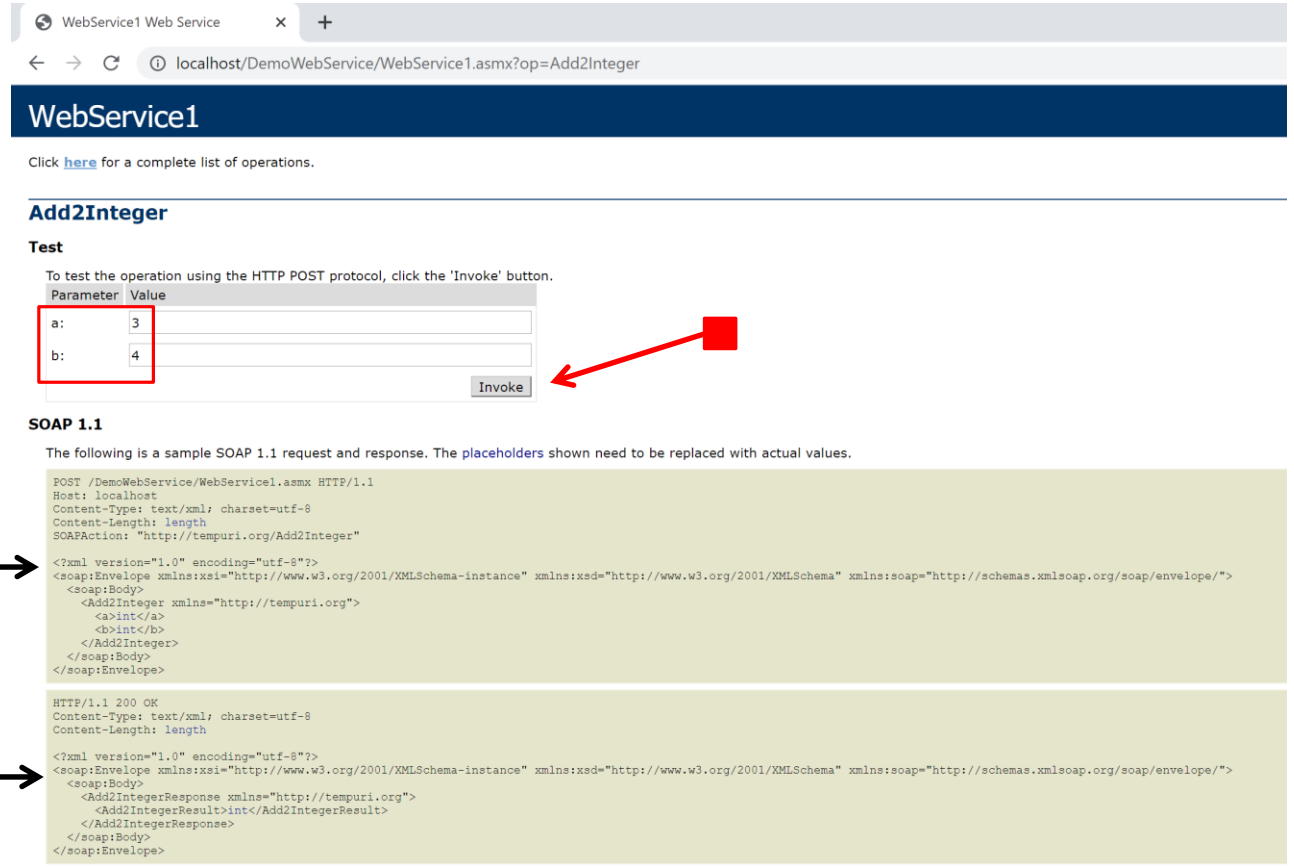


ASMX

WCF

WINDOWS COMMUNICATION FOUNDATION (WCF)

Figures shows WSDL Service Contracts & SOAP Envelopes (ASMX)



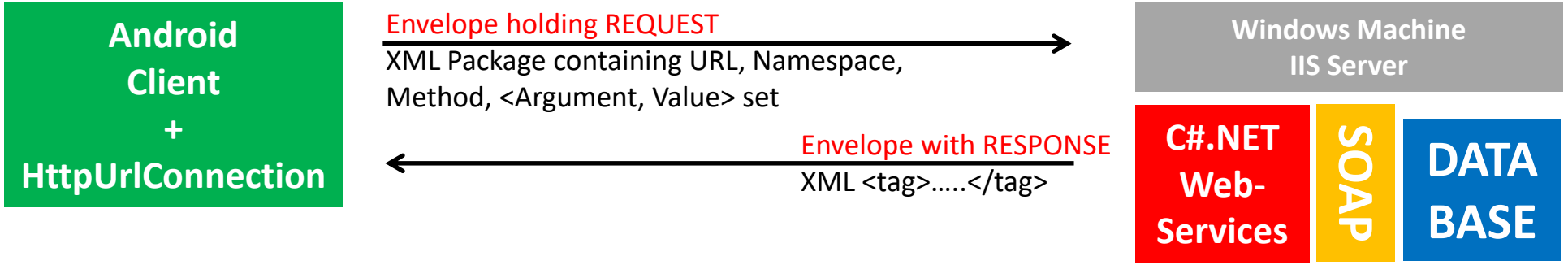
Outgoing Envelop (Request)

Incoming Envelop (Response)

WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

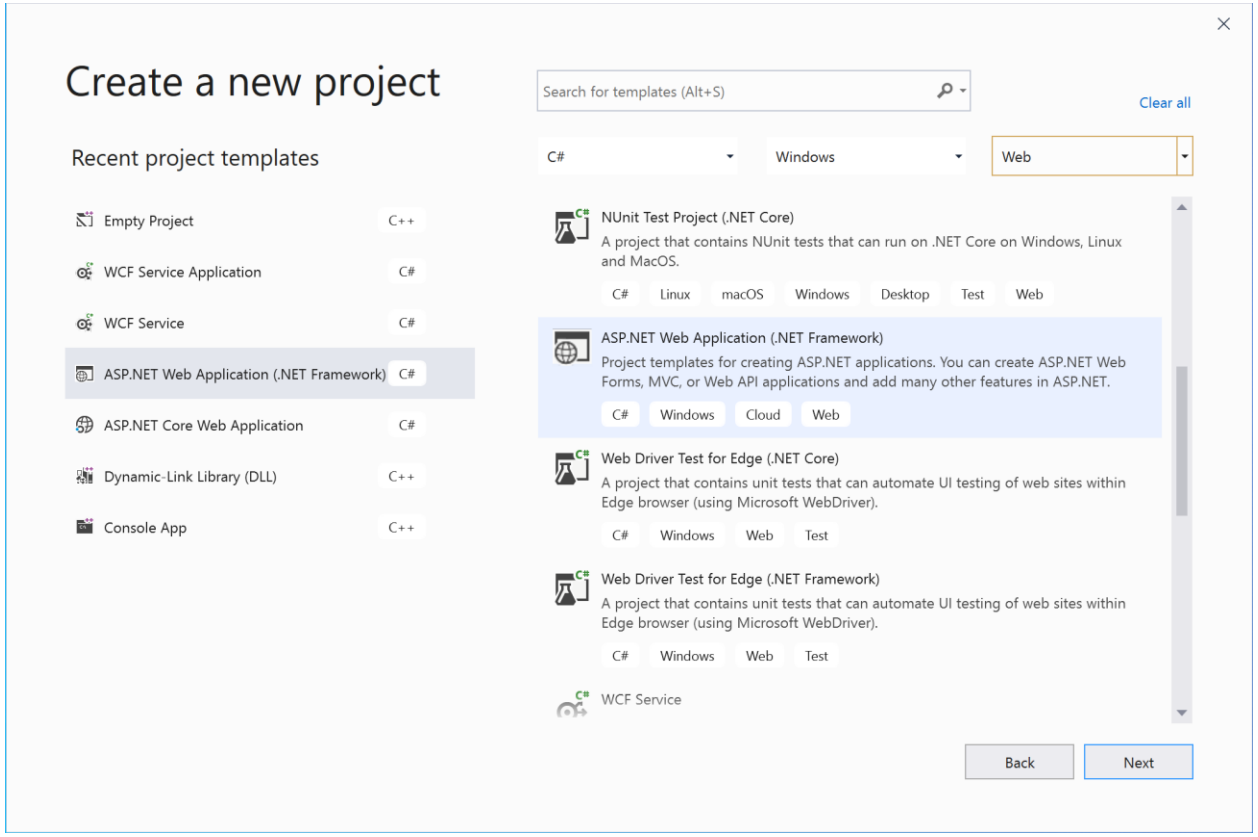
This example consists of two parts. Firstly we construct a server-side EndPoint offering a set of Windows IIS web-services, in the second part we build an Android HttpUrlConnection client that requests and consumes the previous web-services.

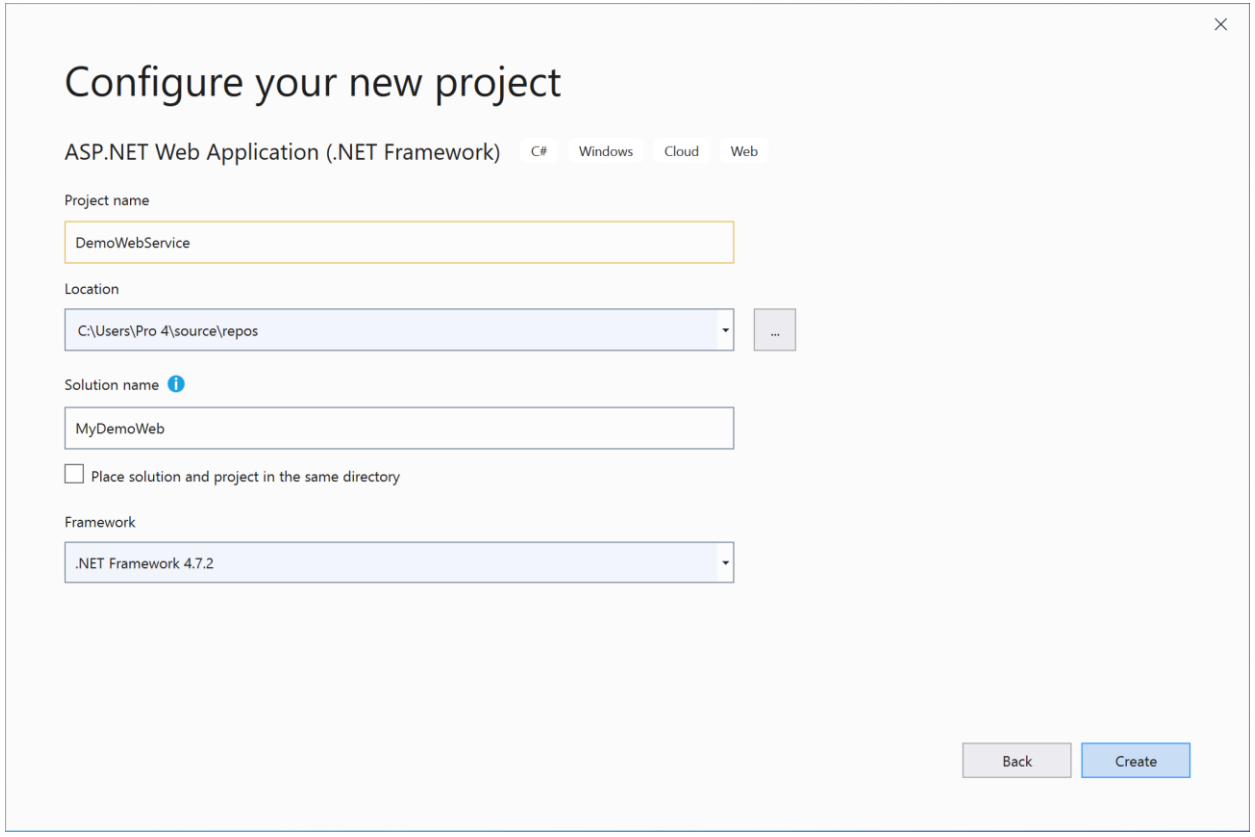
- Server-side software: provide a step-by-step tutorial describing how to create a web-service running on a Windows IIS-Server. We will create additional one methods: add2Integer.
- Client-side: HttpUrlConnection facilitates sending requests and receiving results to/from an IIS server.



WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

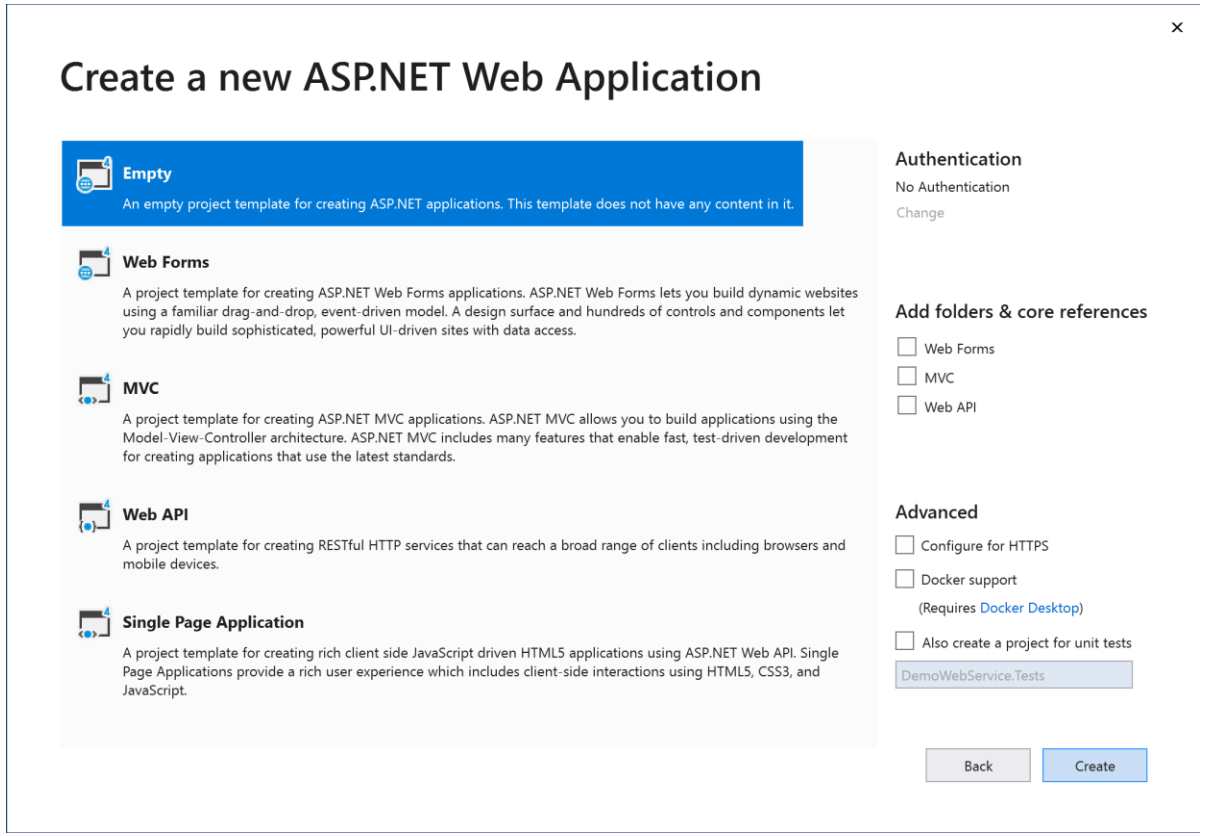
1. Create webservice application project

