

Axis2 ile Web Servis İstemci Örnekleri (Java, Delphi, C Sharp)

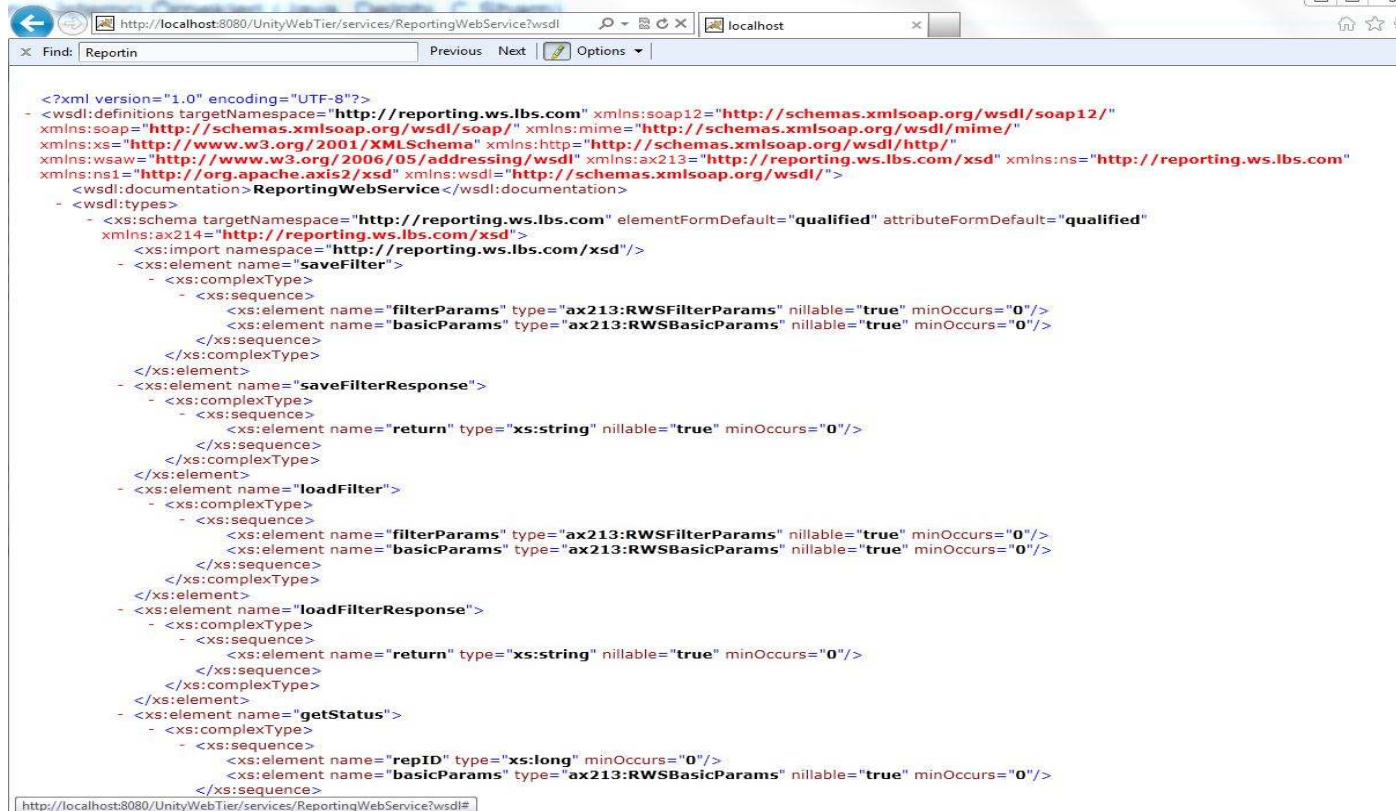
Bu dokümanda Unity On Demand (UOD) ürünündeki web servislerini kullanmak için farklı programlama dillerinde (Java, Delphi ve C#) istemci örneklerinin nasıl yazılacağı ve kullanılacağı anlatılacaktır.

UOD web servisleri ws-security (rampart) ile kullanıcı adı-şifre doğrulaması yaparak güvenli olarak kullanılabilir. Herhangi bir UOD uygulamasındaki web servislerini kullanabilmek için bu UOD'de tanımlı geçerli bir kullanıcıya ait kullanıcı adı ve şifre bilgilerine ihtiyaç vardır. Bu bilgiler olmadan UOD'deki web servislerini kullanmak mümkün değildir. Web servisi kullanan istemci kodlarından bu kullanıcı adı ve şifre bilgileri web servise yollanan SOAP mesajının başlık (header) bölümüne eklenmelidir. Bu işlemi yapabilmek için farklı programlama dillerinde farklı yöntemler bulunur; bu doküman Java, Delphi ve C# dilleri için bu yöntemleri anlatan örnekler üzerinden geçecektir.

Java, Delphi ve C# için örneklerde kullanılacak geliştirme ortamları sırayla Eclipse (3.6 ve üzeri bir sürüm), Delphi 2010 RAD Studio ve Visual Studio 2005'tir. Bu geliştirme ortamlarının tümünde web servise ait istemci kodlarını web servise ait WSDL'dan üreten birer mekanizma bulunur. Bu kod üretimini yapabilmek için web servise ait WSDL erişilebilir durumda olmalıdır. Bunun için aşağıdaki örnekleri denemeye başlamadan önce web servise ait WSDL dosyasının erişilebilir durumda olduğundan emin olunmalıdır. Örneklerde kullanılacak web servise ait WSDL adresi

http://<UOD_ip>:<UOD_port>/UnityWebTier/services/ReportingWebService?wsdl şeklindedir.

Örnek bir adres şöyle olur: <http://172.16.12.239:8080/UnityWebTier/services/ReportingWebService?wsdl> Aşağıdaki örnekleri denemeden önce bir internet tarayıcısında adres çubuğuna bağlanılacak UOD'ye ait bu wsdl adresi yazılıp web servisin erişilebilir durumda olduğu kontrol edilmelidir. Eğer web servis erişilebilir durumda ise aşağıdaki örnek gibi bir sayfa açılmalıdır:

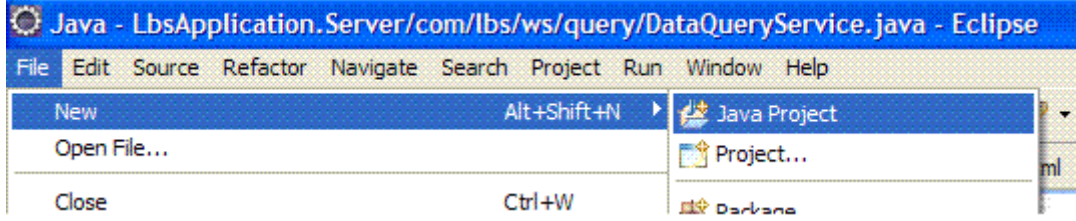


```
<?xml version="1.0" encoding="UTF-8"?>
- <wsdl:definitions targetNamespace="http://reporting.ws.lbs.com" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:ax213="http://reporting.ws.lbs.com/xsd" xmlns:ns="http://reporting.ws.lbs.com"
xmlns:ns1="http://org.apache.axis2/xsd" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:documentation>ReportingWebService</wsdl:documentation>
  - <wsdl:types>
    - <xs:schema targetNamespace="http://reporting.ws.lbs.com" elementFormDefault="qualified" attributeFormDefault="qualified"
xmlns:ax214="http://reporting.ws.lbs.com/xsd">
      <xs:import namespace="http://reporting.ws.lbs.com/xsd"/>
      - <xs:element name="saveFilter">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="filterParams" type="ax213:RWSFilterParams" nillable="true" minOccurs="0"/>
            - <xs:element name="basicParams" type="ax213:RWSBasicParams" nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      - <xs:element name="saveFilterResponse">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="return" type="xs:string" nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      - <xs:element name="loadFilter">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="filterParams" type="ax213:RWSFilterParams" nillable="true" minOccurs="0"/>
            - <xs:element name="basicParams" type="ax213:RWSBasicParams" nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      - <xs:element name="loadFilterResponse">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="return" type="xs:string" nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      - <xs:element name="getStatus">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="repID" type="xs:long" minOccurs="0"/>
            - <xs:element name="basicParams" type="ax213:RWSBasicParams" nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>

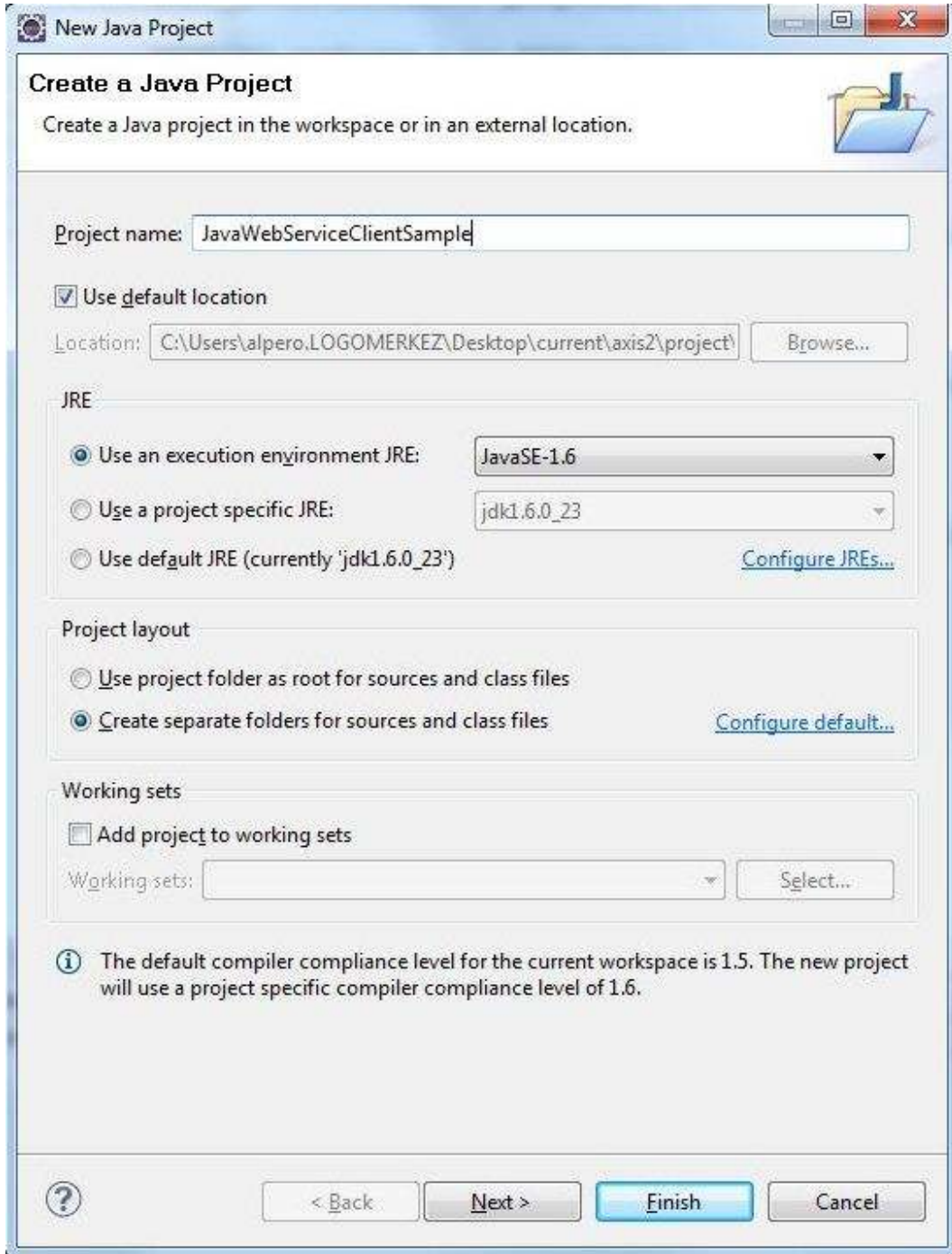
```