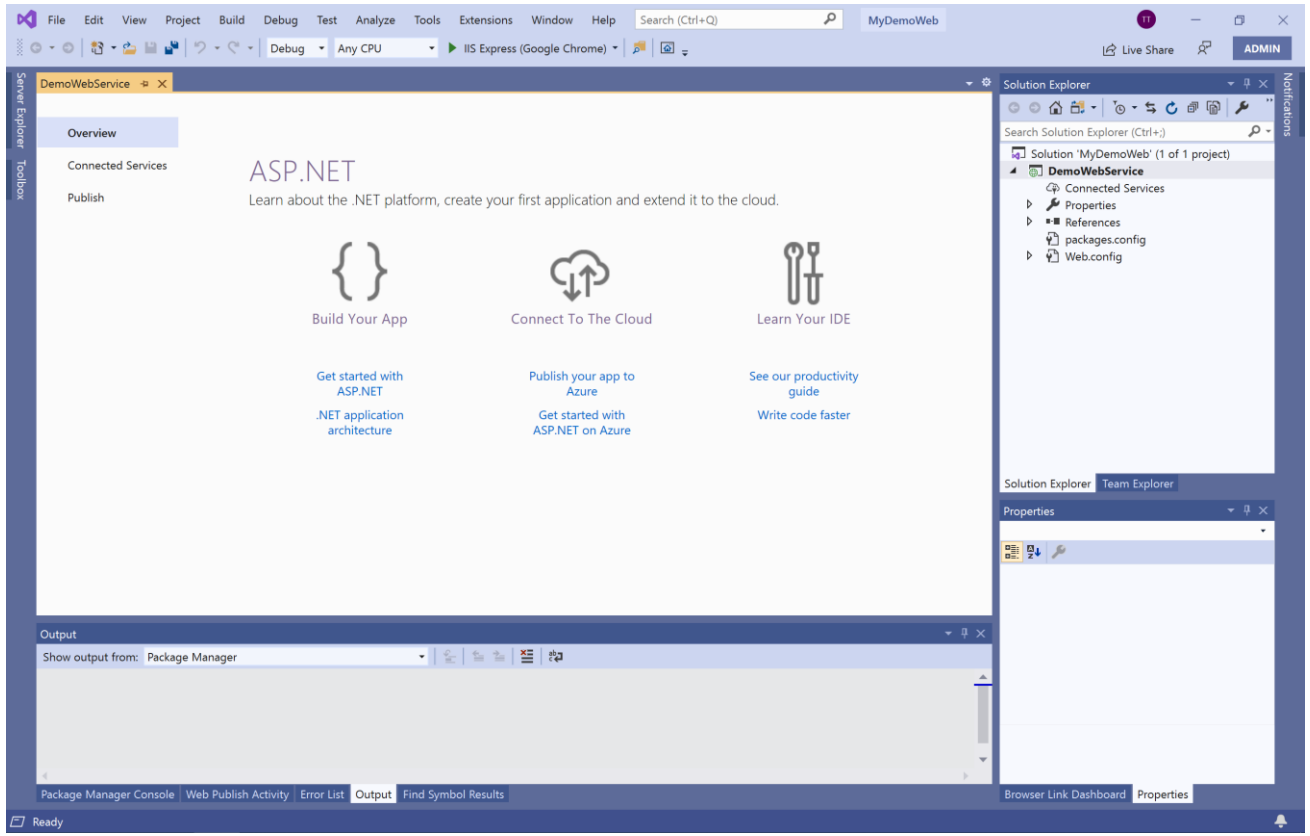




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

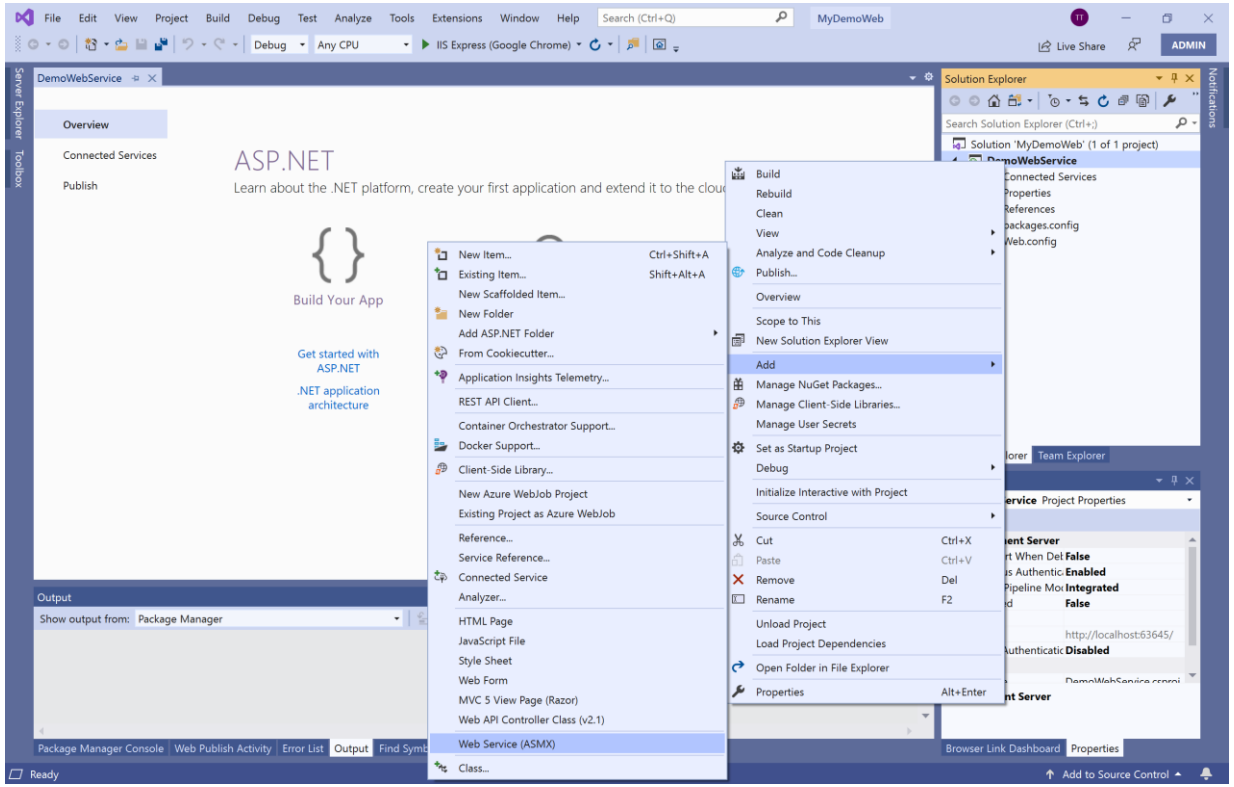
1. Create webservice application project

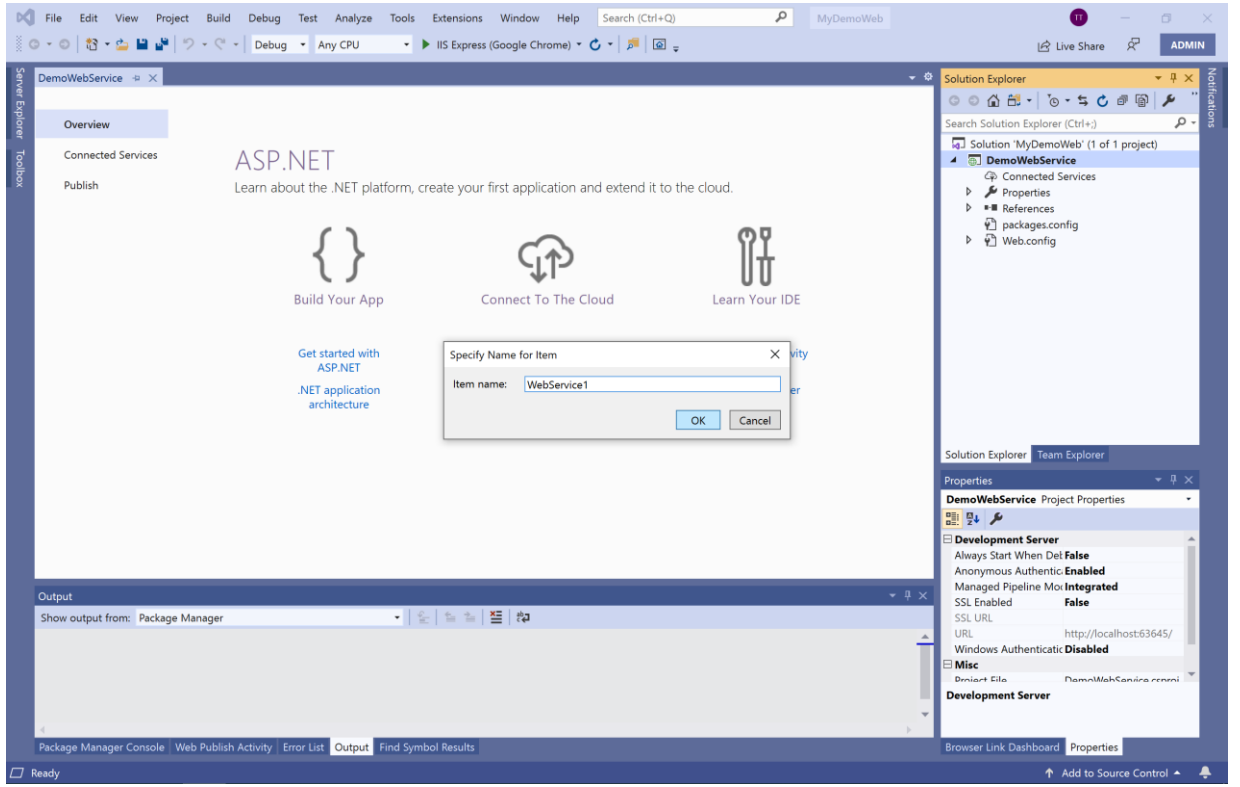




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

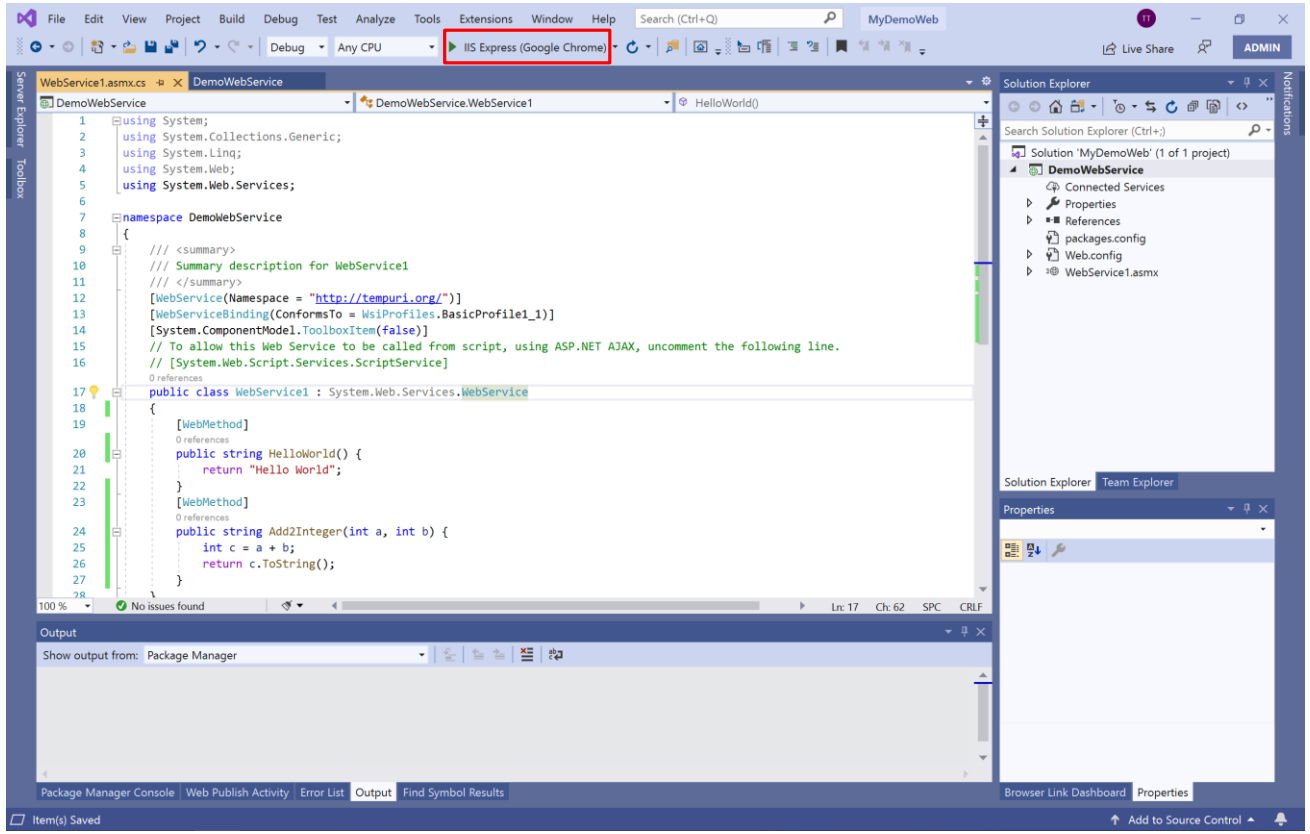
1. Create webservice application project

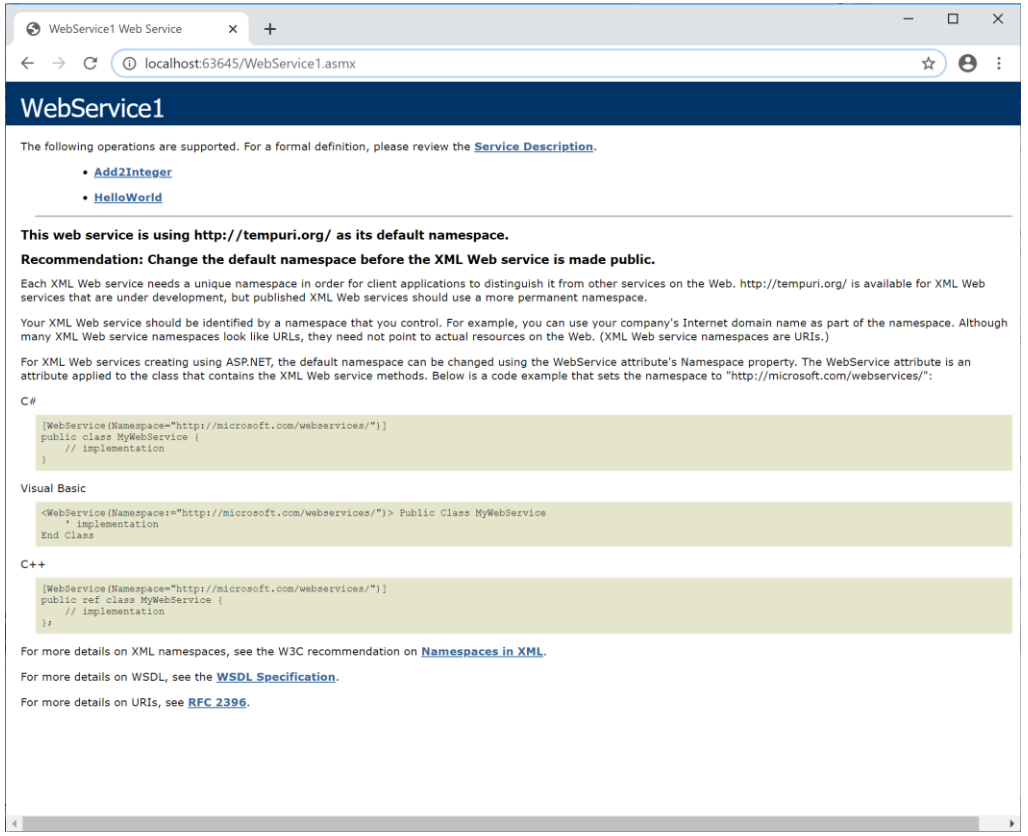




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

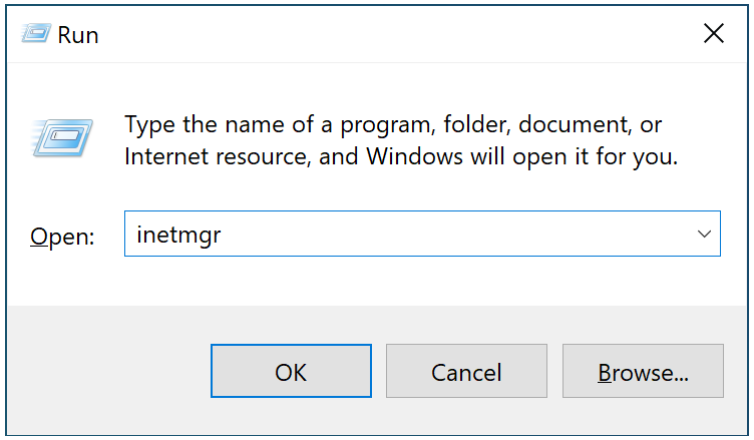
1. Create webservice application project

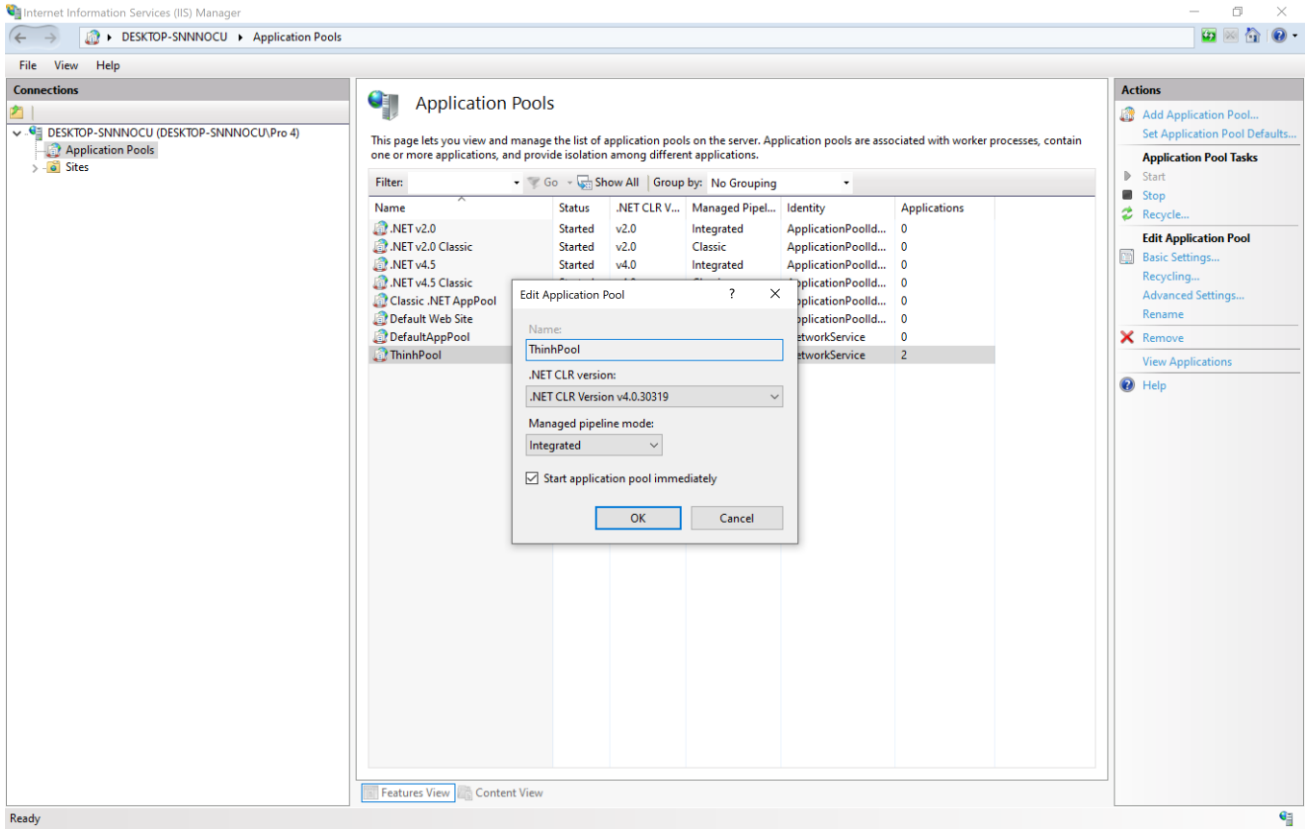




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

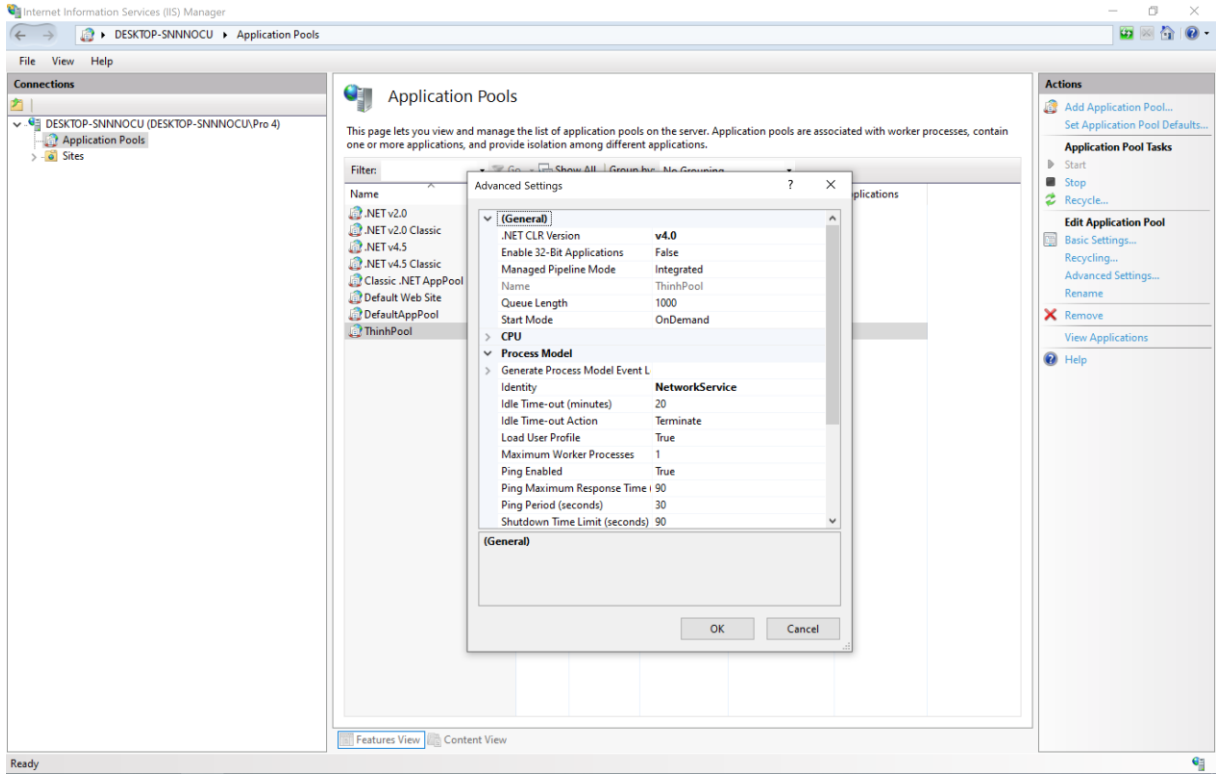
2. Deploy webservice application project in IIS

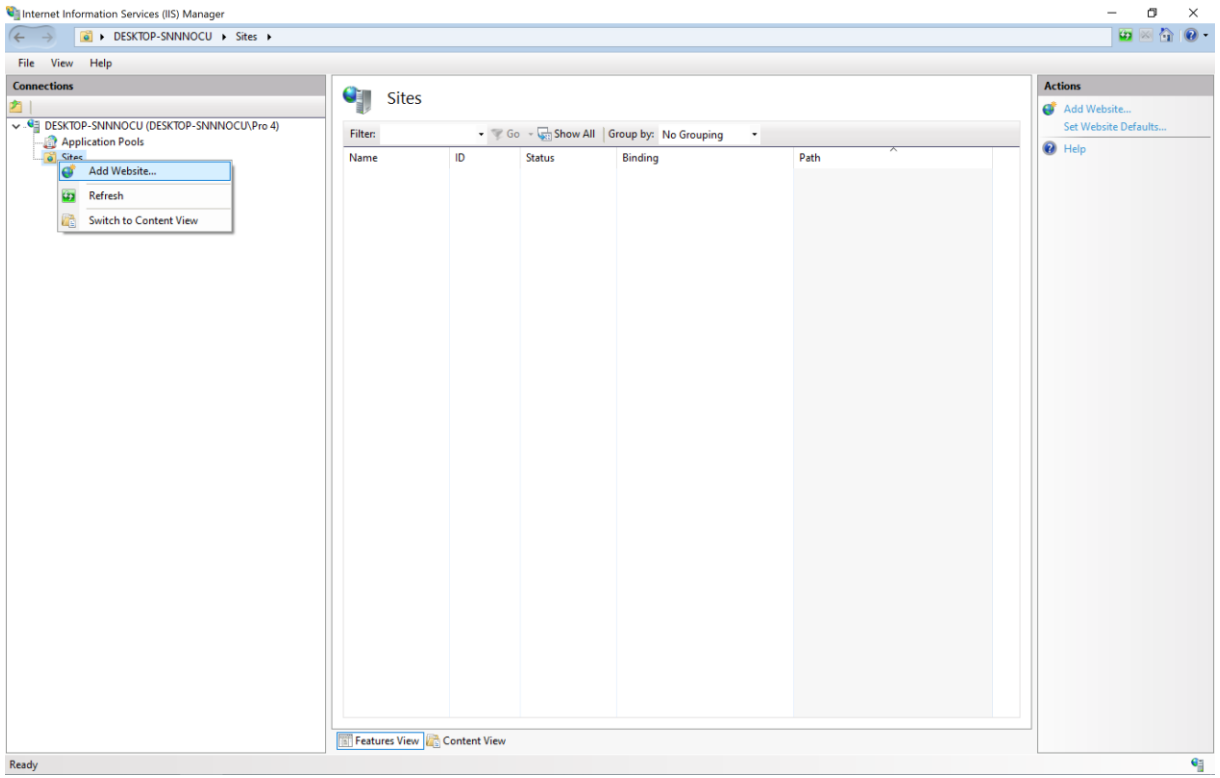




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

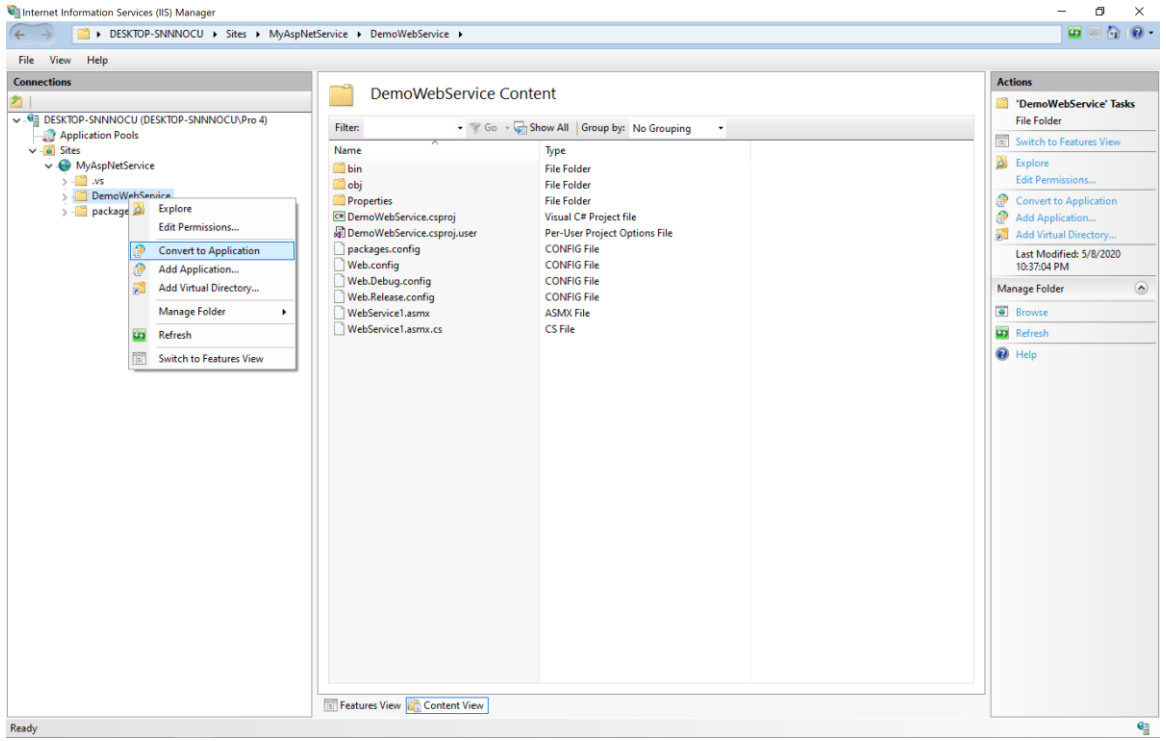
2. Deploy webservice application project in IIS

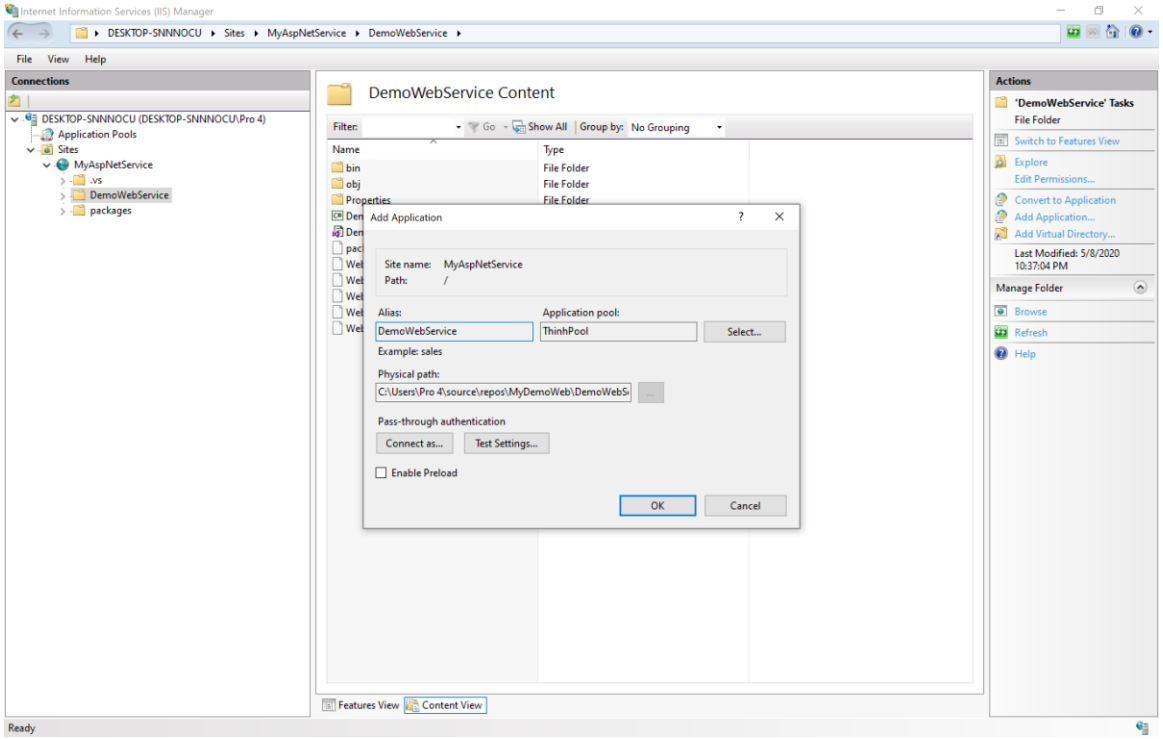




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

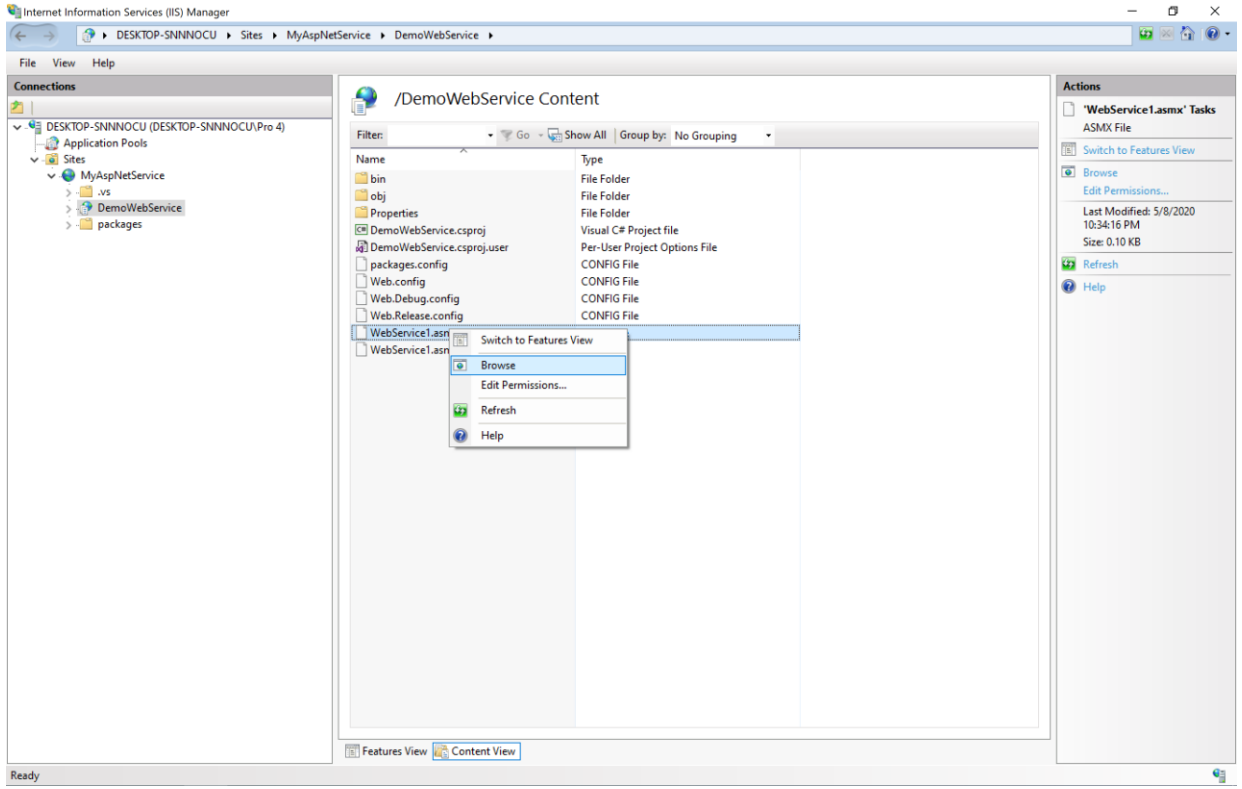
2. Deploy webservice application project in IIS

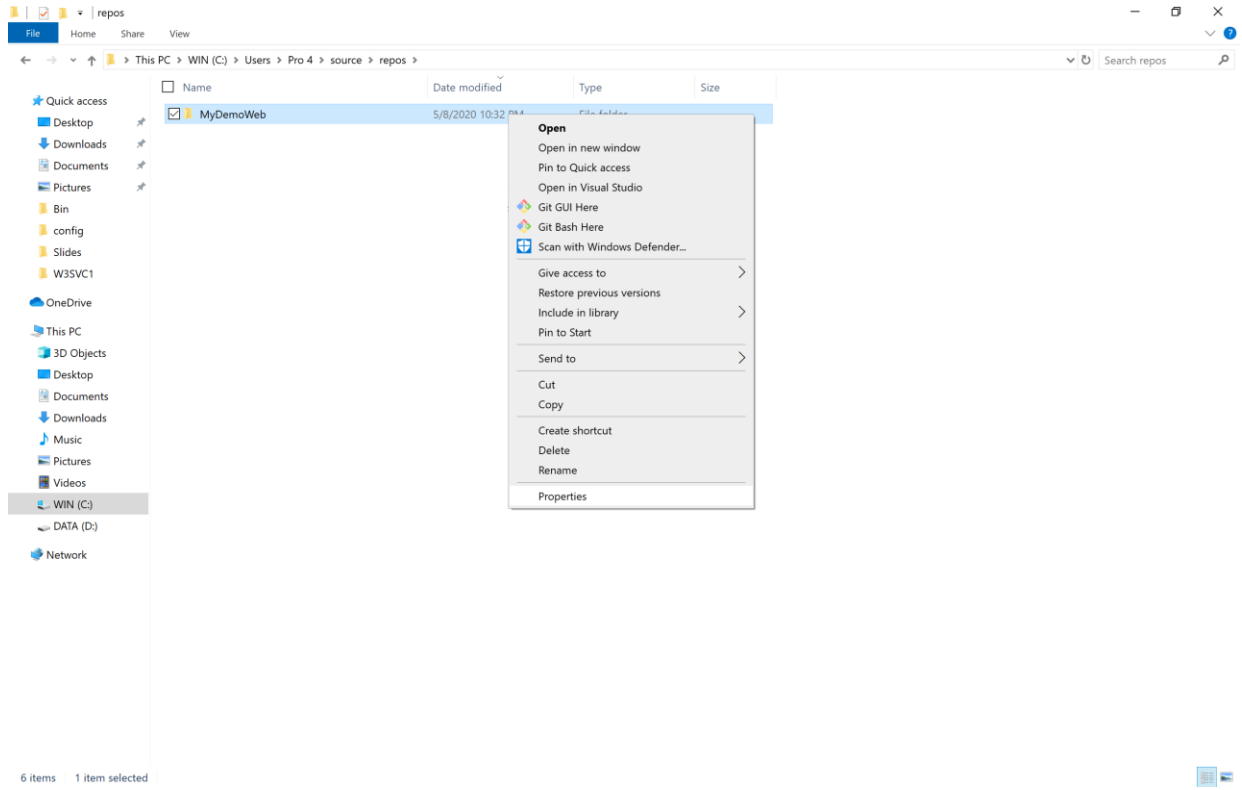




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

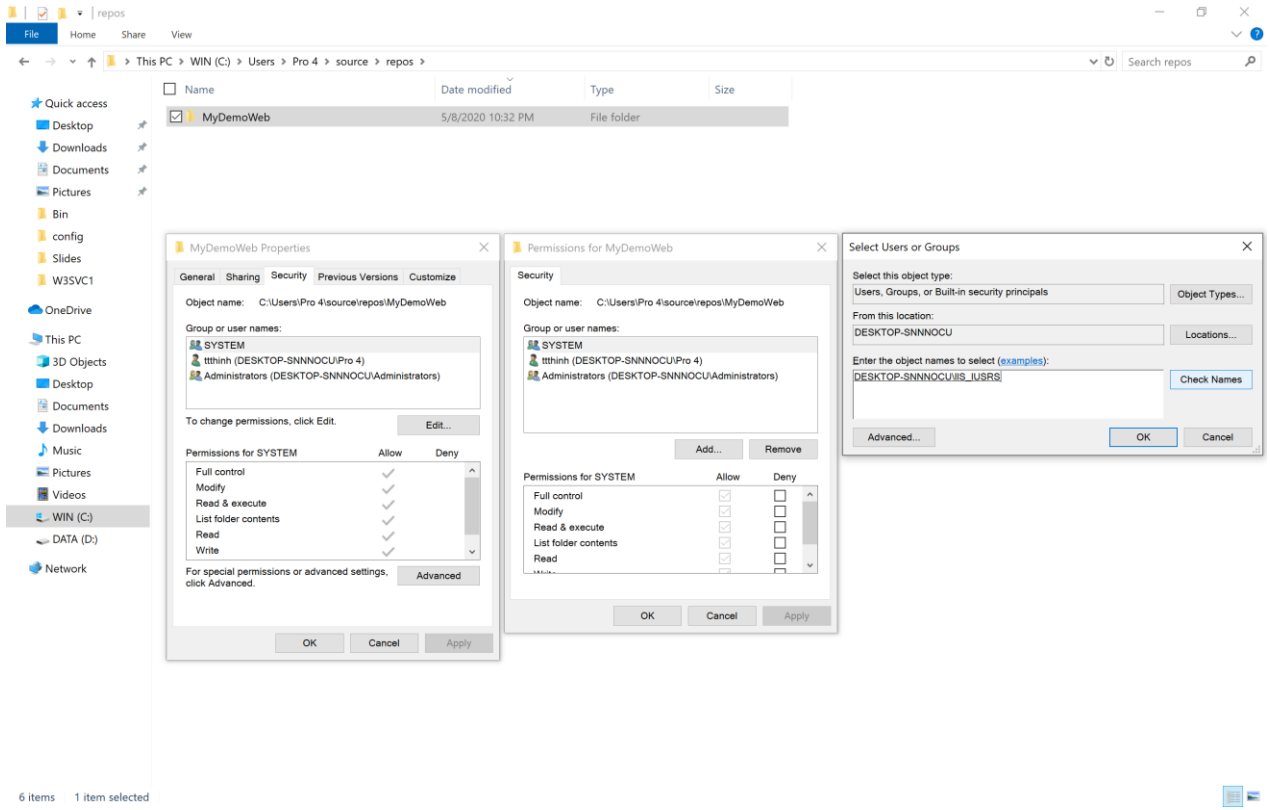
2. Deploy webservice application project in IIS