Java İle İstemci Yazılması

Java ile web servis istemcisi yazabilmek için Eclipse geliştirme ortamı kullanılır. Eclipse içinden öncelikle bir Java projesi yaratılır. (Eğer web servis istemcisi varolan bir java projesinin içine yaratılacaksa bu adım atlanabilir)



Açılan sihirbazda projeye bir isim verilir ve varsayılan ayarlarla java projesi yaratılır.



Axis2 ile web service client oluşturmak için axis2-eclipse-codegen-plugin-1.5.4 plugin inin kurulması gerekiyor. Bu plugin:

<http://www.apache.org/dyn/mirrors/mirrors.cgi/axis/axis2/java/core/1.5.4/axis2-eclipse-codegen-plugin-1.5.4.zip>

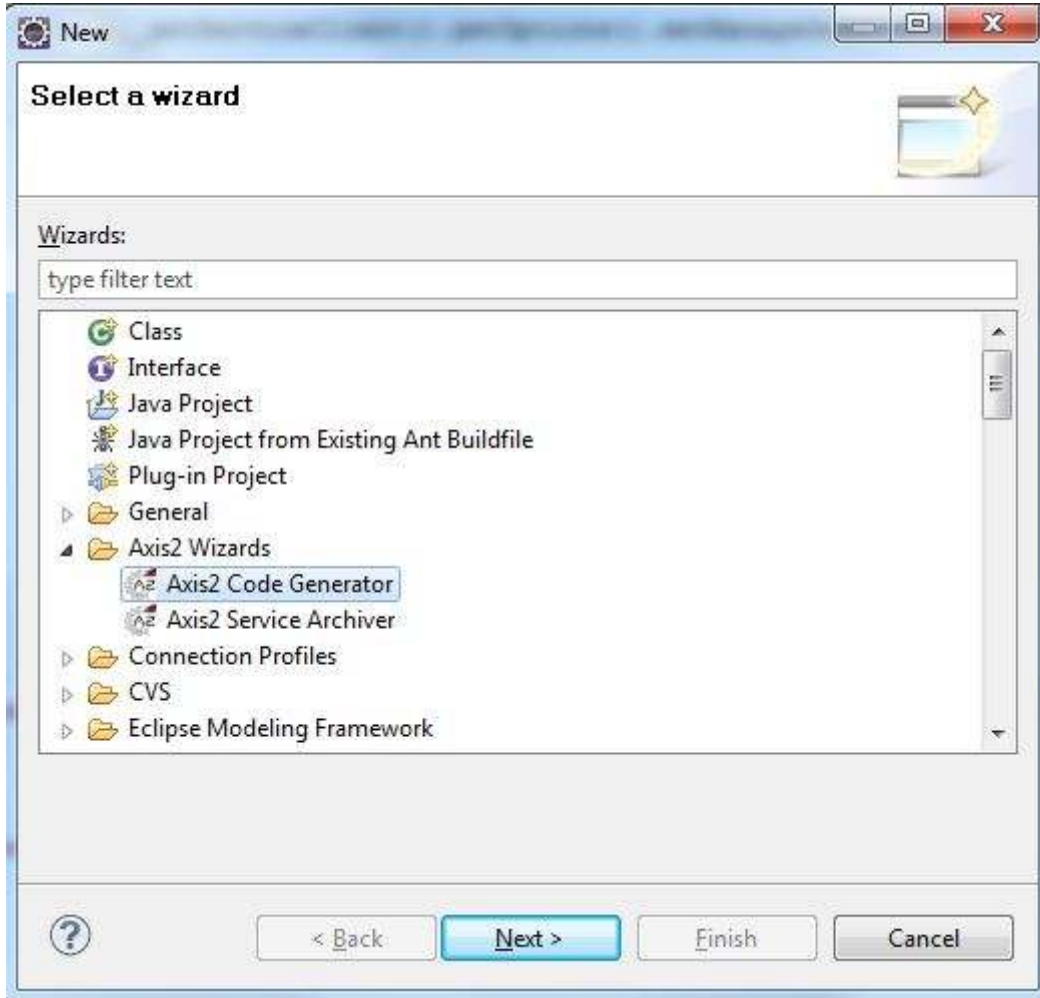
adresinden indirilebilir.

Not: 2.10.0.0 altyapı setiyle birlikte Axis1.6 ya geçilmiştir. Bu set ve sonrası için "axis2-eclipse-codegen-plugin-1.6.1.zip":

<http://www.apache.org/dyn/mirrors/mirrors.cgi/axis/axis2/java/core/1.6.1/axis2-eclipse-codegen-plugin-1.6.1.zip>

adresinden indirilmelidir. Plugin kullanımında bir farklılık yoktur.

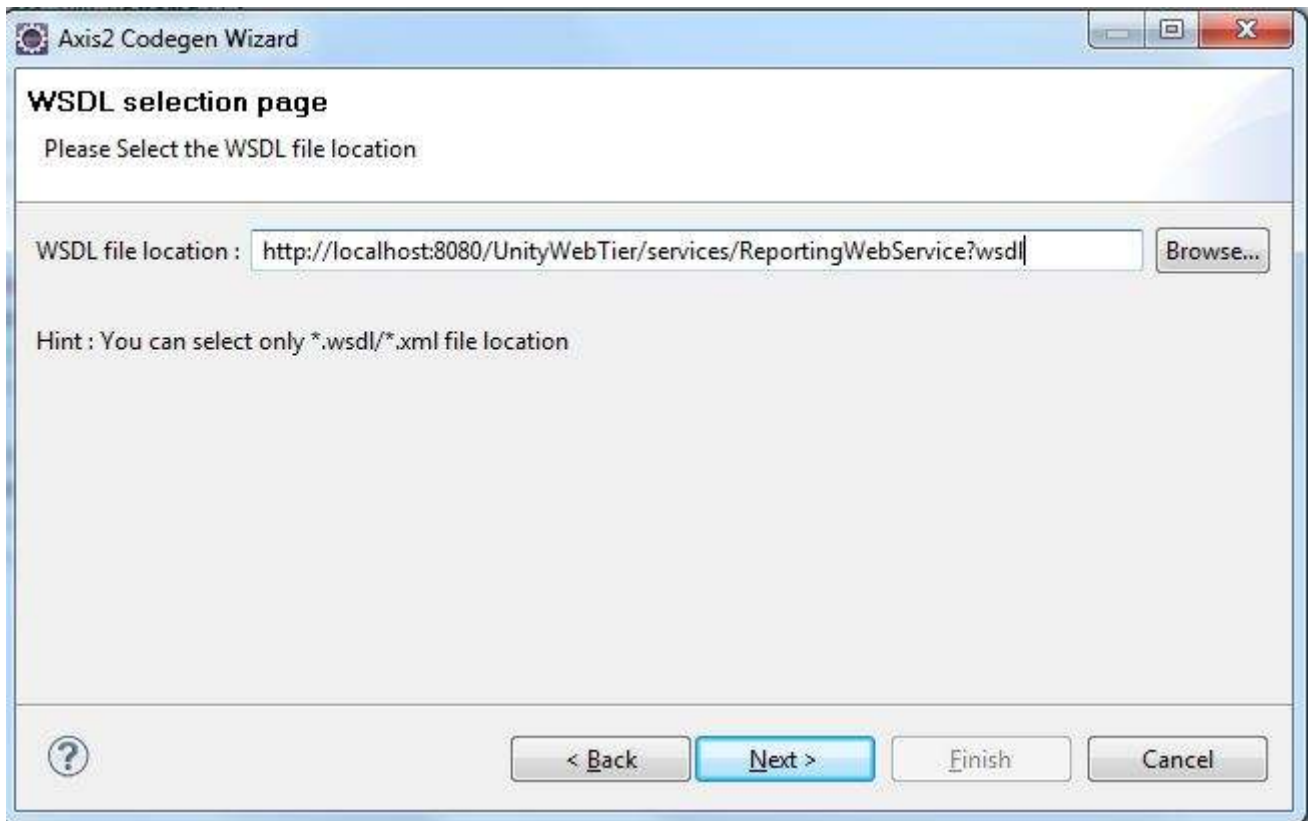
Plugin kurulduktan sonra, oluşan projede (ya da istenen java projesinde) New -> Other seçilir ve açılan pencereden "Axis2 Code Generator" seçeneği seçilir.