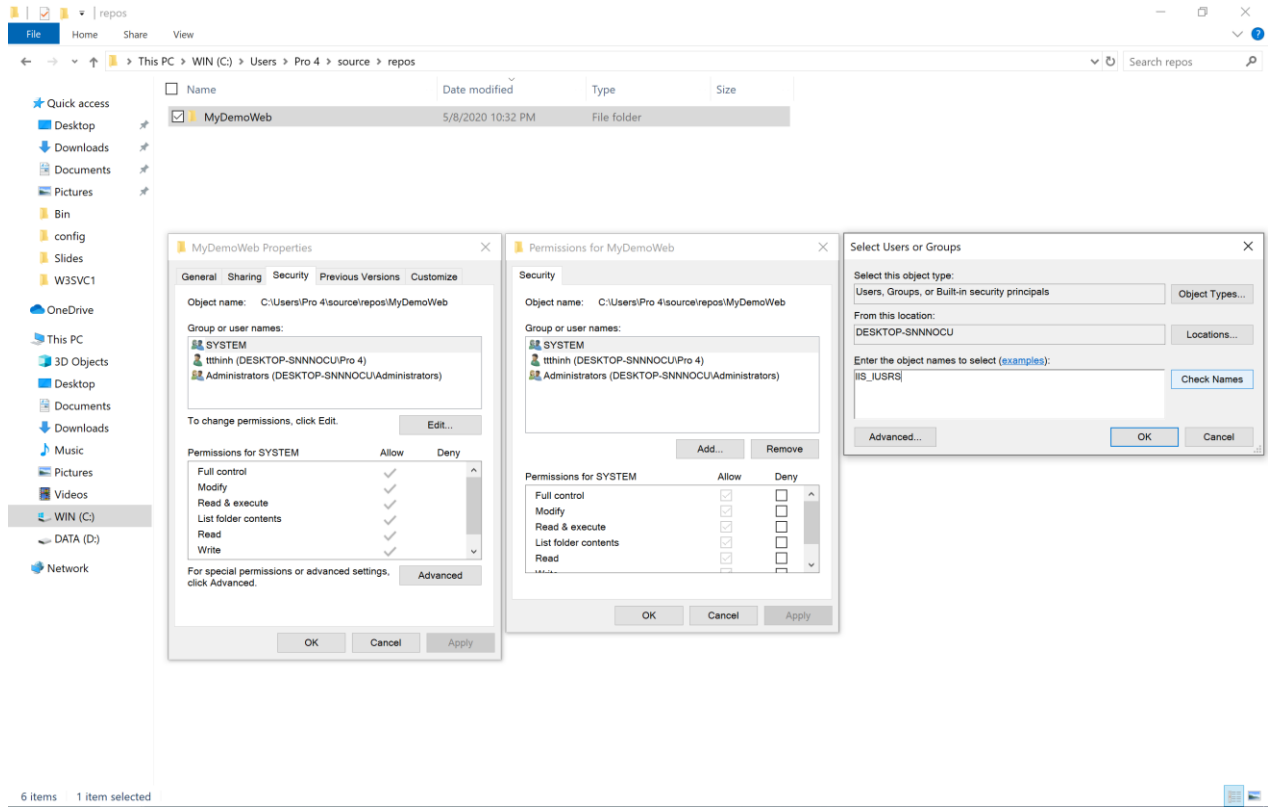




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

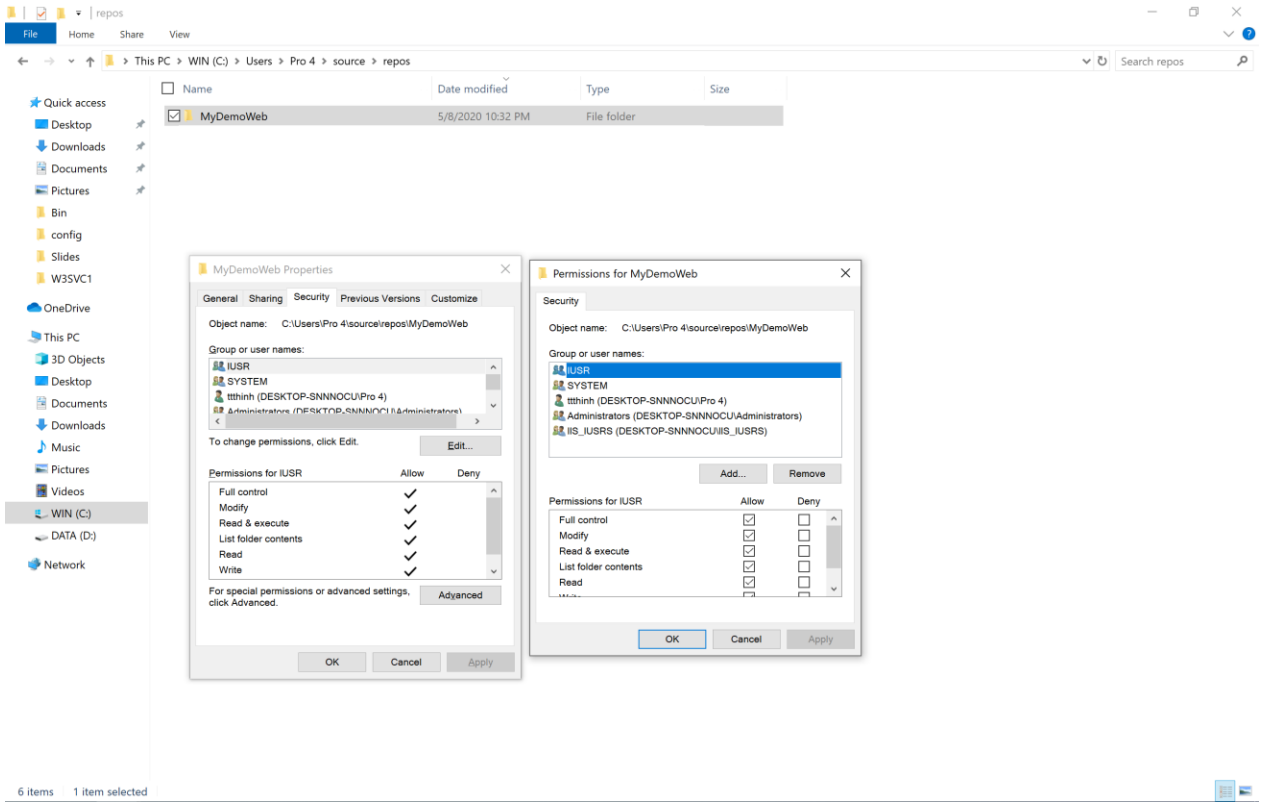
2. Deploy webservice application project in IIS

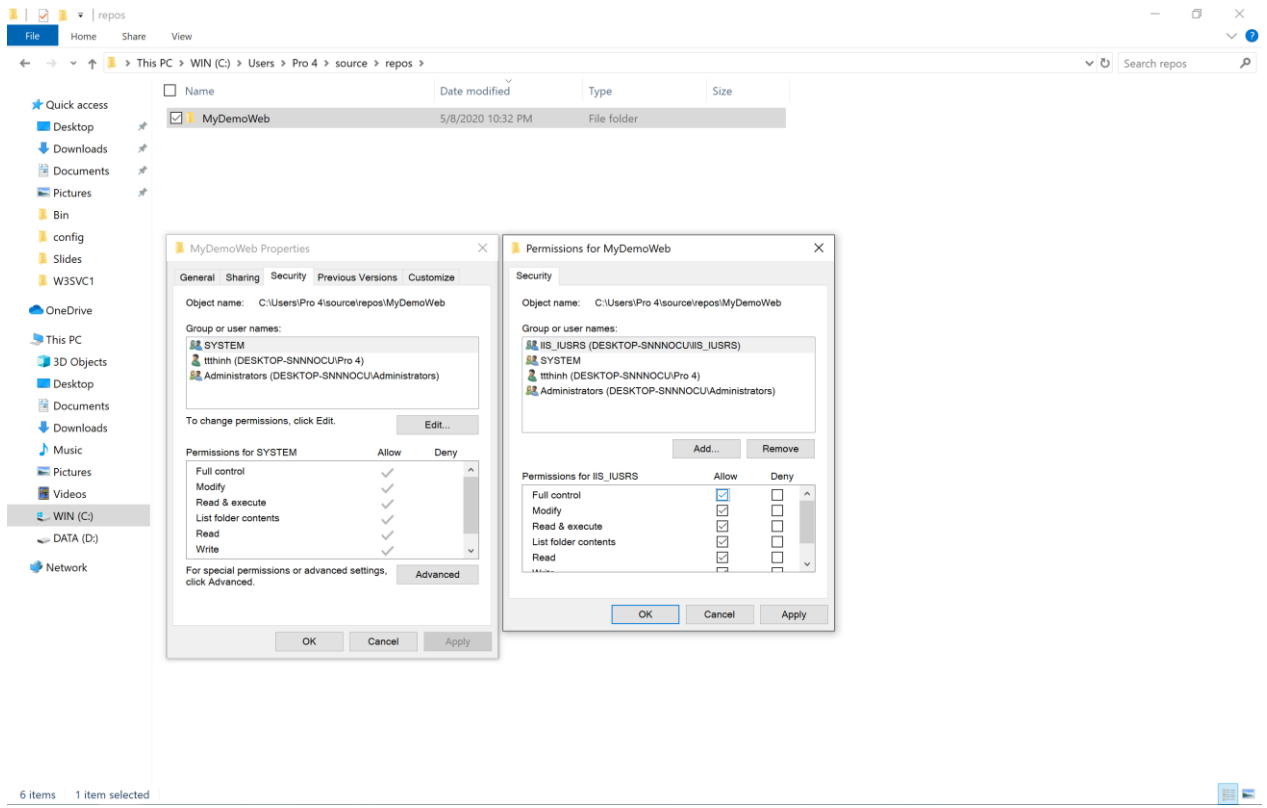




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

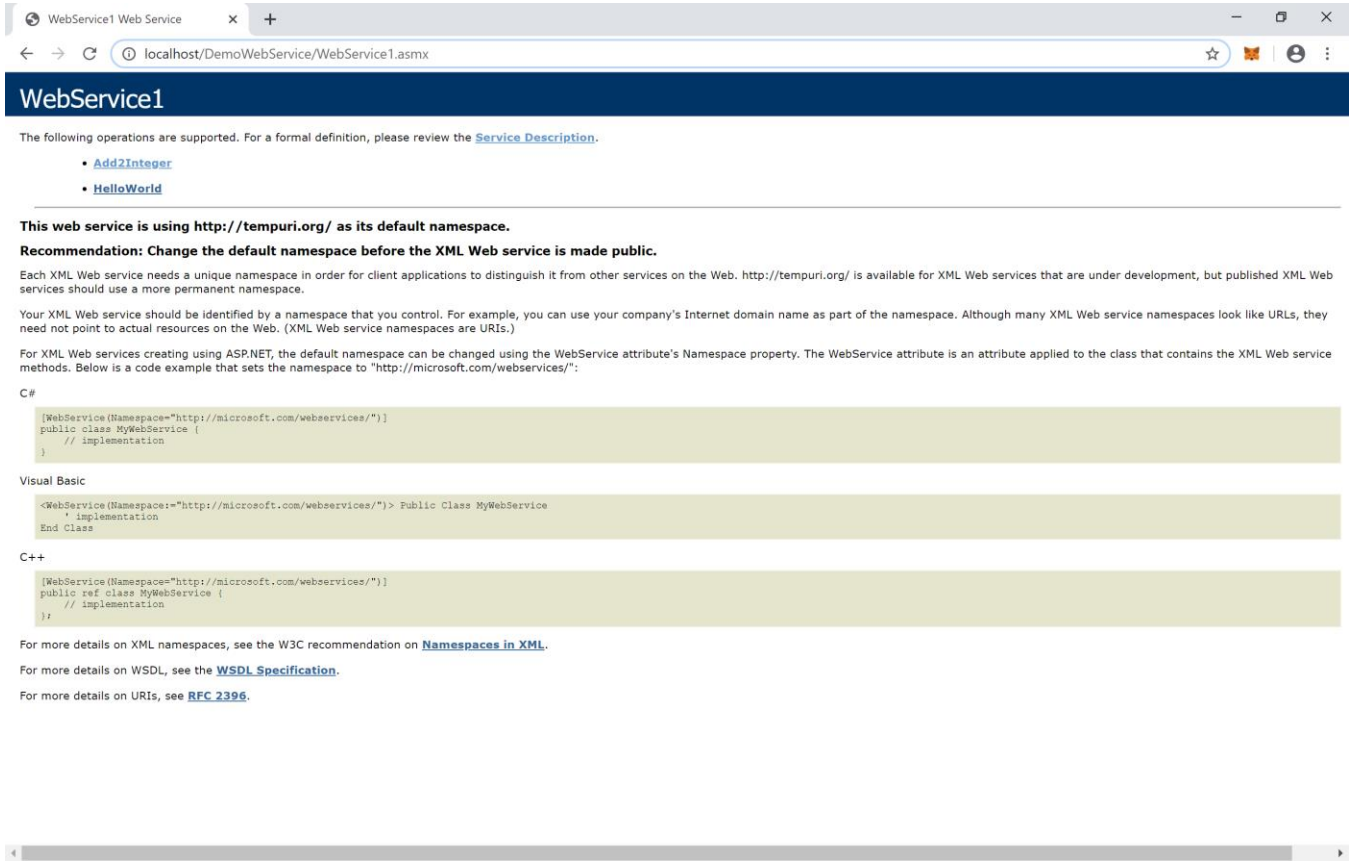
2. Deploy webservice application project in IIS





WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

3. Run webservice application project in IIS



WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

4. Create Android application (XML layout)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
        <TextView           android:id="@+id/txtResult"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:text="Hello World!"
                            app:layout_constraintBottom_toBottomOf="parent"
                            app:layout_constraintLeft_toLeftOf="parent"
                            app:layout_constraintRight_toRightOf="parent"
                            app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

4. Create Android application (MainActivity.java)

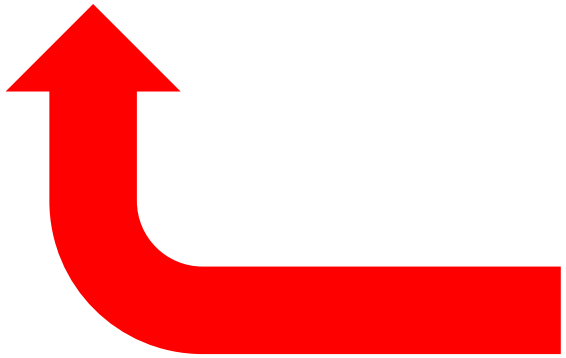
```
public class MainActivity extends Activity {
 TextView result;
 Handler handler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
   super.handleMessage(msg);
   result.setText((String) msg.obj);
  }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  result = (TextView) findViewById(R.id.txtResult);
  Thread slowJob = new Thread() {
   @Override
   public void run() {
    String resultValue = ""; HttpURLConnection con = null;
    try {
     String reqXML = "...";
     // Creating the HttpURLConnection object
     URL oURL = new URL("http://192.168.1.9/DemoWebService/WebService1.asmx");
     con = (HttpURLConnection) oURL.openConnection();
     con.setRequestProperty("Host", "localhost");
     con.setRequestProperty("Content-Type", "text/xml; charset=utf-8");
     con.setRequestProperty("Content-Length", String.valueOf(reqXML.length()));
     con.addRequestProperty("SOAPAction", "http://tempuri.org/Add2Integer");
     con.setRequestMethod("POST"); con.setDoInput(true); con.setDoOutput(true);
     //Xml message
     OutputStreamWriter writer = new OutputStreamWriter(con.getOutputStream(), StandardCharsets.UTF_8);
     writer.write(reqXML);
     writer.flush();
     if(con.getResponseCode() == HttpURLConnection.HTTP_OK) {
      byte[] byteBuf = new byte[1024];
      InputStream resStream = con.getInputStream();
      int resLen = 0, len = resStream.read(byteBuf);
      while (len > -1) {
       resLen += len; resultValue += new String(byteBuf); len = resStream.read(byteBuf);
      }
     }
    }
    catch (Exception ex){ resultValue = "\nERROR: " + ex.getMessage(); }
    // send message to handler so it updates GUI
    Message msg = handler.obtainMessage(); msg.obj = (String) resultValue; handler.sendMessage(msg);
    if(con != null) con.disconnect();
   }
  };
  slowJob.start();
 }
}
```

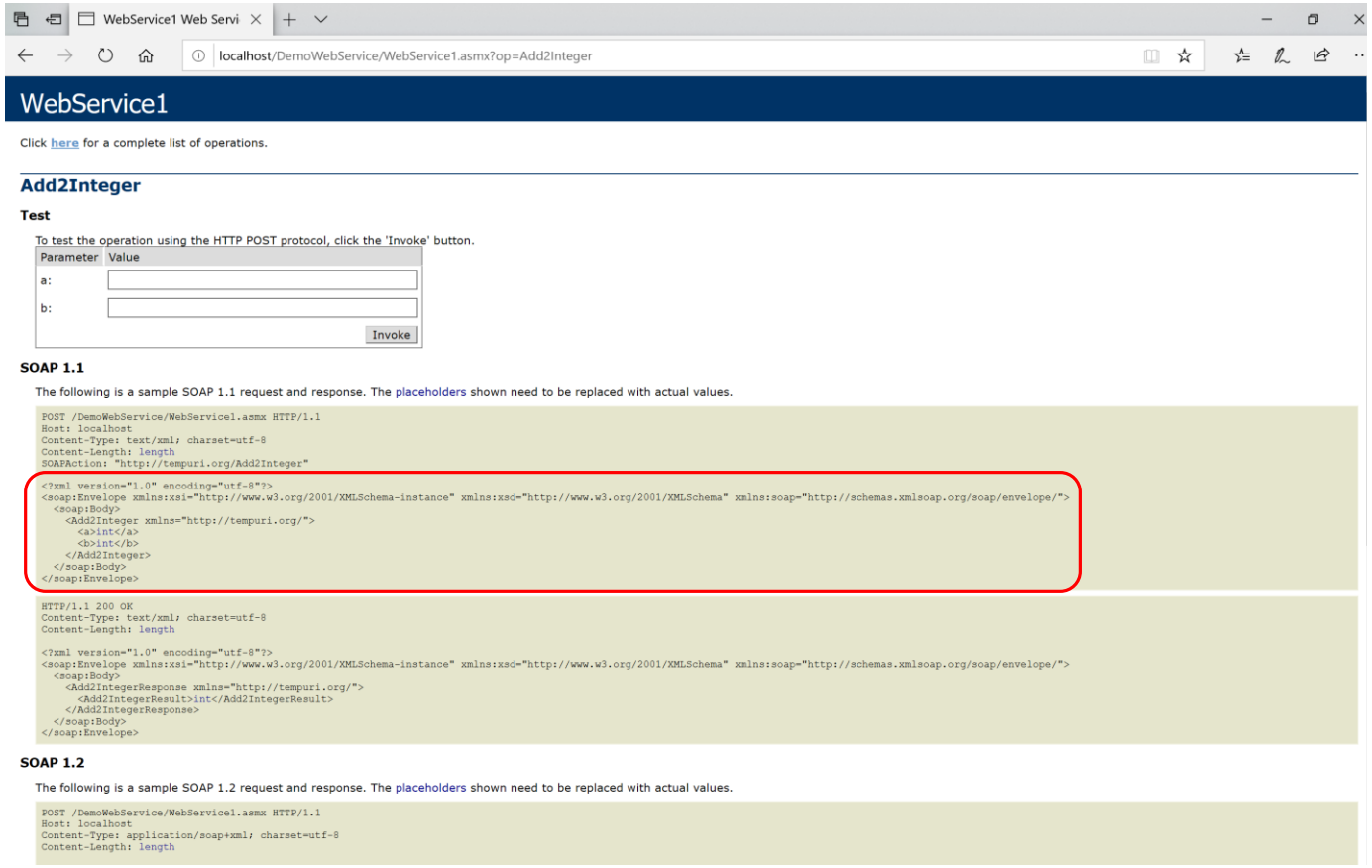
note

WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

4. Create Android application (MainActivity.java)

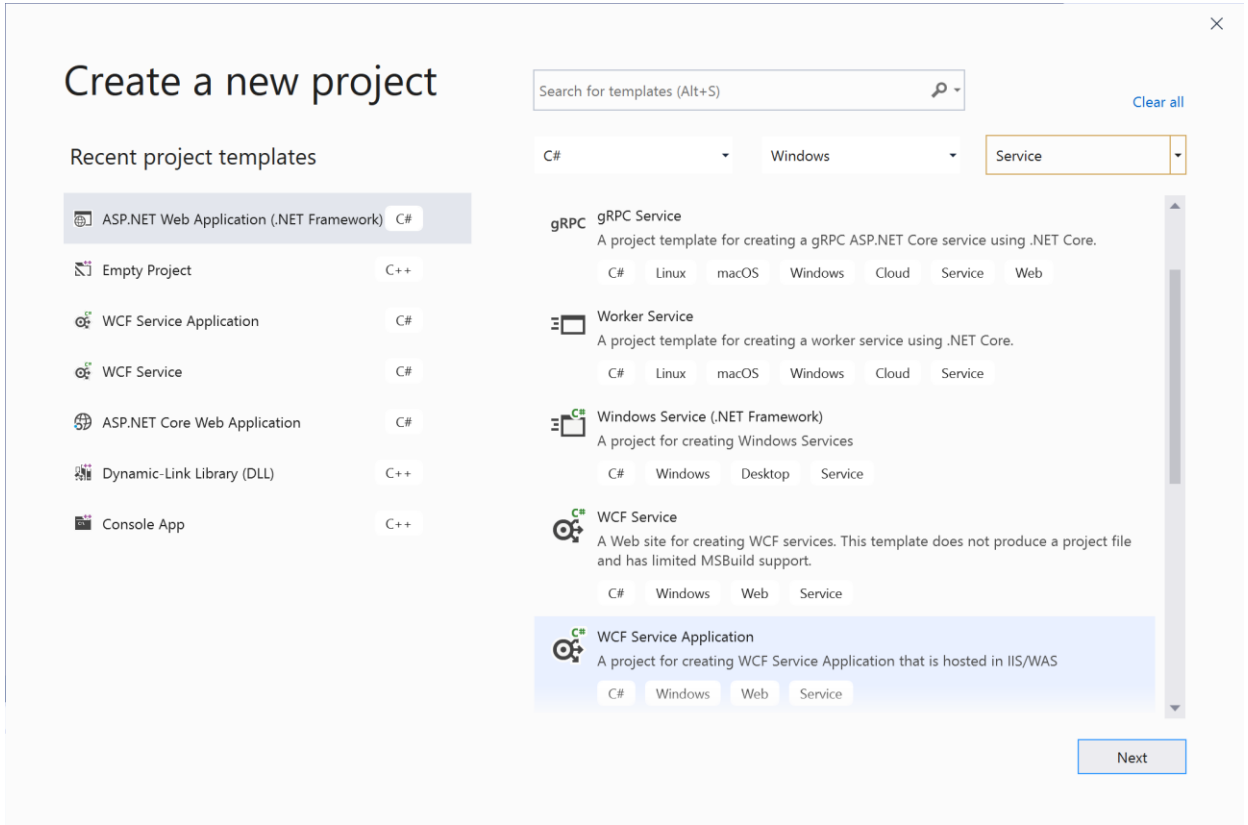
String reqXML = “…” + “<a>” +
5 + “” + 4 + “…”

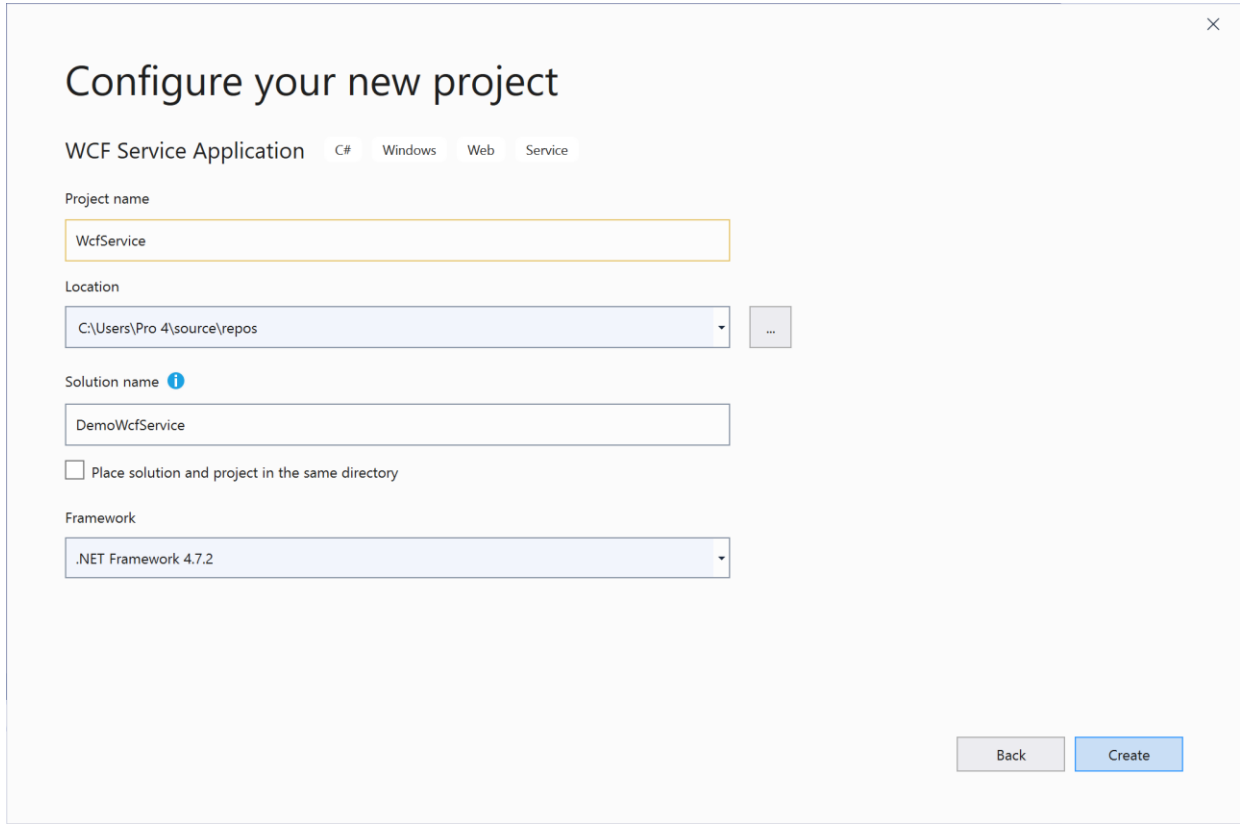




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

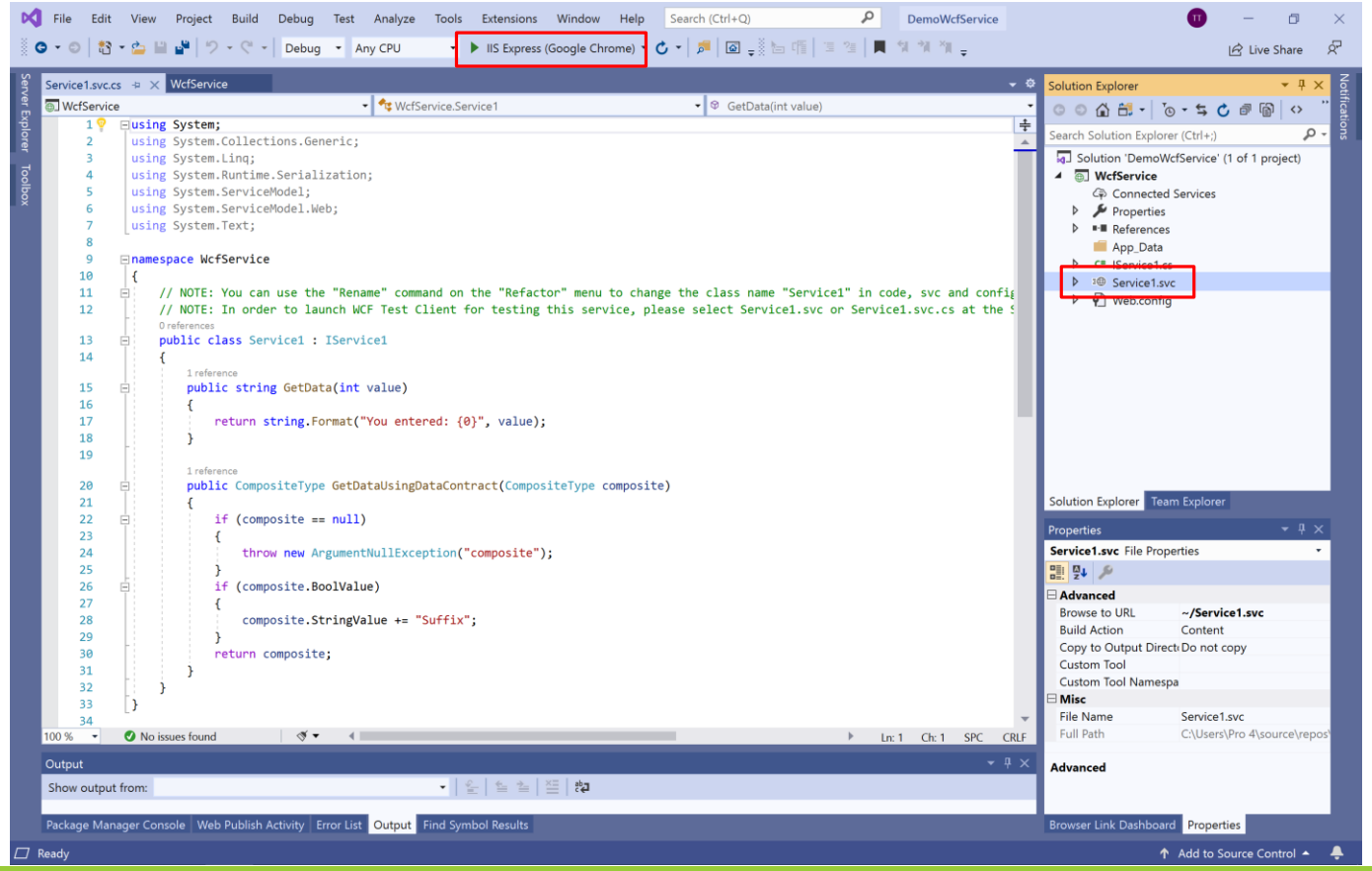
1. Create wcf-service application project





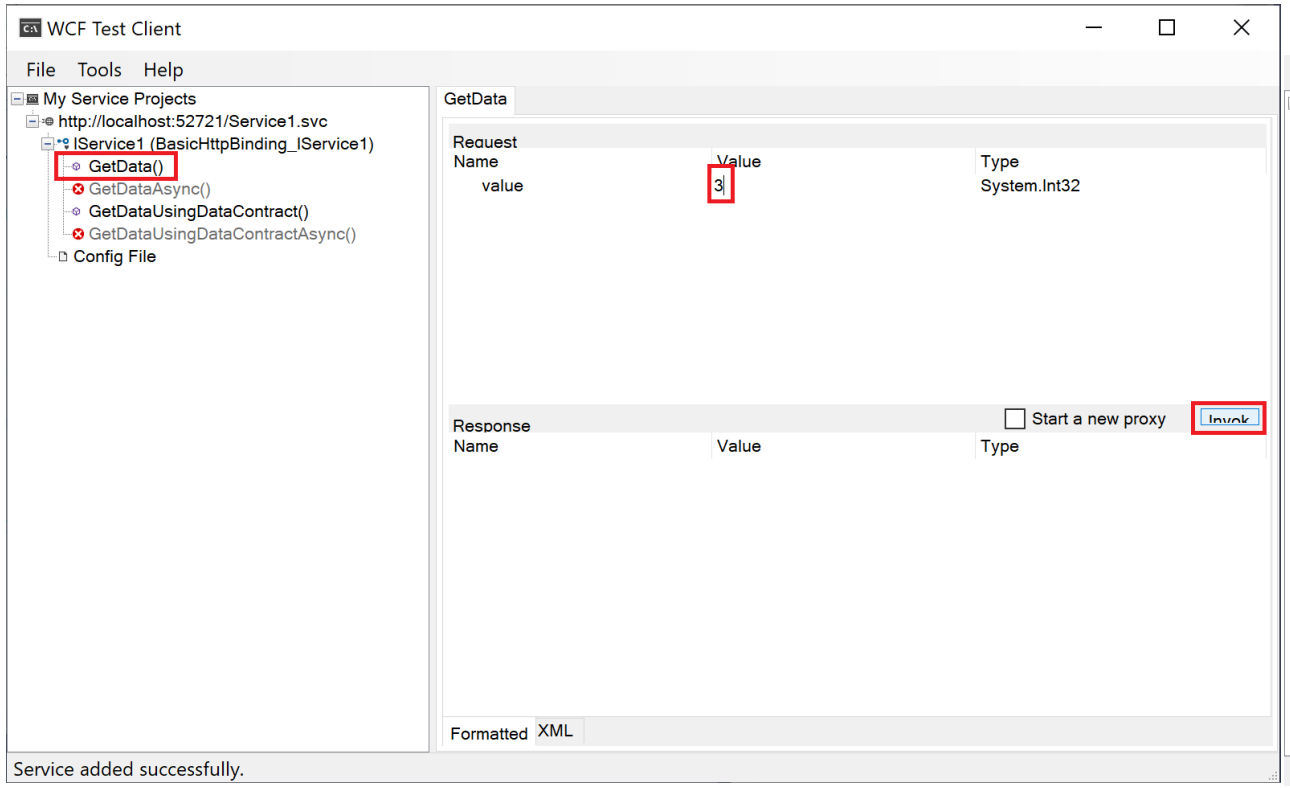
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

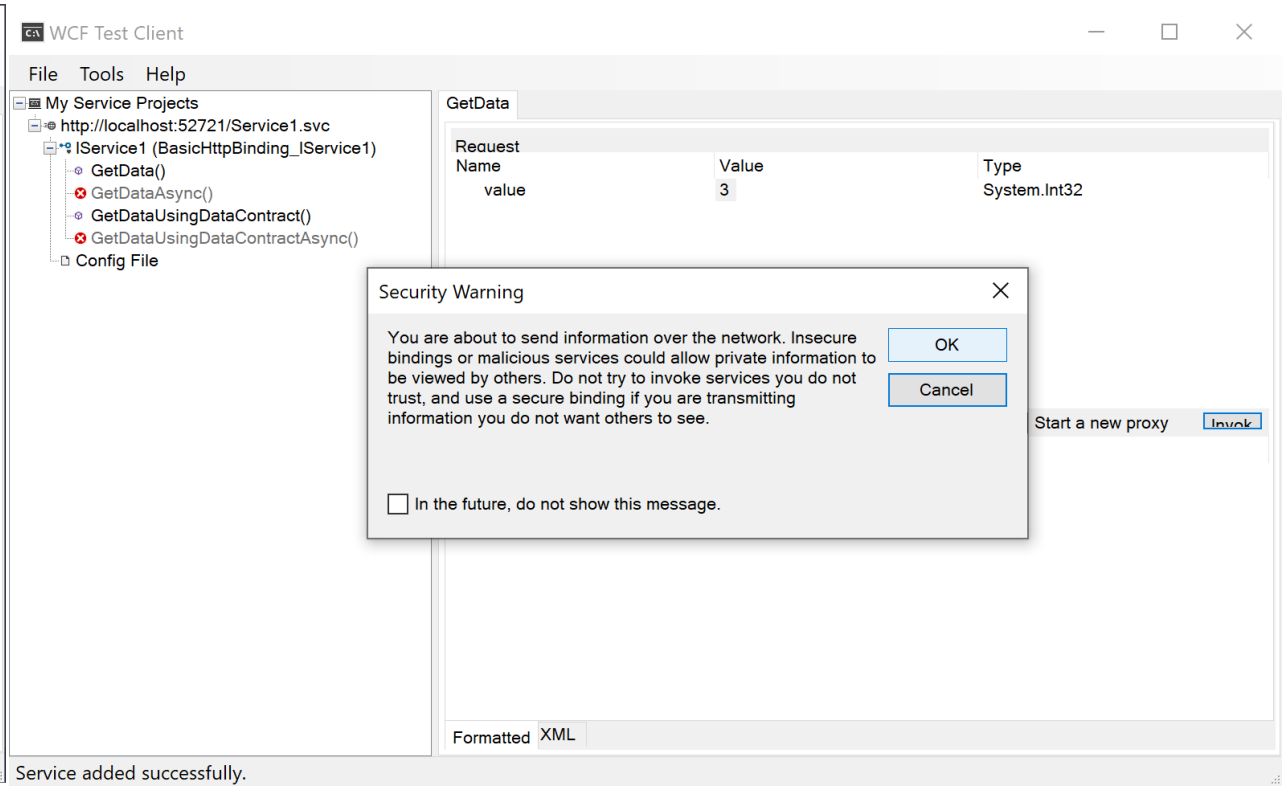
1. Create wcf-service application project



WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

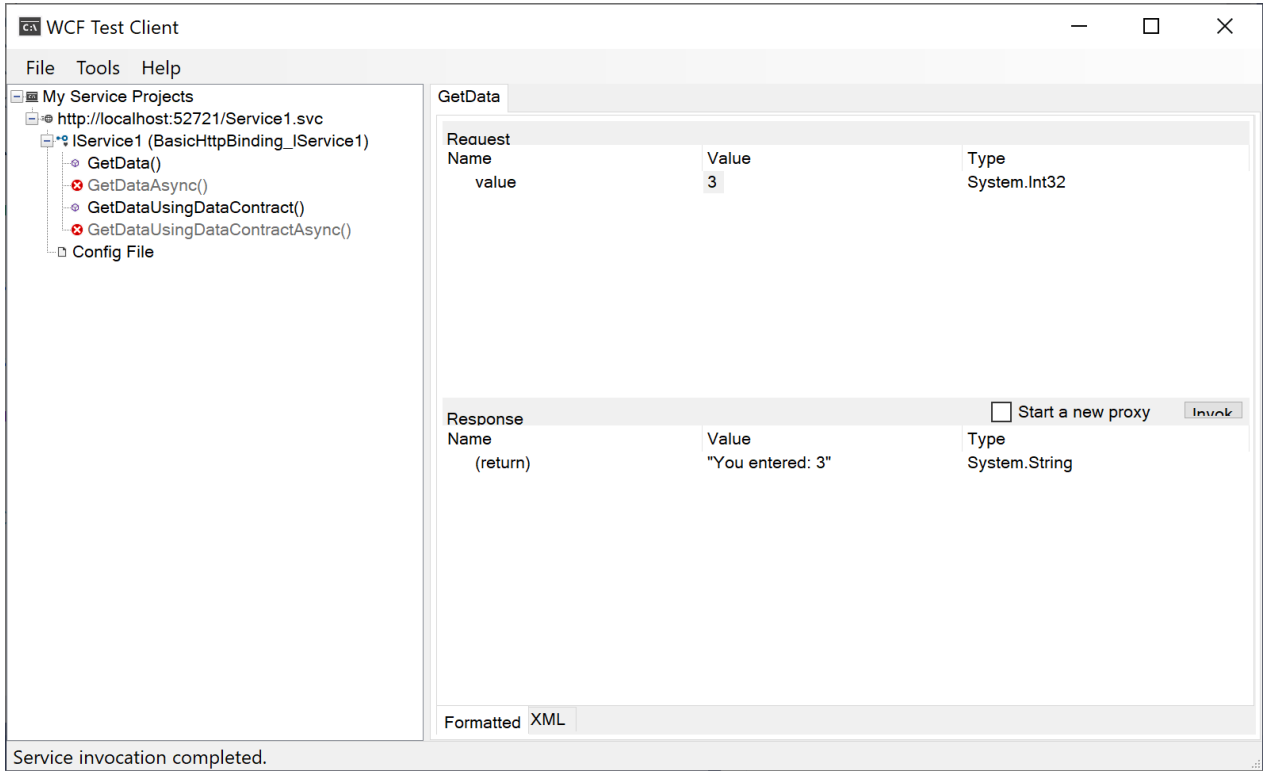
1. Create wcf-service application project

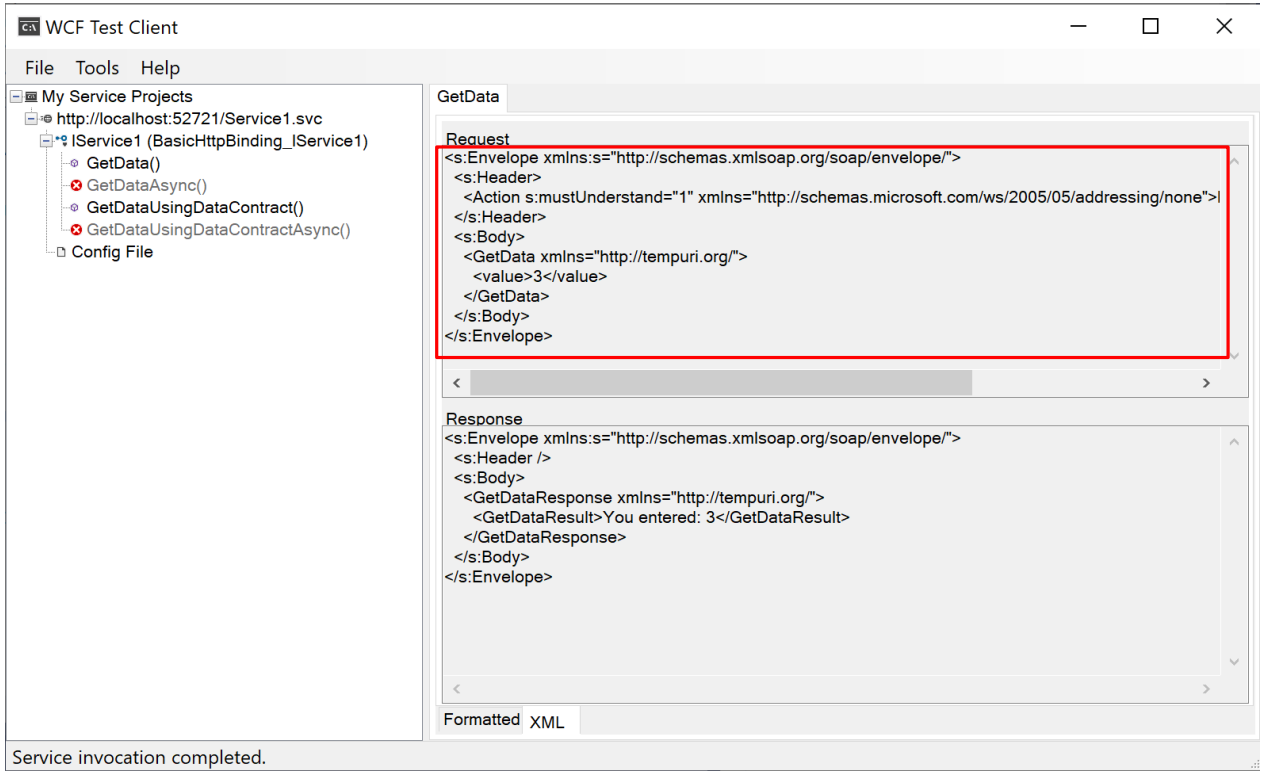




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

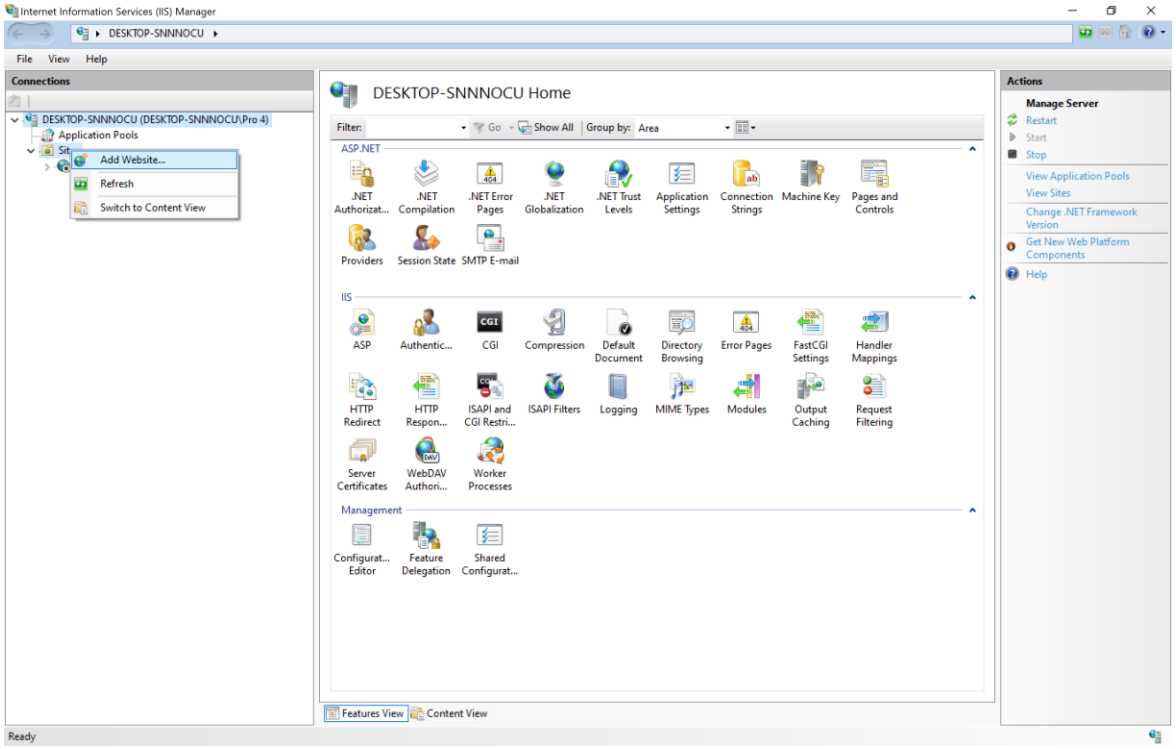
1. Create wcf-service application project

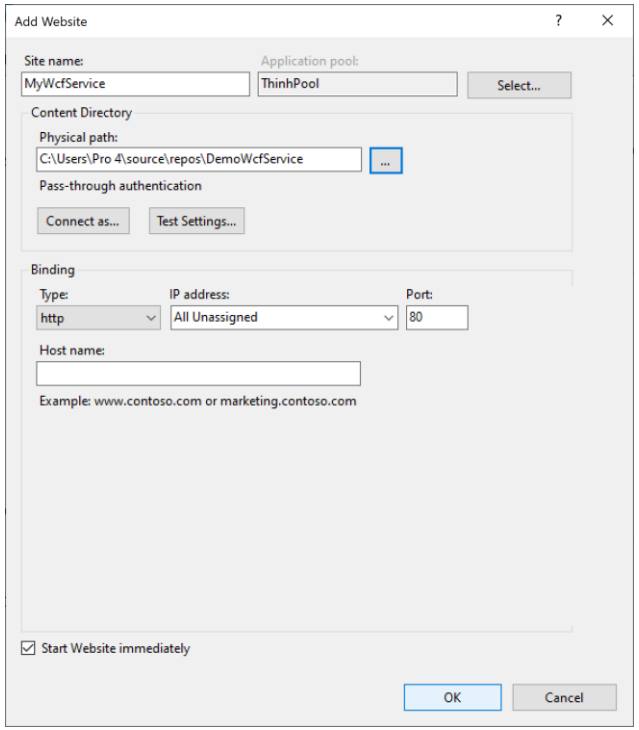


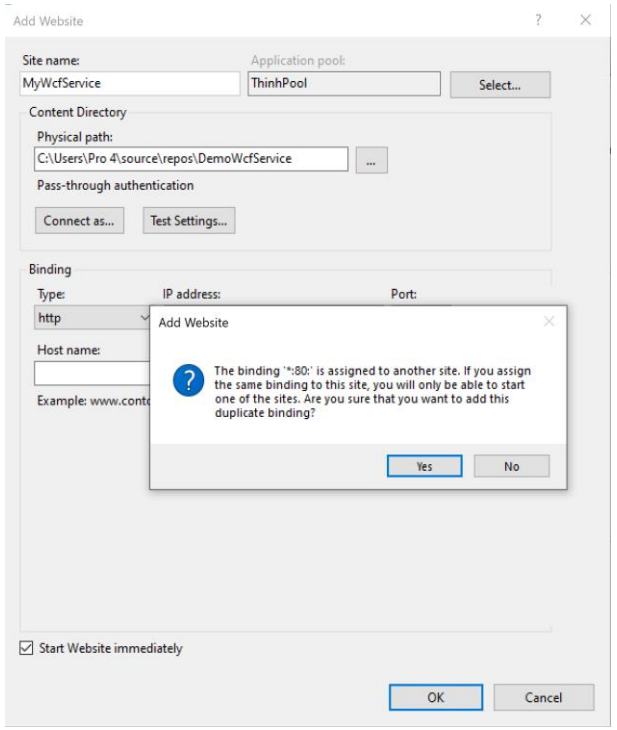


WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

2. Deploy wcf-service application project in IIS

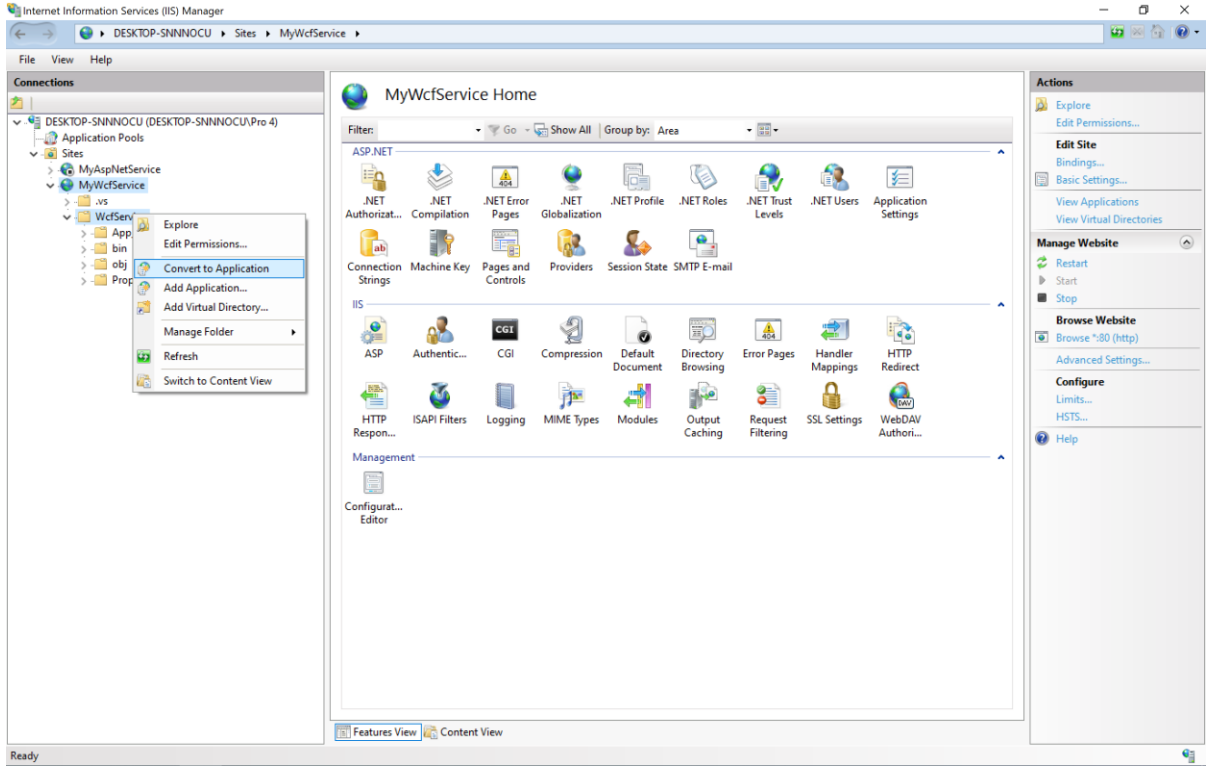


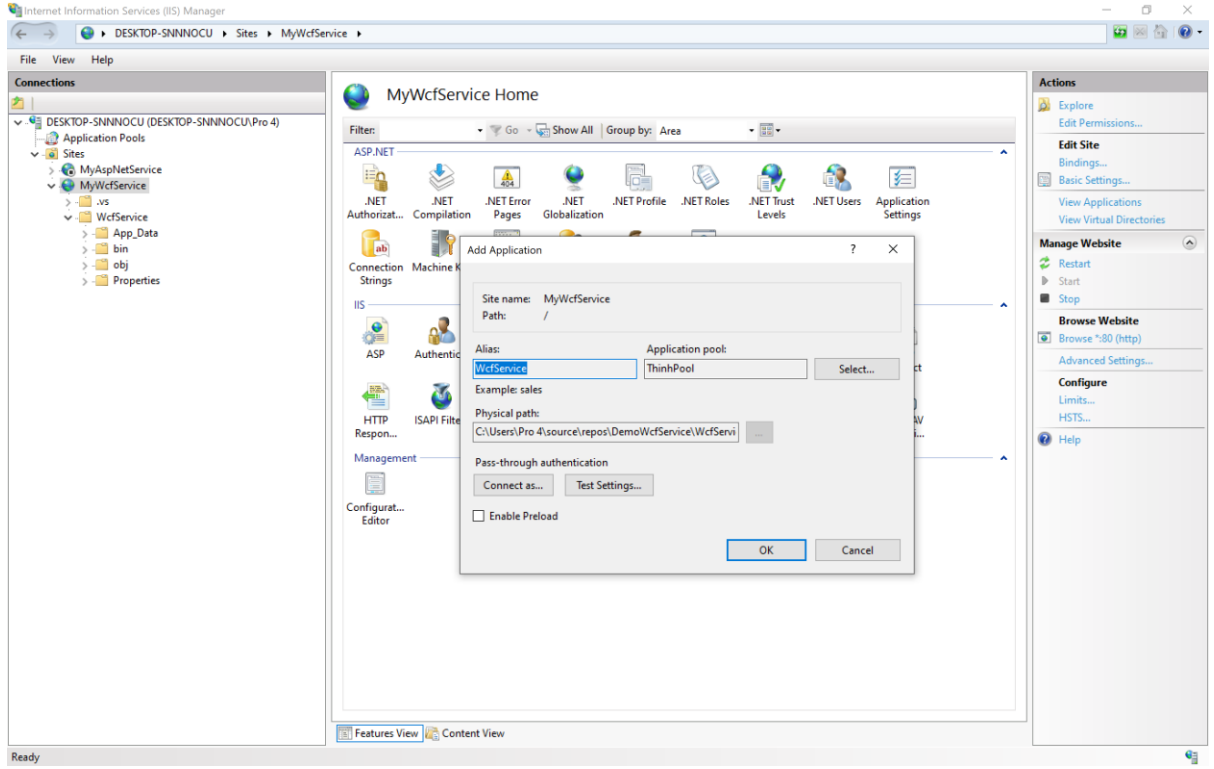




WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

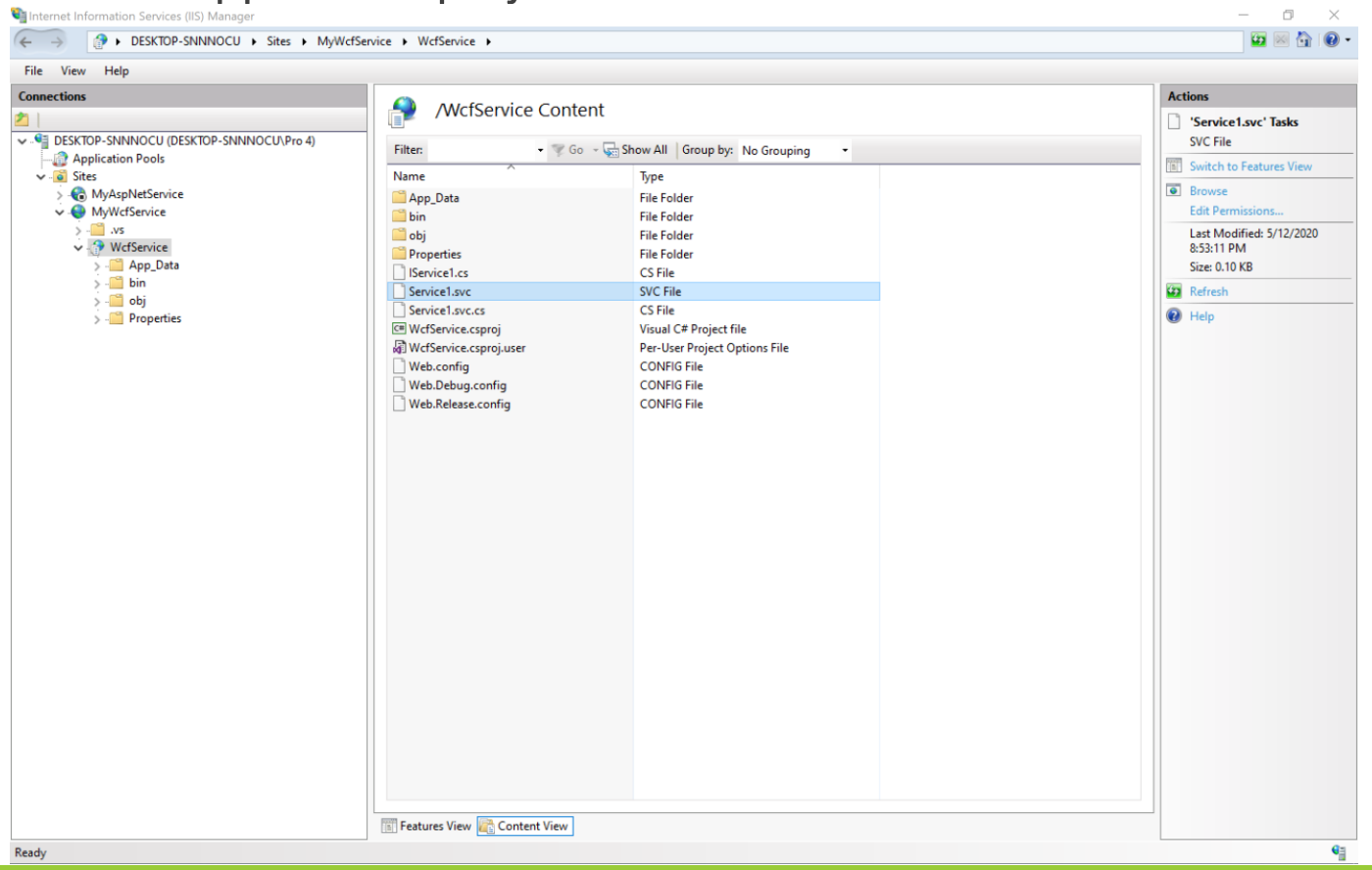
2. Deploy wcf-service application project in IIS