Generate Java source code from WSDL file seçilir ve "Next" tıklanır.



Burada client ını yazmak istediğimiz servisin wsdl url i yazılır ve "Next" tıklanır.



Bu plugin vasıtasıyla 4 farklı şekilde web service client oluşturulabilir. Biz burada varsayılan databinding olan ADB yi seçeceğiz. Dolayısıyla bu ekranda herhangi değişiklik yapmadan "Next" tıklarız.

Options

Set the options for the code generator. If you wish to edit the codegen options, Select custom option from Codegen Option drop down list.

Codegen option: default

Output language: java

Service Name: ReportingWebService

Port Name: ReportingWebServiceHttpSoap12Endpoint

Databinding Name: adb

Custom package name: com.lbs.ws.reporting

Generate test case

Generate client side code

Generate both sync and async Generate sync style only Generate async style only

Generate server side code

Generate a default services.xml Generate an Interface for Skeleton

Generate Both with all classes for every elements on Schemas

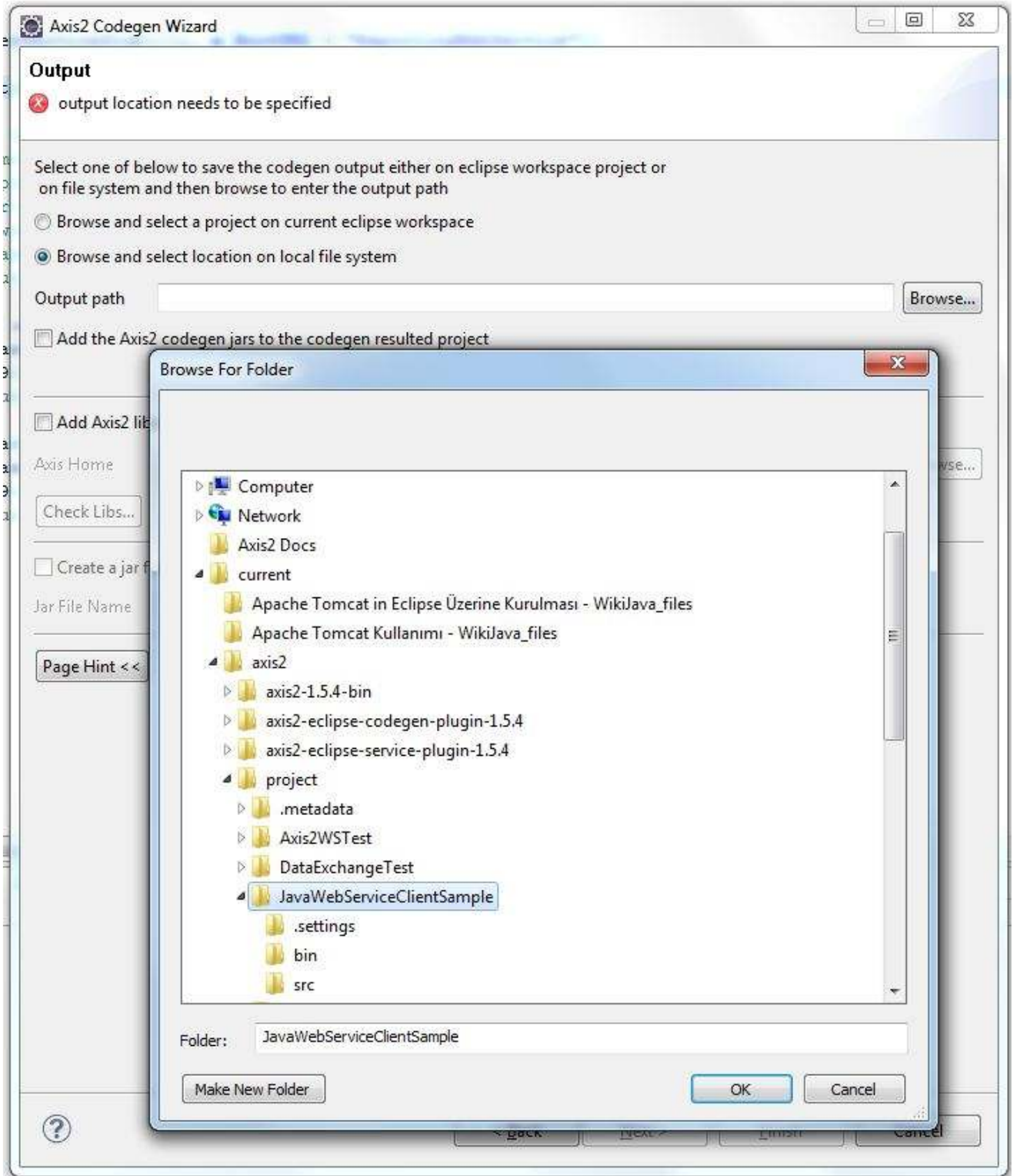
Namespace to package mappings

| Namespace | Custom package name |
|---|----------------------------------|
| http://org.apache.axis2/xsd | axis2.apache.org.xsd |
| http://reporting.ws.lbs.com | com.lbs.ws.reporting |
| http://reporting.ws.lbs.com/xsd | com.lbs.ws.reporting.xsd |
| http://www.w3.org/2006/05/addressing/wsdl | org.w3.www._2006_05.addressin... |
| http://schemas.xmlsoap.org/wsdl/ | org.xmlsoap.schemas.wsdl |
| http://schemas.xmlsoap.org/wsdl/http/ | org.xmlsoap.schemas.wsdl.http |
| http://www.w3.org/2001/XMLSchema | org.w3.www._2001.xmlschema |
| http://schemas.xmlsoap.org/wsdl/soap/ | org.xmlsoap.schemas.wsdl.soap |
| http://schemas.xmlsoap.org/wsdl/mime/ | org.xmlsoap.schemas.wsdl.mime |
| http://schemas.xmlsoap.org/wsdl/soap12/ | org.xmlsoap.schemas.wsdl.soap12 |