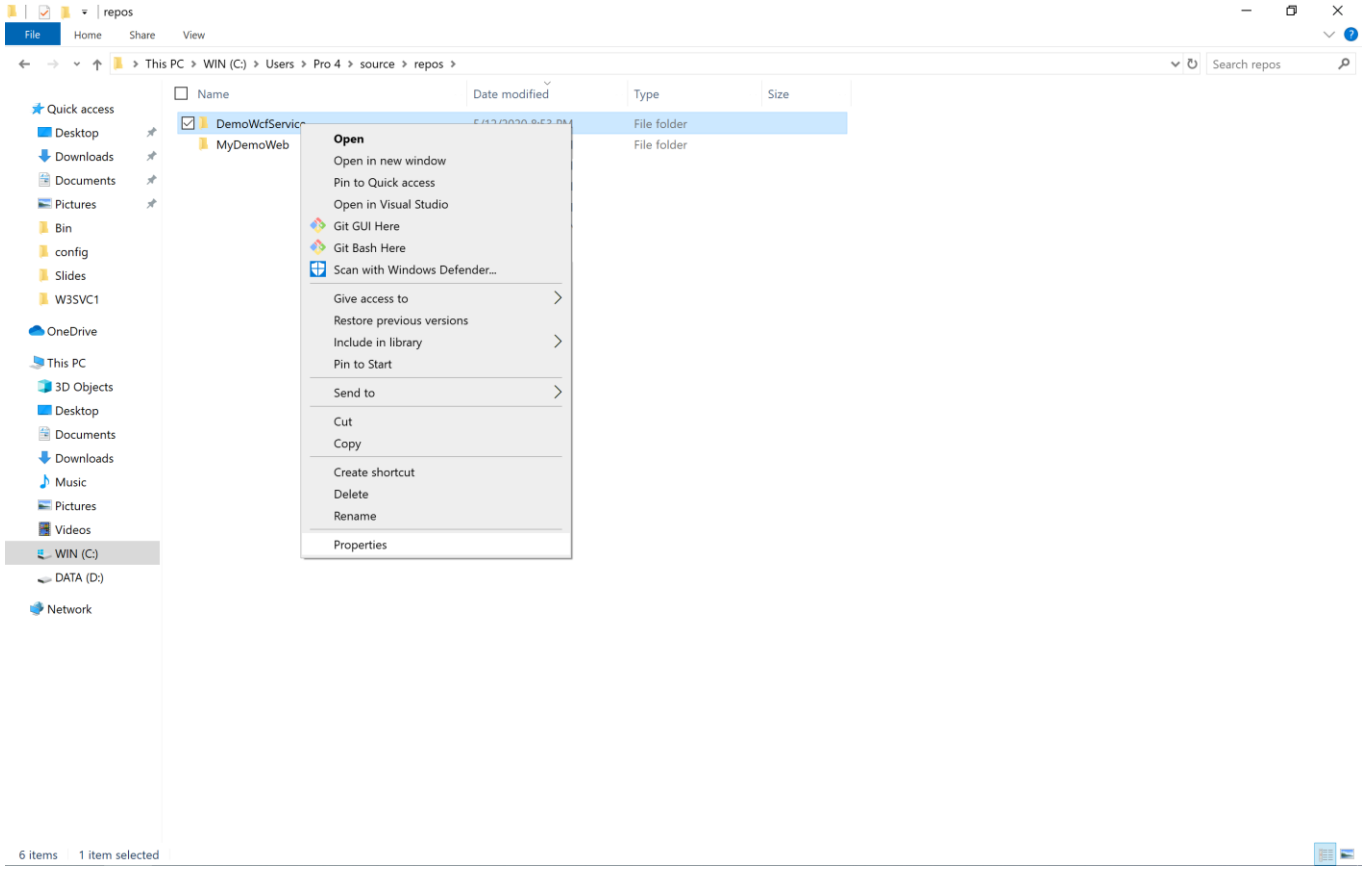
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

2. Deploy wcf-service application project in IIS



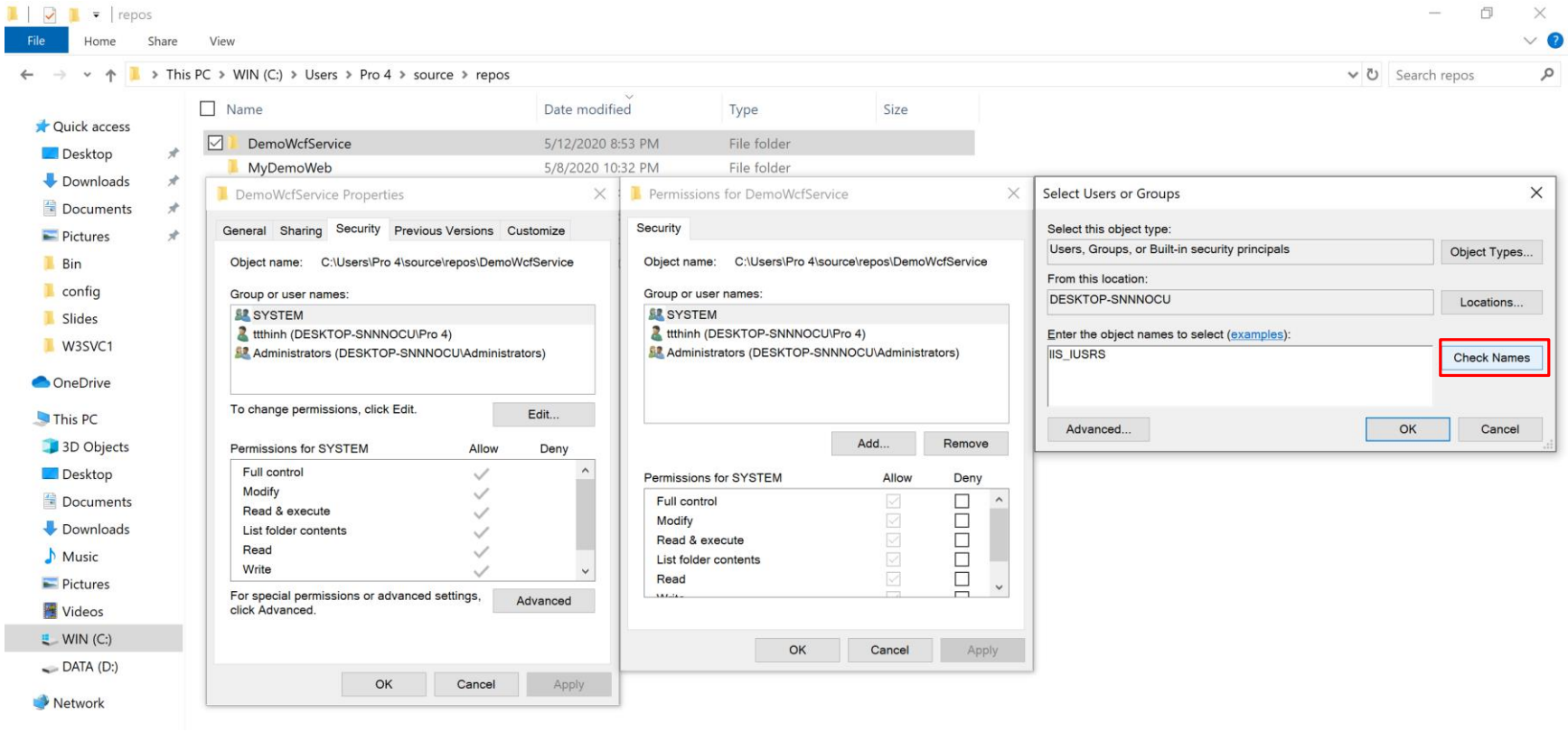
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

2. Deploy wcf-service application project in IIS



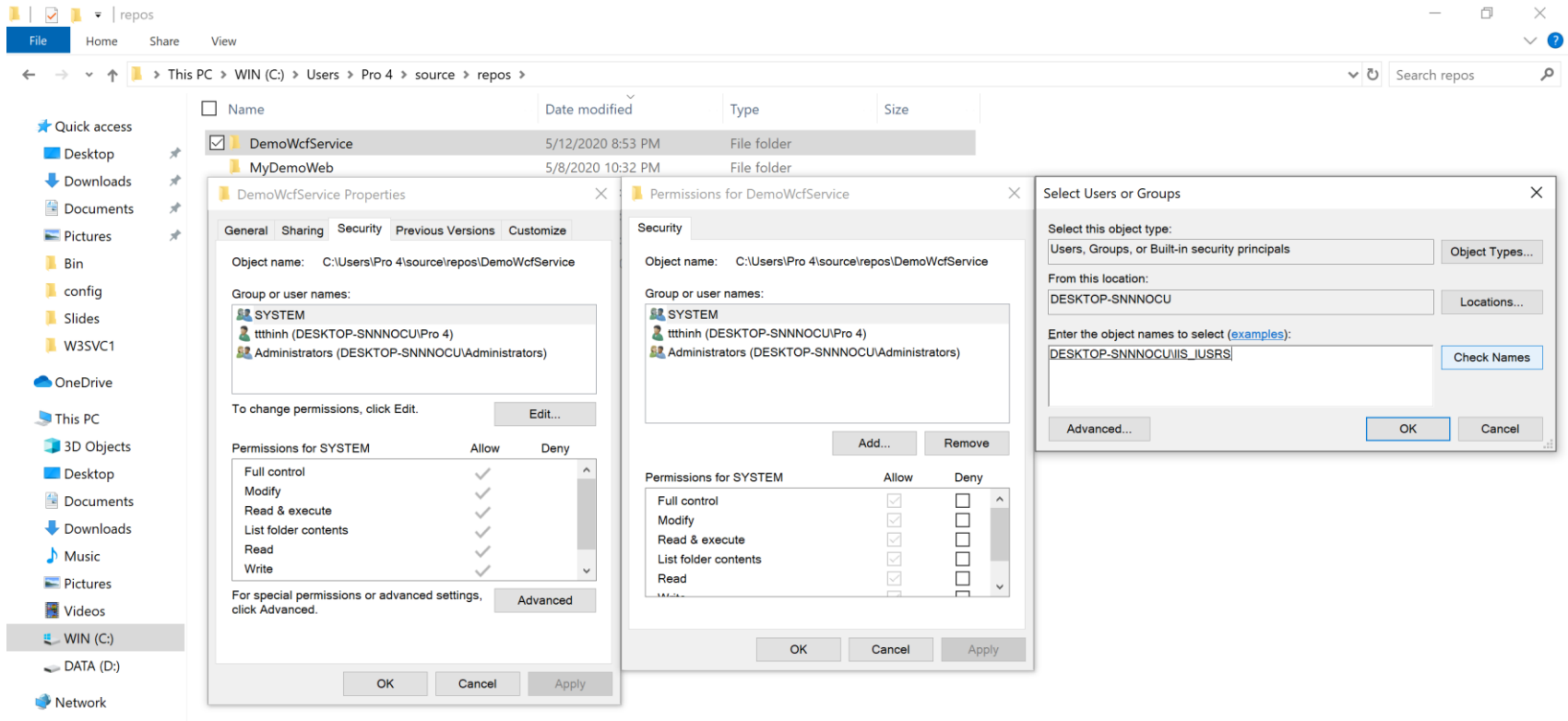
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

2. Deploy wcf-service application project in IIS



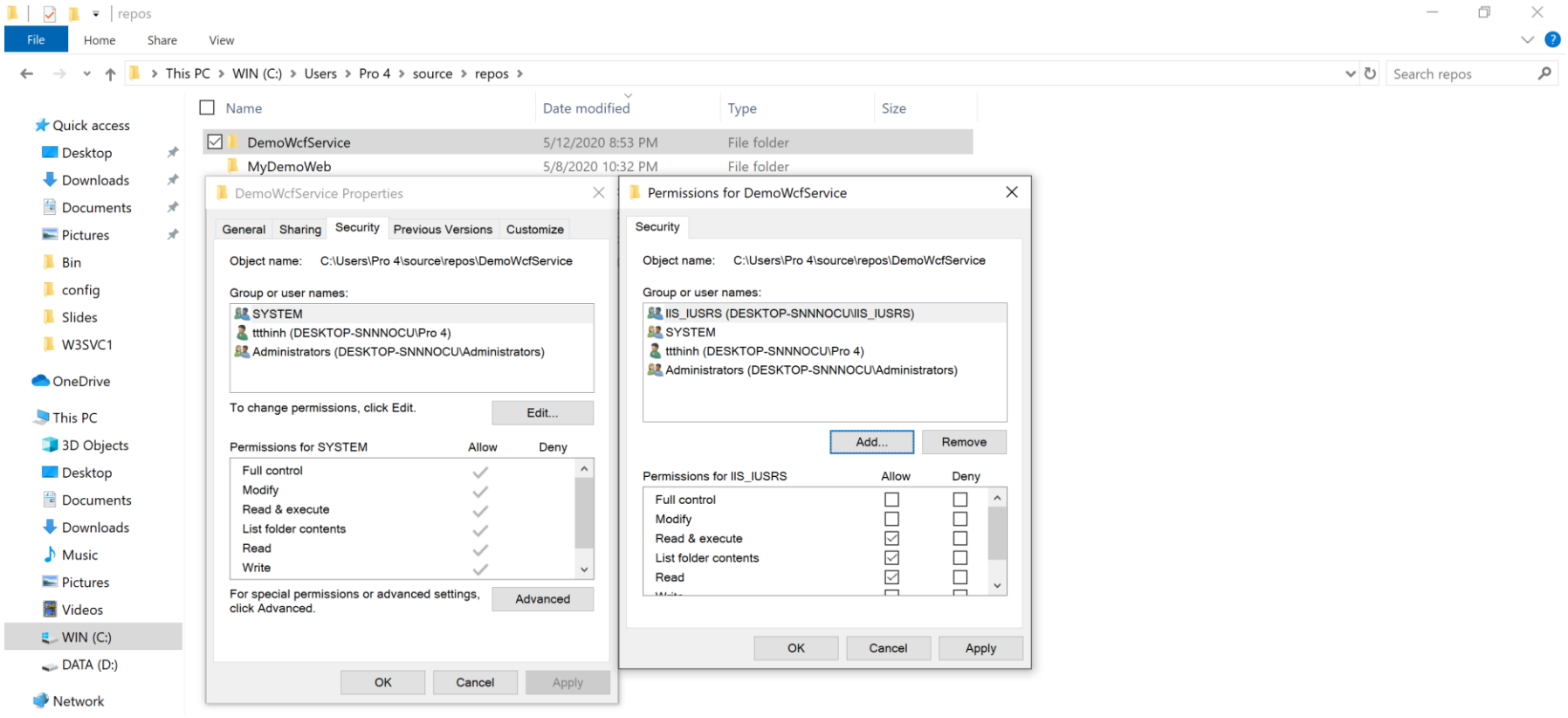
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

2. Deploy wcf-service application project in IIS



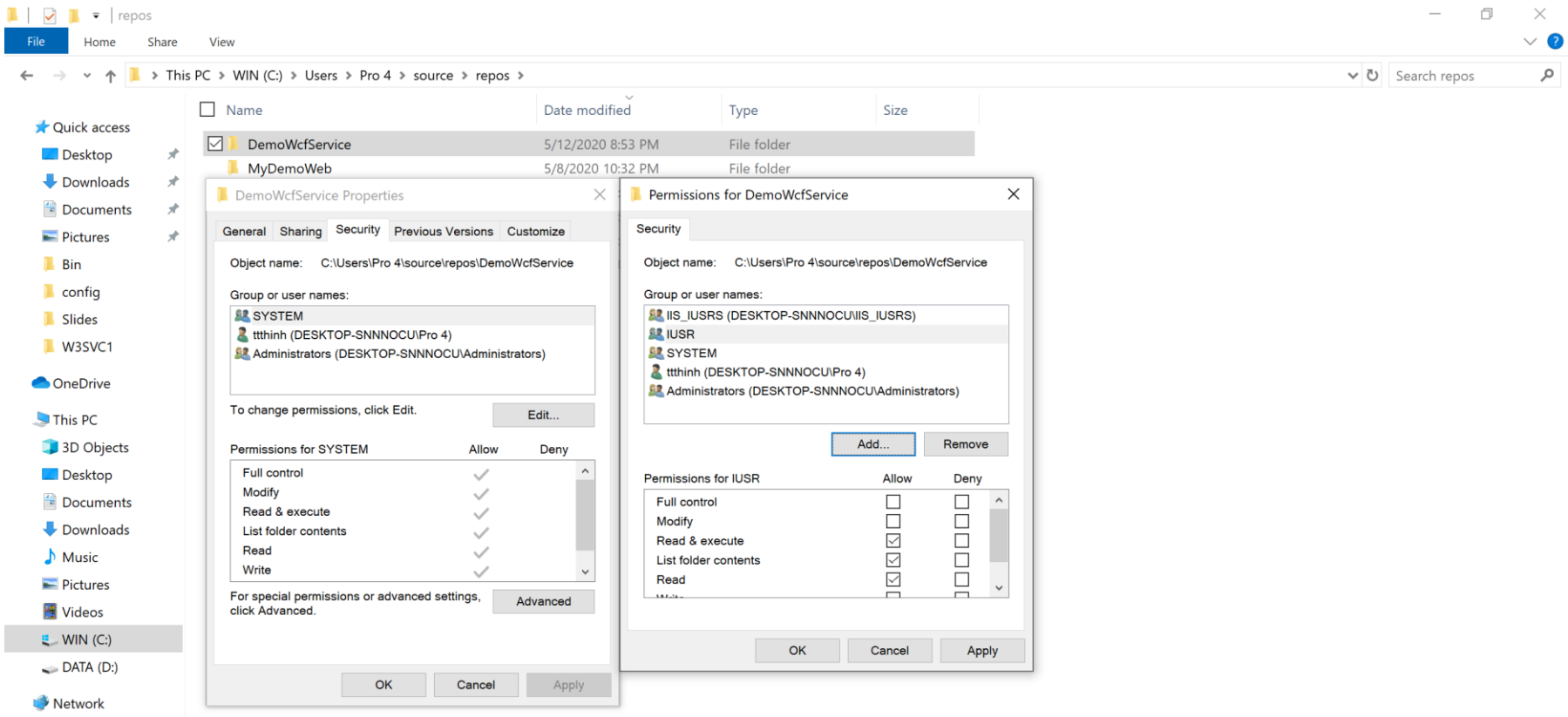
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