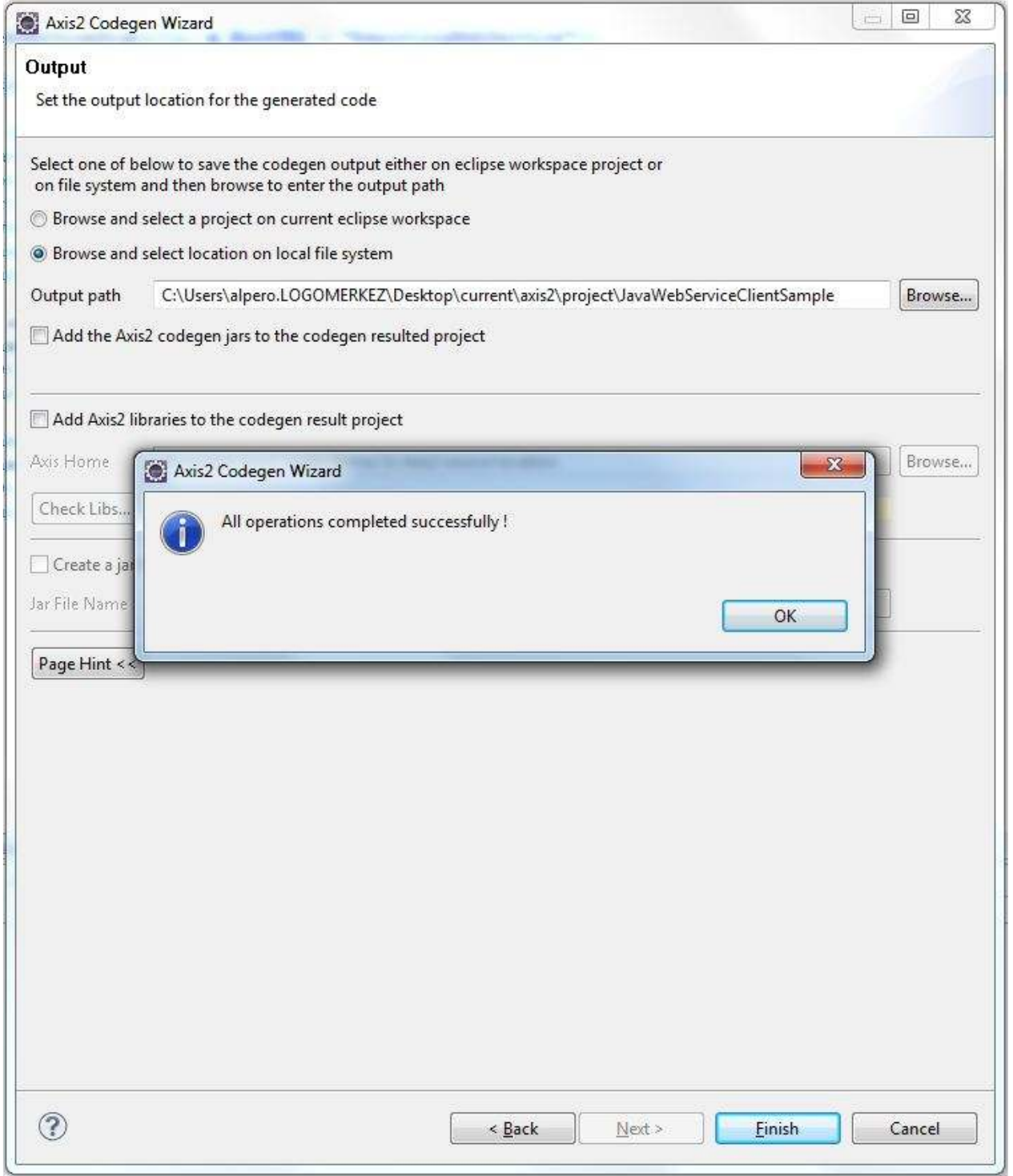
Advance Options

? < Back Next > Finish Cancel

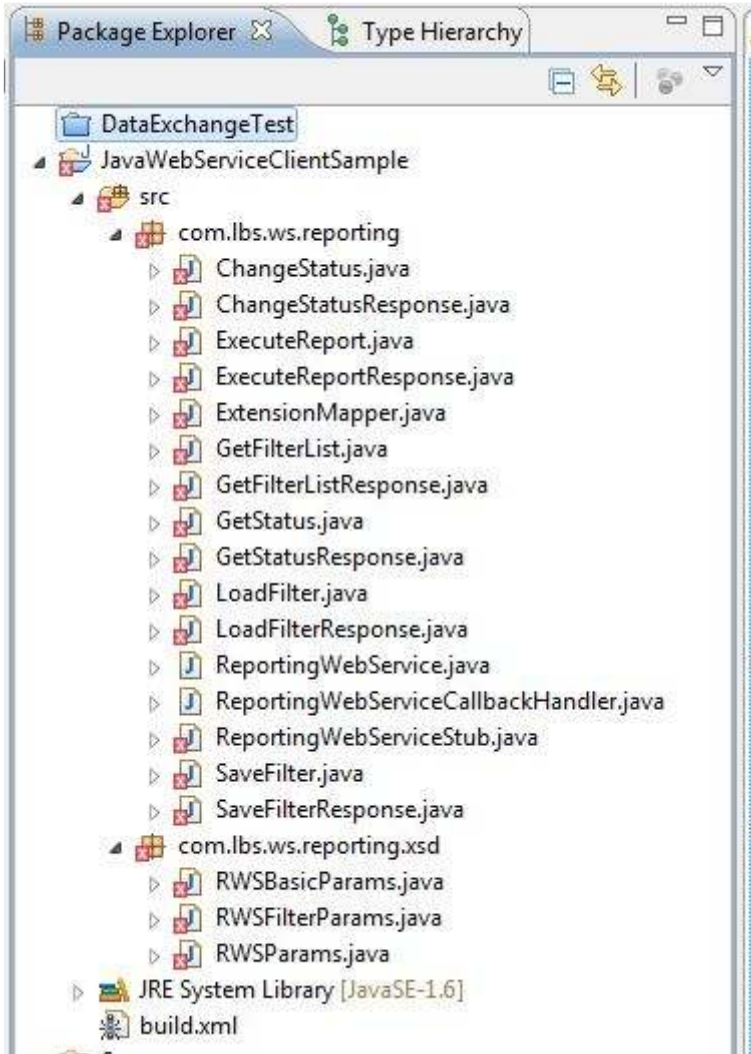
Gelen ekranda "Output path" olarak yukarıda yaratılan proje seçilir ve "Finish" tıklanır.



Aşağıdaki ekran görüldüğünde "Ok" tıklanır ve oluşturulan projede sağ tıklanarak "Refresh" uygulanır.



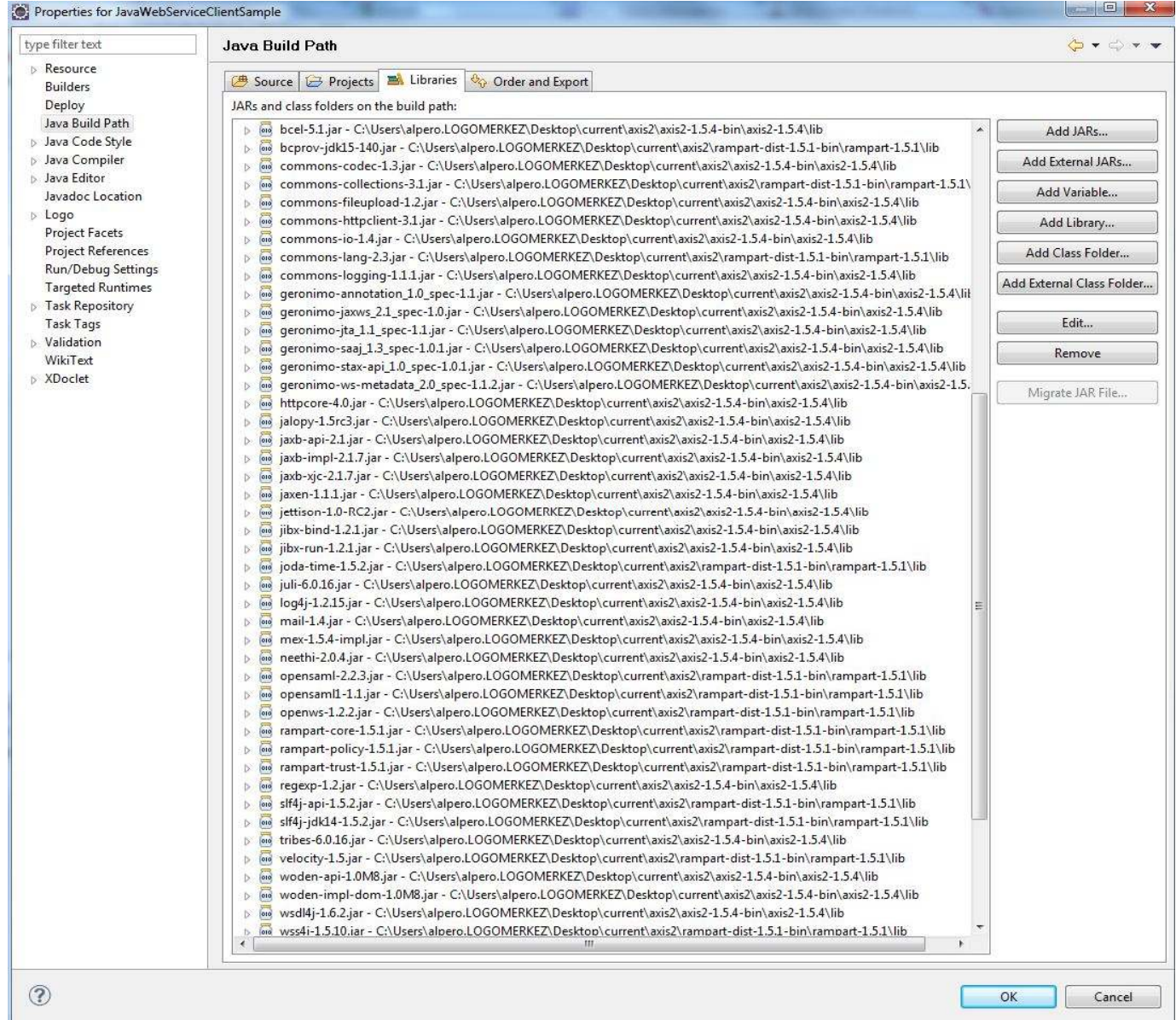
Aşağıdaki yapı görülür.



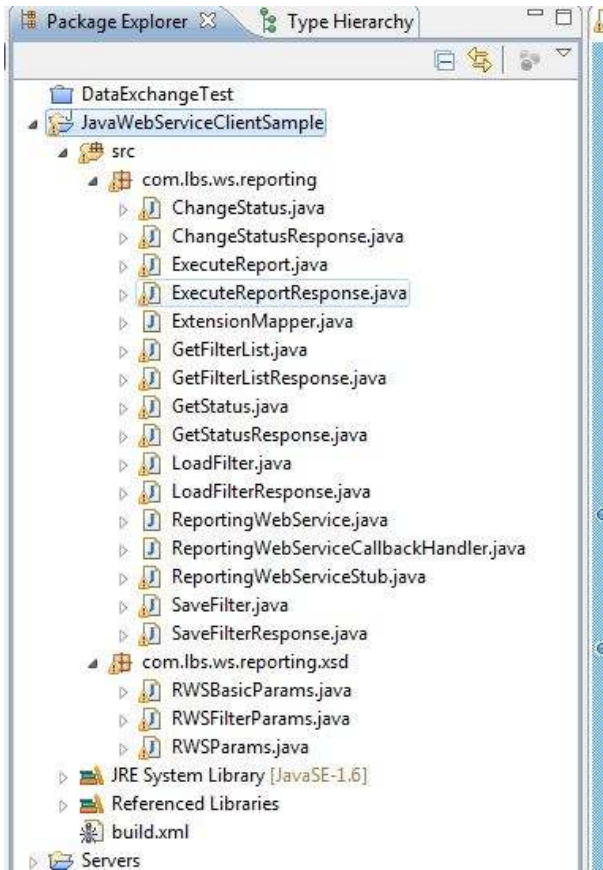
Daha sonra bu projenin classpath ine Axis2 ve Rampart kütüphaneleri eklenir. Axis2 kütüphanelerine <http://axis.apache.org/axis2/java/core/download.cgi> adresinden, Rampart kütüphanelerine <http://axis.apache.org/axis2/java/rampart/download.html> adresinden ulaşılabilir. Burada anlatılan örnek için Axis2 1.6.0 ve Rampart 1.6.0 sürümleri kullanılmıştır. Buradan indirilen zip dosyaları herhangi bir yere açılır ve lib dizinleri altında yer alan jarlar projenin classpath ine eklenir.

Not 2: Axis2 için Binary Distribution ve Rampart için de Standart Binary Distribution linklerinden ilgili versiyonlar indirilebilir.

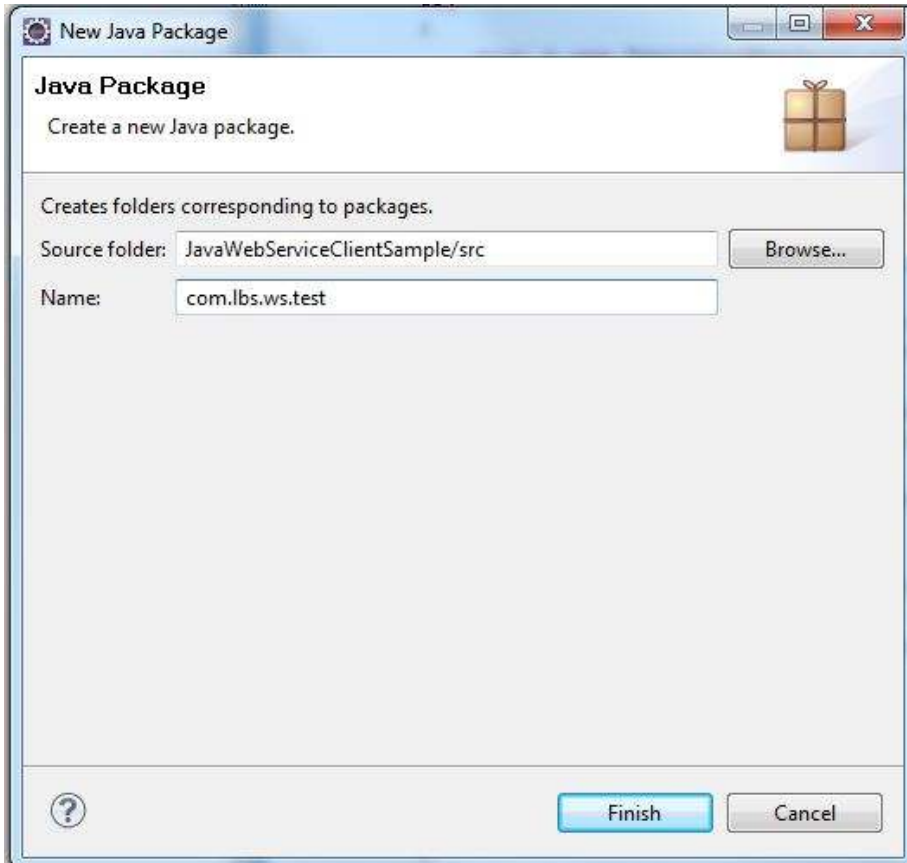
Not 3: Axis kütüphanelerine axis2-1.6.0-bin.zip\axis2-1.6.0\lib altından ve Rampart kütüphanelerine ise rampart-dist-1.6.0-bin.zip\rampart-1.6.0\lib altından ulaşılabilir.



Daha sonra projedeki hataların düzeldiği görülür.



Yukarıda oluşturulan projeye com.lbs.ws.test adında bir paket eklenir.



Bu pakete "testService" adında bir class oluşturulur.

New Java Class

Java Class

⚠ Type name is discouraged. By convention, Java type names usually start with an uppercase letter

Source folder:

Package:

Enclosing type:

Name:

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)
 Constructors from superclass
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

```

public class testService
{
    private static String m_RootURL = "http://localhost:8080/UnityWebTier/services/";

    public static void main(java.lang.String args[])
    {
        try
        {
            ConfigurationContext cfg =
            ConfigurationContextFactory.createConfigurationContextFromFileSystem("D:\\Adressing", null);
            ReportingWebServiceStub stub = new ReportingWebServiceStub(cfg,
            m_RootURL + "ReportingWebService");

            ServiceClient sc = stub._getServiceClient();
            sc.engageModule("addressing");
            sc.engageModule("rampart");

            OutflowConfiguration outflowConfig = new OutflowConfiguration();
            outflowConfig.setActionItems("UsernameToken");
            outflowConfig.setUser("admin");

            outflowConfig.setPasswordCallbackClass("com.lbs.ws.test.SampleCallBackHandler");
            sc.getOptions().setProperty(WSSHandlerConstants.OUTFLOW_SECURITY,
            outflowConfig.getProperty());

            RWSBasicParams basicParams = new RWSBasicParams();
            basicParams.setFirm(1);
            basicParams.setPeriod(1);
            basicParams.setLanguage("TRTR");

            RWSParams rwsParams = new RWSParams();
            rwsParams.setReportName("MMRPItemsList");
            rwsParams.setOutputType(2);

            ExecuteReport exeReport = new ExecuteReport();
            exeReport.setParams(rwsParams);
            exeReport.setBasicParams(basicParams);

            System.out.println(stub.executeReport(exeReport).get_return());
        }
        catch (RemoteException e)
        {
            e.printStackTrace();
        }
    }
}

```

Yukarıdaki methodda yapılan işlemler:

```

ConfigurationContext cfg =
ConfigurationContextFactory.createConfigurationContextFromFileSystem("D:\\Adressing", null);

```

Burada ConfigurationContext oluşturuluyor. Burada verilen ilk parametre MODULES dizindir. Yapısı aşağıda anlatılacaktır. İkinci parametre ise "axis2.xml" dosyasının gösteren parametredir (dosyanın ismi önemli değildir). null verildiğinde Axis2 kütüphanelerinde yer alan varsayılan "axis2.xml" kullanılır.

```

ServiceClient sc = stub._getServiceClient();
sc.engageModule("addressing");
sc.engageModule("rampart");

OutflowConfiguration outflowConfig = new OutflowConfiguration();
outflowConfig.setActionItems("UsernameToken");
outflowConfig.setUser("admin");
outflowConfig.setPasswordCallbackClass("com.lbs.ws.test.SampleCallBackHandler");
sc.getOptions().setProperty(WSSHandlerConstants.OUTFLOW_SECURITY,outflowConfig.getProperty());

```

Burada Web Service Security (rampart) kullanacağımızı ve password bilgisinin SampleCallBackHandler clasından alınacağını tanımladık. Kullanıcı ismi olarak "admin" tanımladık.

SampleCallBackHandler içeriği aşağıdaki şekildedir:

```
package com.lbs.ws.test;
import java.io.IOException;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import javax.security.auth.callback.Callback;
import org.apache.ws.security.WSPasswordCallback;

public class SampleCallBackHandler implements CallbackHandler {
    public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException
    {
        for (int i = 0; i < callbacks.length; i++) {
            WSPasswordCallback pwcb = (WSPasswordCallback) callbacks[i];
            String id = pwcb.getIdentifier();
            if ("admin".equals(id)) {
                pwcb.setPassword("1");
            }
        }
    }
}
```

Burada "admin" kullanıcısı için şifreyi "1" olarak belirledik. Dolayısıyla bağlanacağımız Unity On Demand deki admin kullanıcısının şifresinin "1" olması gerekmektedir.

Adressing klasörünün içeriği şu şekildedir:

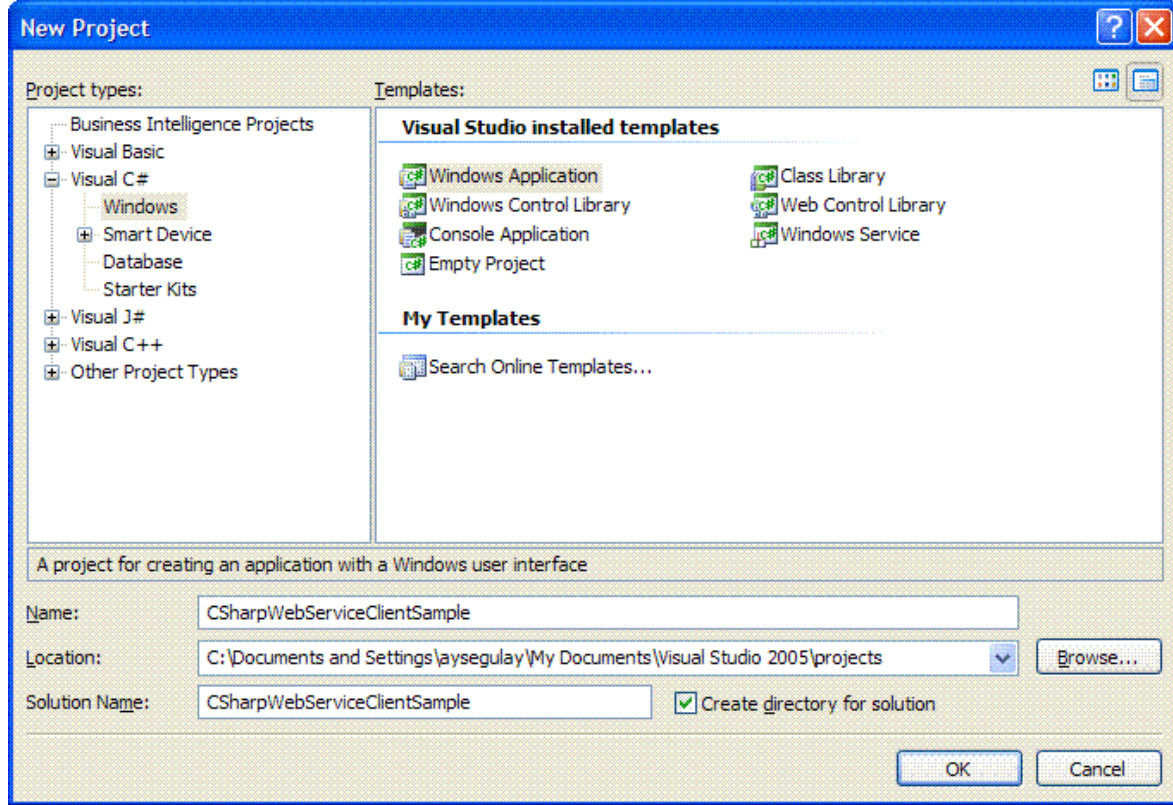
```
--D:\Adressing
--MODULES
    addressing-1.6.0.mar (Axis2 kütüphanesi ile gelir ve repository/modules dizini altında yer alır)
    rampart-1.6.0.mar (Rampart kütüphanesi ile gelir ve modules dizini altında yer alır)
```

Örnek projenin kodlarına **JavaWebServiceClientSample.rar** içerisinden ulaşılabilir.