2. Deploy wcf-service application project in IIS



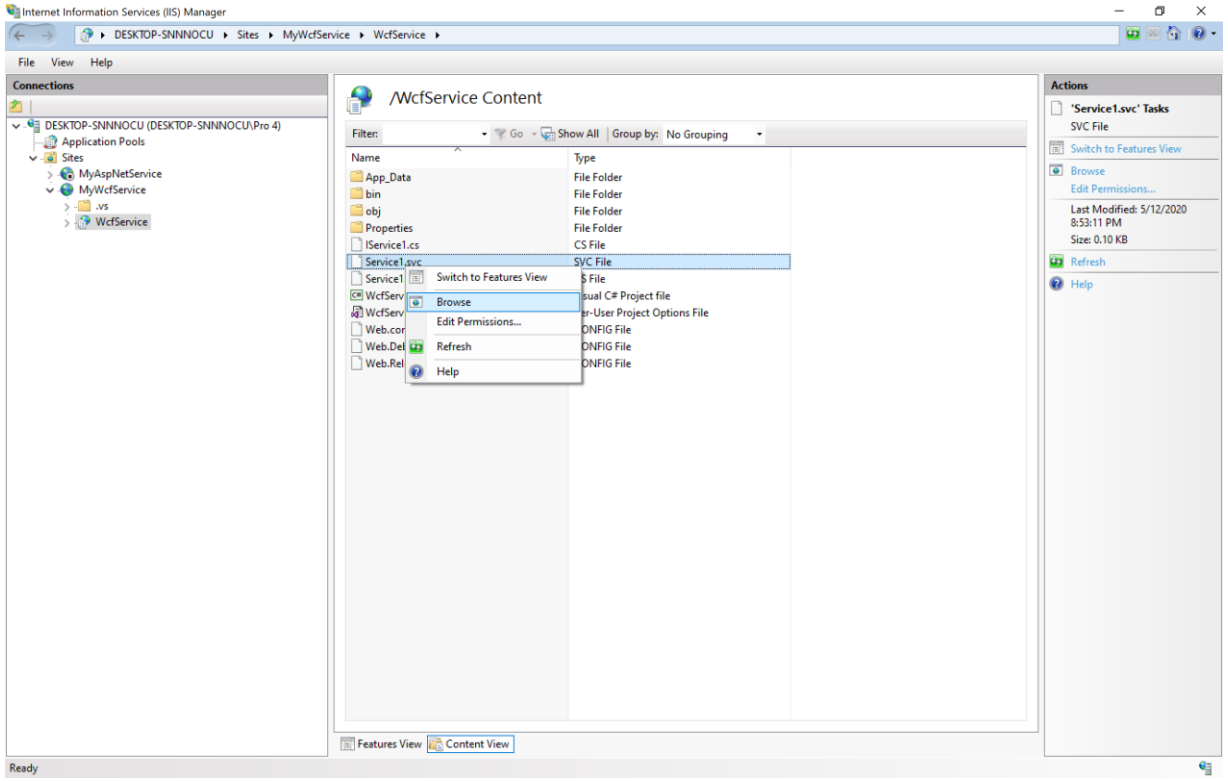
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

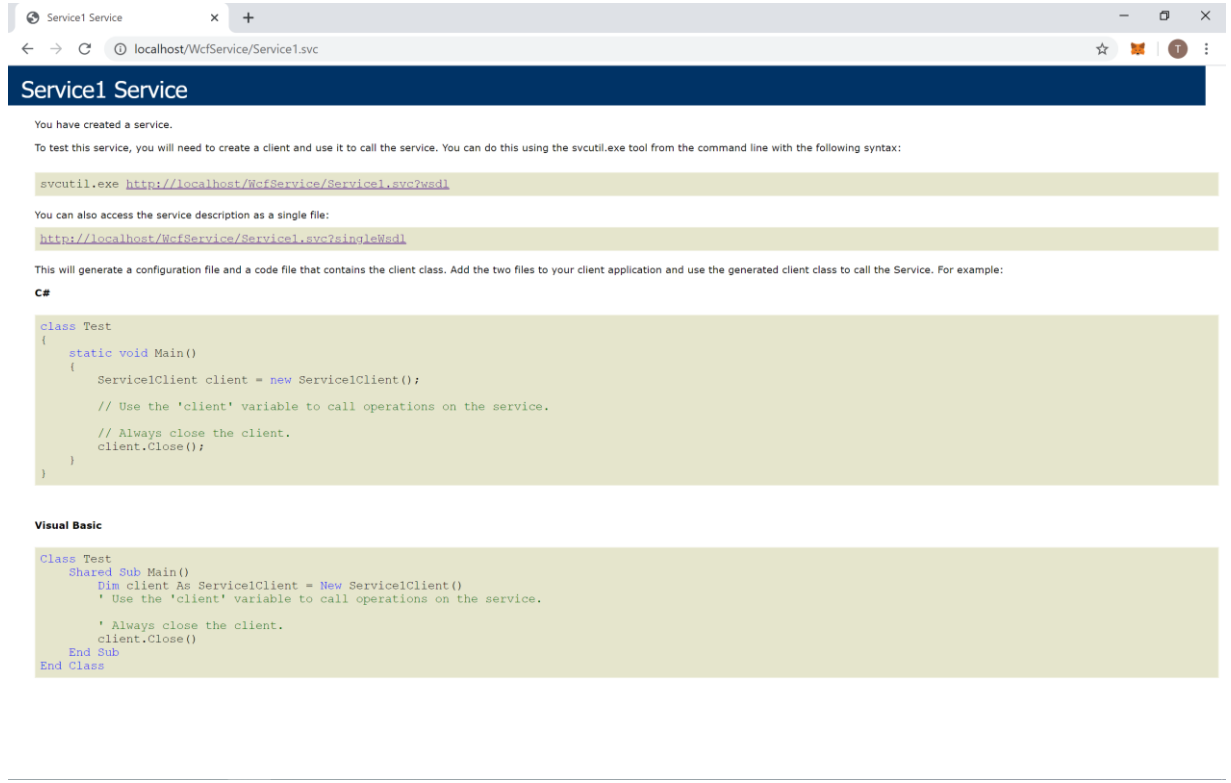
2. Deploy wcf-service application project in IIS



WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

3. Run wcf-service application project in IIS





WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

4. Create Android application (XML layout)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
        <TextView           android:id="@+id/txtResult"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:text="Hello World!"
                            app:layout_constraintBottom_toBottomOf="parent"
                            app:layout_constraintLeft_toLeftOf="parent"
                            app:layout_constraintRight_toRightOf="parent"
                            app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

4. Create Android application (MainActivity.java)

```
public class MainActivity extends Activity {
 TextView result;
 Handler handler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
   super.handleMessage(msg);
   result.setText((String) msg.obj);
  }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  result = (TextView) findViewById(R.id.txtResult);
  Thread slowJob = new Thread() {
   @Override
   public void run() {
    String resultValue = ""; HttpURLConnection con = null;
    try {
     String reqXML = "…";
     // Creating the HttpURLConnection object
     URL oURL = new URL("http://192.168.1.9/WcfService/Service1.svc");
     con = (HttpURLConnection) oURL.openConnection();
     con.setRequestProperty("Host", "localhost");
     con.setRequestProperty("Content-Type", "text/xml; charset=utf-8");
     con.setRequestProperty("Content-Length", String.valueOf(reqXML.length()));
     con.addRequestProperty("SOAPAction", "http://tempuri.org/IService1/GetData");
     con.setRequestMethod("POST"); con.setDoInput(true); con.setDoOutput(true);
     //Xml message
     OutputStreamWriter writer = new OutputStreamWriter(con.getOutputStream(), StandardCharsets.UTF_8);
     writer.write(reqXML);
     writer.flush();
     if(con.getResponseCode() == HttpURLConnection.HTTP_OK) {
      byte[] byteBuf = new byte[1024];
      InputStream resStream = con.getInputStream();
      int resLen = 0, len = resStream.read(byteBuf);
      while (len > -1) {
       resLen += len; resultValue += new String(byteBuf); len = resStream.read(byteBuf);
      }
     }
    }
    catch (Exception ex){ resultValue = "\nERROR: " + ex.getMessage(); }
    // send message to handler so it updates GUI
    Message msg = handler.obtainMessage(); msg.obj = (String) resultValue; handler.sendMessage(msg);
    if(con != null) con.disconnect();
   }
  };
  slowJob.start();
 }
}
```

note

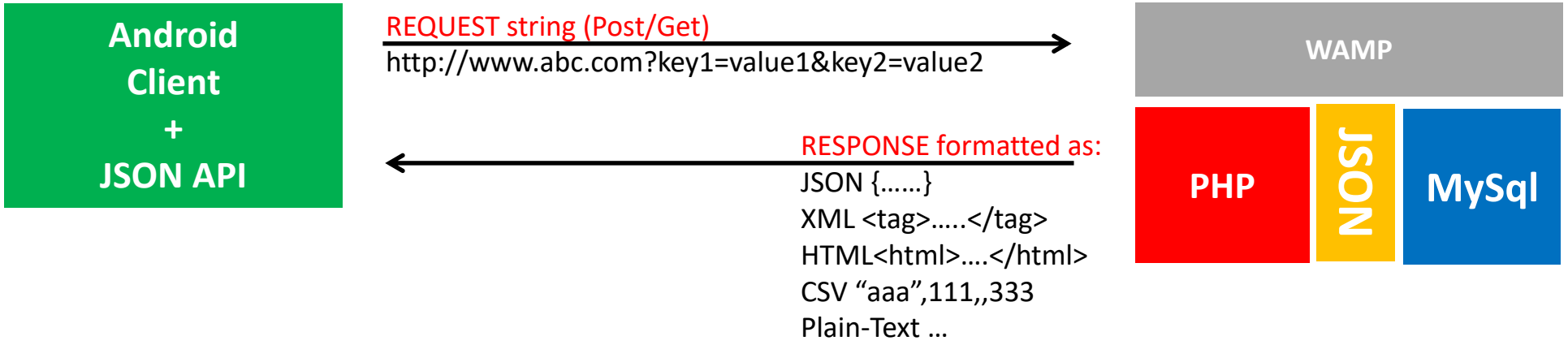
WINDOWS COMMUNICATION FOUNDATION (Example 1: Java HttpUrlConnection web-client)

4. Create Android application (MainActivity.java)

```
String reqXML = "<s:Envelope xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\">\n" +
                "  <s:Header/>\n" +
                "  <s:Body>\n" +
                "    <GetData xmlns=\"http://tempuri.org/\">\n" +
                "      <value>" + 2 + "</value>\n" +
                "    </GetData>\n" +
                "  </s:Body>\n" +
                "</s:Envelope>";
```

REPRESENTATIONAL STATE TRANSFER (REST)

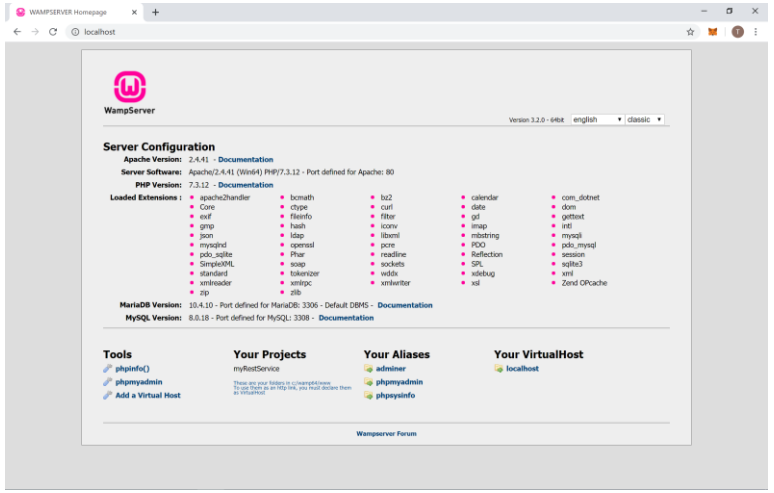
In this second example an Android client uses REST protocol to interact with a WAMP Server. WebServices are implemented as a set of PHP programs. We use JSON encoding for the client-server data exchange.



REPRESENTATIONAL STATE TRANSFER (Example 2: Java HttpUrlConnection web-client)

1. Create webservice application project
   - Install Wamp server
   - Start Webserver (Maybe there is some errors when starting PHP engines)

   - Open location: C:\wamp64\www and create a folder called myRestService (any names you want)
   - Go into that folder, and create 2 files: rest_services.php and restful_api.php
     - restful_api.php: code to handle REST request/response
     - rest_services.php: code of api you want to provide (ex: add2Integer function)



REPRESENTATIONAL STATE TRANSFER (Example 2: Java HttpUrlConnection web-client)

1. Create webservice application project (restful_api.php)

```
<?php
 class restful_api {
  protected $method = '';
  // Name of function called
  protected $endpoint = '';
  // parameters, ex: /<endpoint>/<param1>/<param2>
  protected $params = array();
  // file stored by PUT
  protected $file = null;
  public function __construct(){ $this->_input(); $this->_process_api(); }
  private function _input(){
     header("Access-Control-Allow-Orgin: *"); header("Access-Control-Allow-Methods: *");
     $this->params  = explode('/', trim($_SERVER['PATH_INFO'], '/'));
     $this->endpoint = array_shift($this->params);
     $method = $_SERVER['REQUEST_METHOD'];
     $allow_method  = array('GET', 'POST', 'PUT', 'DELETE');
     if (in_array($method, $allow_method))$this->method = $method;
     switch ($this->method) {
        case 'POST': $this->params = $_POST; break;
        case 'GET': // Không cần nhận, bởi params đã được lấy từ url
          break;
        case 'PUT': $this->file = file_get_contents("php://input"); break;
        case 'DELETE': // Không cần nhận, bởi params đã được lấy từ url
          break;
        default: $this->response(500, "Invalid Method"); break;
     }}
```

```
  // Thực hiện xử lý request
    private function _process_api(){
       if (method_exists($this, $this->endpoint)) $this->{$this->endpoint}();
       else $this->response(500, "Unknown endpoint");
    }
    protected function response($status_code, $data = NULL){
       header($this->_build_http_header_string($status_code));
       header("Content-Type: application/json");
       echo json_encode($data);
    }
    private function _build_http_header_string($status_code){
       $status = array(200 => 'OK', 404 => 'Not Found', 405 => 'Method Not Allowed', 500 => 'Internal Server Error');
       return "HTTP/1.1 " . $status_code . " " . $status[$status_code];
    }
  }
?>
```

REPRESENTATIONAL STATE TRANSFER (Example 2: Java HttpUrlConnection web-client)

1. Create webservice application project (rest_services.php)

```
<?php
 require 'restful_api.php';
 class rest_services extends restful_api {
  function __construct(){ parent::__construct();}
  function Add2Integer(){
   if ($this->method == 'GET'){
    $data = intval($this->params[0]) + intval($this->params[1]);
    $this->response(200, $data);
   }
   elseif ($this->method == 'POST'){ // body: a=1&b=2
    $data = intval($this->params['a']) + intval($this->params['b']);
    $this->response(200, $data);
   }
   elseif ($this->method == 'PUT'){ //body: a=1&b=2
    $p = explode("&", $this->file);
    $str = explode("=", $p[0])[1] . " + " . explode("=", $p[1])[1] . " = " . (intval(explode("=", $p[0])[1]) + intval(explode("=", $p[1])[1])); // 1 + 2 = 3
    $data = "successfully updated";
    file_put_contents("smile.txt", $str);
    $this->response(200, $data);
   }
   elseif ($this->method == 'DELETE'){ // code to delete data
    // return data by calling: $this->response(200, $data)
   }
  }
 }
 $user_rest_services = new rest_services(); ?>
```

REPRESENTATIONAL STATE TRANSFER (Example 2: Java HttpUrlConnection web-client)

2. Create Android application (MainActivity.java)

```
public class MainActivity extends Activity {
 TextView result;
 Handler handler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
   super.handleMessage(msg);
   result.setText((String) msg.obj);
  }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  result = (TextView) findViewById(R.id.txtResult);
  Thread slowJob = new Thread() {
   @Override
   public void run() {
    String resultValue = “”; HttpURLConnection con = null;
    try {
     Uri.Builder builder = new Uri.Builder().appendQueryParameter(“a”, “1”).appendQueryParameter(“b”, “2”);
     String query = builder.build().getEncodedQuery();
     URL oURL = new URL(“http://192.168.1.9/myRestService/rest_services.php/Add2Integer”);
     con = (HttpURLConnection) oURL.openConnection();
     con.setRequestMethod(“PUT”); //con.setRequestMethod(“POST”);
     con.setDoInput(true);
     con.setDoOutput(true);
     BufferedWriter writer = new BufferedWriter(
                                  new OutputStreamWriter(con.getOutputStream(), “UTF-8”));
     writer.write(query);
     writer.flush();
     if(con.getResponseCode() == HttpURLConnection.HTTP_OK) {
      byte[] byteBuf = new byte[1024];
      InputStream resStream = con.getInputStream();
      int resLen = 0, len = resStream.read(byteBuf);
      while (len > -1) {
       resLen += len; resultValue += new String(byteBuf); len = resStream.read(byteBuf);
      }
     }
    }
    catch (Exception ex){ resultValue = “\nERROR: ” + ex.getMessage(); }
    // send message to handler so it updates GUI
    Message msg = handler.obtainMessage(); msg.obj = (String) resultValue;
    handler.sendMessage(msg);
    if(con != null) con.disconnect();
   }
  };
  slowJob.start();
 }
}
```