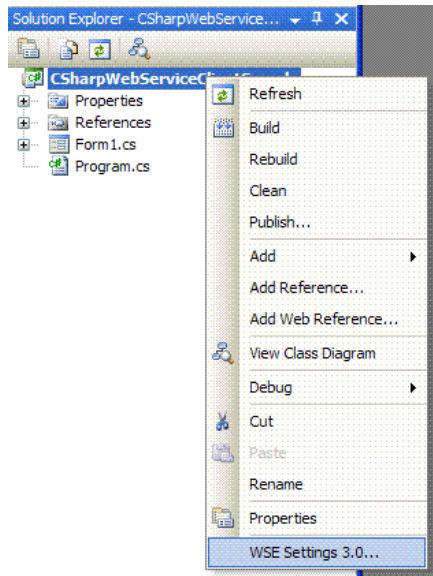
C Sharp ile istemci yazılması

C# ile web servis istemci örneği için geliştirme ortamı olarak Visual Studio 2005 kullanılacaktır. Web servis güvenlik kısımları için kullanılacak WSE 3.0 ile VS 2008 entegrasyonu desteklenmediği için örnek VS 2005 üzerinden anlatılacaktır. VS 2005'le devam etmeden önce bilgisayara WSE 3.0 kurulması gerekir. Windows update sitesinden Web Service Enhancements (WSE) 3.0 indirilir ve makineye kurulur. Bu kurulum sırasında mutlaka Visual Studio 2005 ile geliştirme yapma kutucuğu seçilip kurulur. Yoksa VS 2005 içinde WSE 3.0 ile ilgili kısımlar çalışmaz.

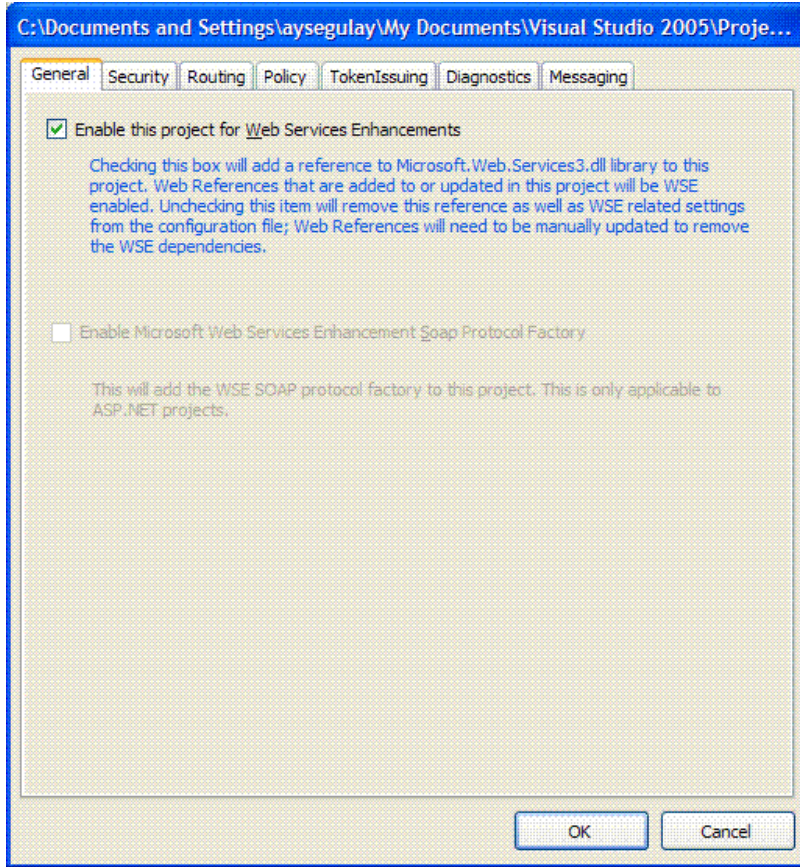
WSE 3.0 kurulduktan sonra VS 2005 açılır ve yeni bir C# - Windows Application projesi yaratılır. (File -> New -> Project)



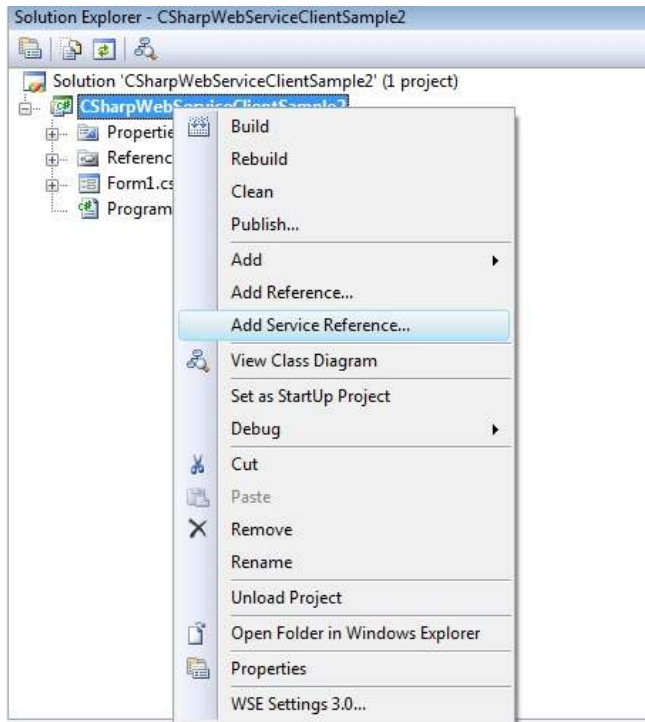
Bu projede sağ tuş menüsünden "WSE Settings 3.0" seçeneği seçilir.



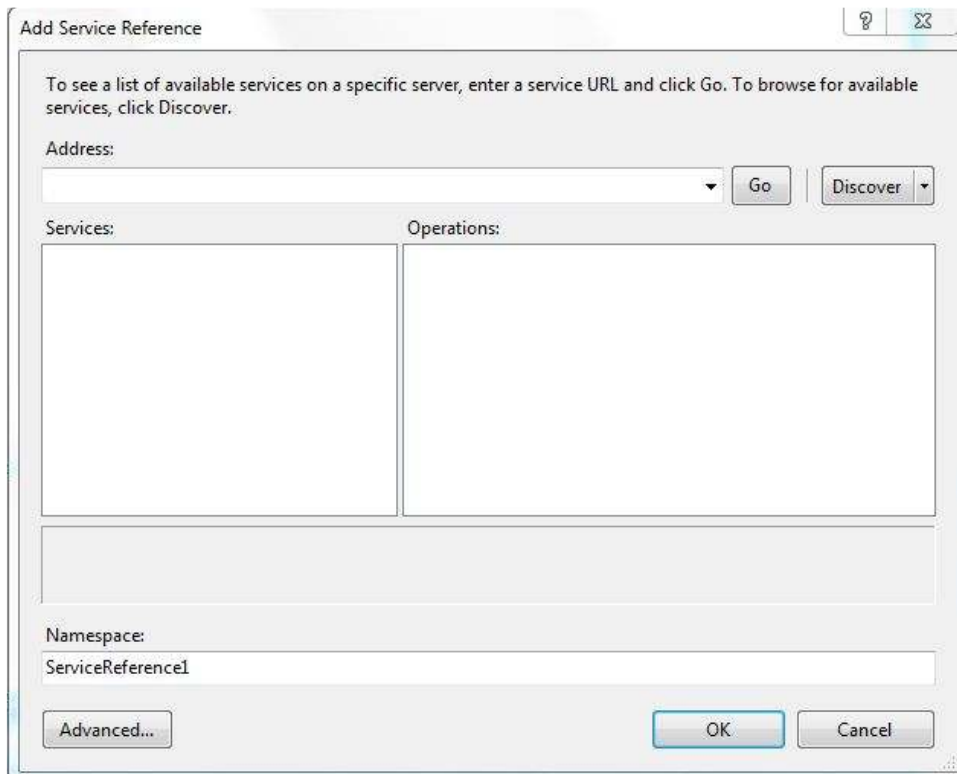
Açılan pencereden "Enable this project for Web Services Enhancements" kutucuğu seçilir ve OK ile kapatılır.



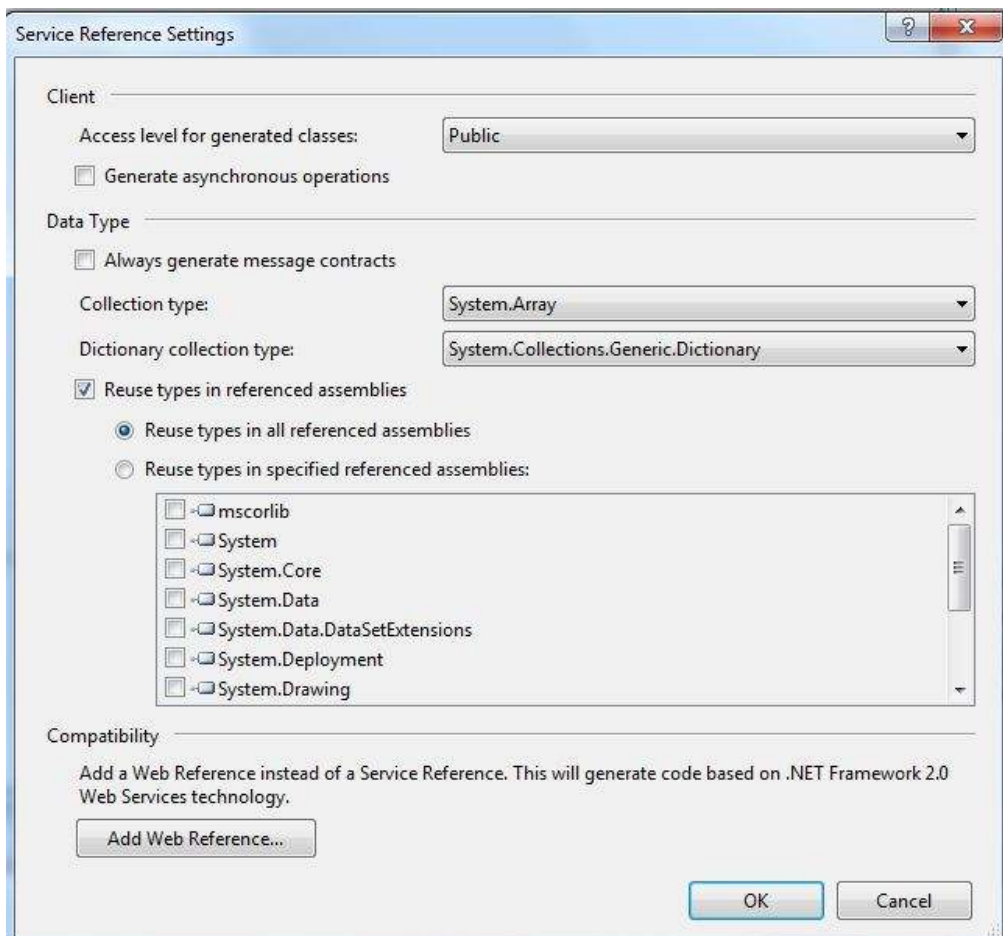
Yine sağ tuş menüsünden "Add Service Reference" seçeneği seçilir.



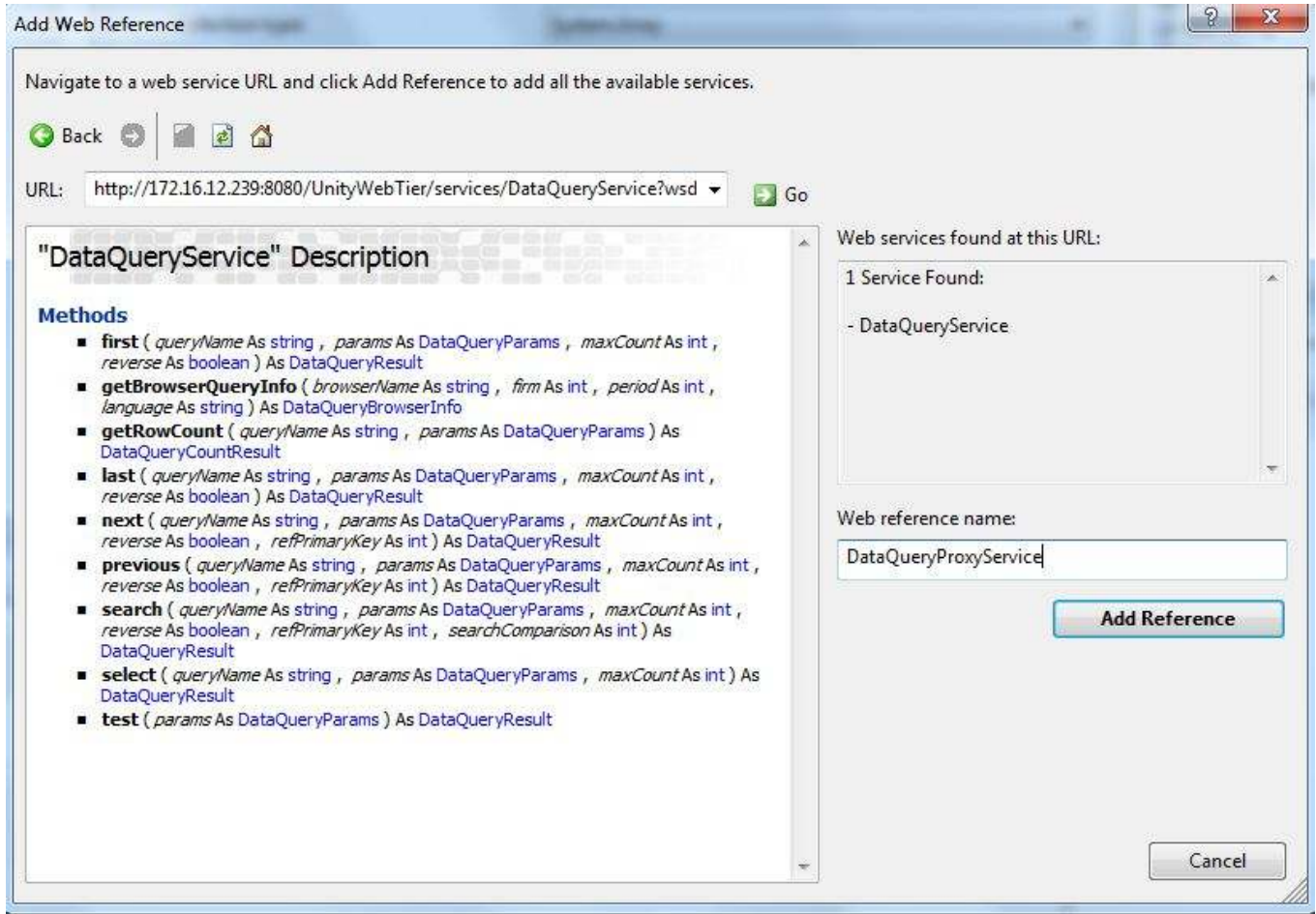
"Advanced" butonuna tıklanır.



"Add Web Reference" butonuna tıklanır.



Açılan pencerede "URL" kısmına wsdl adresi yazılır ve Go tuşuna basılır. Go tuşuna basılınca servisle ilgili bilgiler soldaki bölüme dolar. Bundan sonra "Web Reference Name" özelliğine servisle ilgili anlamlı bir isim (örnekte DataQueryProxyService) verilip "Add Reference" tuşuna basılır.



Bu adımla istemci için kodlar üretilir.

Bu adımdan sonra **CSharp_WebServis.zip** kataloğu indirilir ve bir yere açılır. Bu katalogta bulunan **TimestampRemoverOutputFilter.cs** ve **TimestampRemoverAssertion.cs** dosyaları (Proje üzerinde sağ tuşla Add -> Existing Item seçeneği ile) projeye eklenir. Eklenen bu dosyalardaki **namespace** özellikleri projenin adına göre güncellenir. Form1.cs formuna gidilir ve test için bu forma bir tuş eklenir. Bu tuşun click metoduna web servis istemcisi test kodları yazılır. Örnek click metodu şöyle olur:

```
private void button1_Click(object sender, EventArgs e)
{
    DataQueryServiceProxy.DataQueryServiceServiceWse svc
= new DataQueryServiceProxy.DataQueryServiceServiceWse();

    svc.Url = "http://172.16.12.239:8080/UnityWebTier/services/DataQueryService?wsdl";

    Microsoft.Web.Services3.Security.Tokens.UsernameToken token
= new Microsoft.Web.Services3.Security.Tokens.UsernameToken("admin", "logo", Microsoft.Web.Services3.Security.Tokens.PasswordOption.SendPlainText);

    svc.SetClientCredential(token);

    Policy myPolicy = new Policy();

    myPolicy.Assertions.Add(new UsernameOverTransportAssertion());

    myPolicy.Assertions.Add(new TimestampRemoverAssertion());
}
```

```

svc.SetPolicy(myPolicy);

DataQueryServiceProxy.DataQueryParams prm = new DataQueryServiceProxy.DataQueryParams();
DataQueryServiceProxy.DataQueryResult res = svc.test(prm);

if (res != null)
{
    MessageBox.Show("Done for the test method");
}
else
    MessageBox.Show("Testmethod returned null, check ResponseNamespace properties of generated clientproxy!");

prm.firm = 1;
prm.period = 1;

res = svc.first("MMQOItemBrowser",prm, 10, false);

if (res != null)
    MessageBox.Show("Done for first method");
else
    MessageBox.Show("First method returned null, check ResponseNamespaceproperties of generated client proxy!");
}

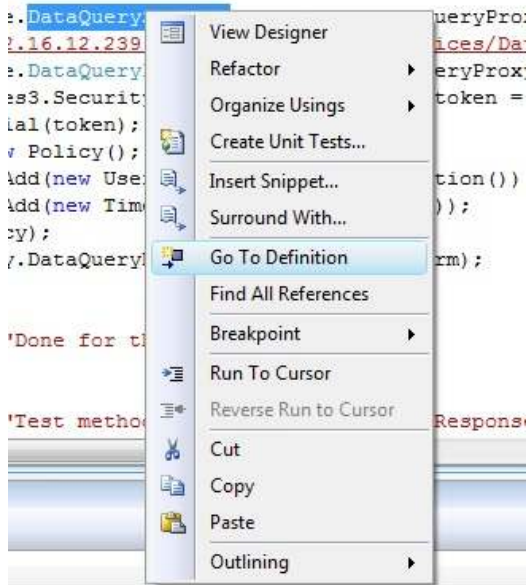
```

Bu kodların düzgün çalışabilmesi için yapılması gereken birkaç değişiklik vardır.

Aşağıda gösterildiği gibi "DataQueryServiceServiceWse" tanımına gidilir ve aşağıdaki değişiklikler yapılır.

using Microsoft.Web; satırı eklenir.

System.Web.Services.Protocols.SoapHttpClientProtocol yerine Microsoft.Web.Services3.WebServiceClientProtocol yazılır.



```
CSharpWebServiceClientSample2.DataQueryProxyService.DataQueryService selectOperationCompleted
#pragma warning disable 1591

namespace CSharpWebServiceClientSample2.DataQueryProxyService {
    using System.Diagnostics;
    using System.Web.Services;
    using System.ComponentModel;
    using System.Web.Services.Protocols;
    using System;
    using System.Xml.Serialization;

    /// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.Web.Services", "2.0.50727.4927")]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Web.Services.WebServiceBindingAttribute(Name="DataQueryServiceSoap11Binding", Namespace="http://query.ws.lbs.com")]
    public partial class DataQueryService : System.Web.Services.Protocols.SoapHttpClientProtocol {

        private System.Threading.SendOrPostCallback selectOperationCompleted;

        private System.Threading.SendOrPostCallback testOperationCompleted;

        private System.Threading.SendOrPostCallback firstOperationCompleted;

        private System.Threading.SendOrPostCallback getBrowserQueryInfoOperationCompleted;

        private System.Threading.SendOrPostCallback previousOperationCompleted;

        private System.Threading.SendOrPostCallback getRowCountOperationCompleted;

        private System.Threading.SendOrPostCallback lastOperationCompleted;

        private System.Threading.SendOrPostCallback searchOperationCompleted;

        private System.Threading.SendOrPostCallback nextOperationCompleted;
    }
}
```

```
CSharpWebServiceClientSample2.DataQueryProxyService.DataQueryService selectOperationCompleted
#pragma warning disable 1591

namespace CSharpWebServiceClientSample2.DataQueryProxyService {
    using System.Diagnostics;
    using System.Web.Services;
    using System.ComponentModel;
    using System.Web.Services.Protocols;
    using System;
    using System.Xml.Serialization;
    using Microsoft.Web;

    /// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.Web.Services", "2.0.50727.4927")]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Web.Services.WebServiceBindingAttribute(Name="DataQueryServiceSoap11Binding", Namespace="http://query.ws.lbs.com")]
    public partial class DataQueryService : Microsoft.Web.Services3.WebServicesClientProtocol {

        private System.Threading.SendOrPostCallback selectOperationCompleted;

        private System.Threading.SendOrPostCallback testOperationCompleted;

        private System.Threading.SendOrPostCallback firstOperationCompleted;

        private System.Threading.SendOrPostCallback getBrowserQueryInfoOperationCompleted;

        private System.Threading.SendOrPostCallback previousOperationCompleted;

        private System.Threading.SendOrPostCallback getRowCountOperationCompleted;

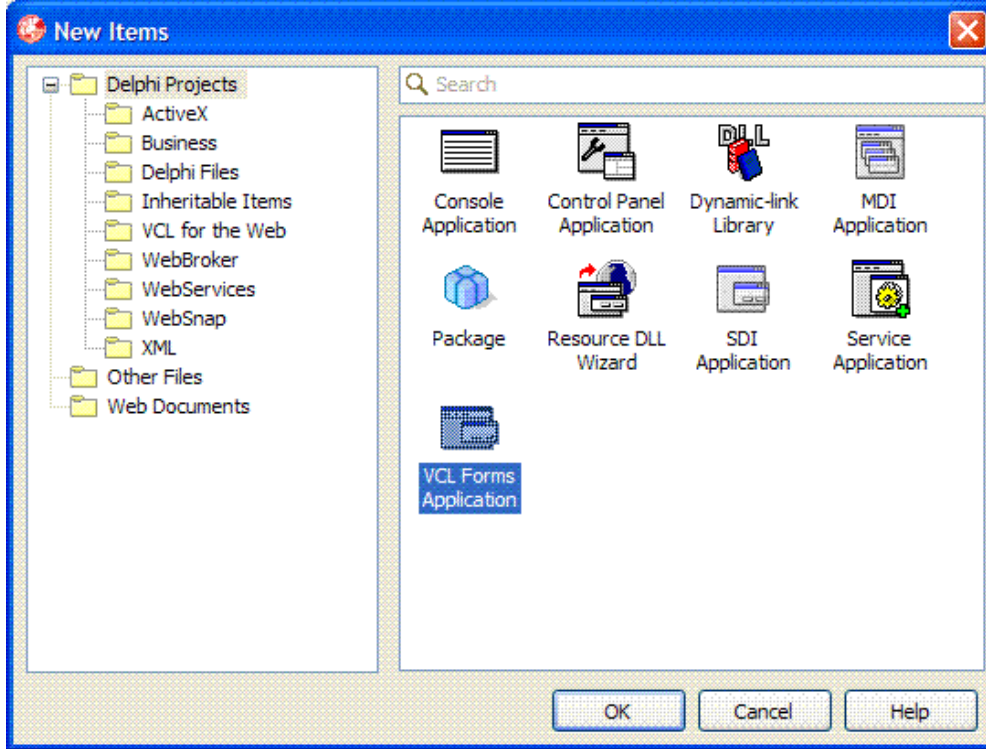
        private System.Threading.SendOrPostCallback lastOperationCompleted;

        private System.Threading.SendOrPostCallback searchOperationCompleted;
    }
}
```

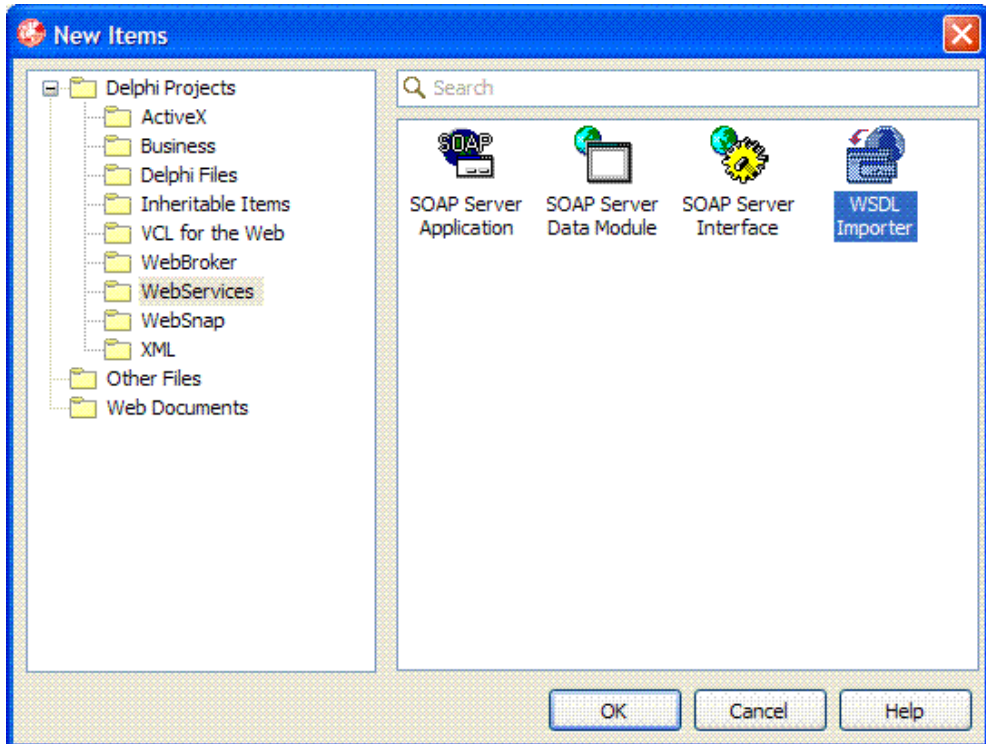
Bu örnekte sarıyla boyalı yerler UOD kullanıcı adı ve şifresinin belirtildiği yerler. Bu örnekte 3. satırdan itibaren 6 satır güvenlik ayarlarının yapıldığı satırlar. Yaratılan web servis istemci sınıfının policy olarak kullandığı nesne üzerinde UsernameToken ile kullanıcı-adi şifre güvenliği ekleniyor ve bu güvenlik kısmından Timestamp bölümü siliniyor. Java ile geliştirilen güvenlik altyapısında Timestamp kullanılmadığı için burada da bu bilgi SOAP mesajının başlık (header) bölümünden kaldırılıyor.

Delphi İle İstemci Yazılması

Delphi ile web servis istemci örneği için geliştirme ortamı olarak Delphi 2010 RAD Studio kullanılacaktır, ama burada anlatılanlar ve örnek proje büyük ihtimalle Delphi 2007'de çalışacaktır. Örnek için Delphi 2010 açılır ve yeni bir "VCL Forms Application" projesi yaratılır.



Yaratılan projenin ve içindeki dosyaların isimleri istendiği gibi ayarlanır. Bundan sonra web servis istemci kodlarını üretirmek için proje üzerinde File -> New -> Other seçilir ve açılan seçim penceresinden "Web Services" bölümünden "WSDL Importer" seçilir.



Açılan pencereden "WSDL Source" bölümüne wsdl adresi yazılır ve Next tuşuna basılır.

Bundan sonraki sayfalardan varsayılan ayarlarla sihirbaz bitirilir (duruma özel seçenekler istenirse ayarlanabilir). Bu sihirbaz sonunda web servis istemci kodları DataQueryService1.pas adında bir dosyaya üretilir. Bu adımdan sonra Delphi_WebServis.zip kataloğu bir yere açılır. Bu katalogdan çıkan WSSE.pas dosyası projeye eklenir. Form1 formuna bir tuş eklenir ve bu tuşun click metodu aşağıdaki örnekte olduğu gibi doldurulur:

```
procedure TForm3.Button1Click(Sender:TObject);
var
  Service: DataQueryService1.DataQueryService;
  Params: DataQueryParams;
  Result: DataQueryResult;
  Hdr:Security;
  Headers: ISOAPHeaders;
begin
  Service := DataQueryService1.GetDataQueryService(False,
  'http://172.16.12.163:9080/UnityWebTier/services/DataQueryService'.nil);
  Params:= DataQueryParams.Create;

  Hdr :=Security.Create;
  Hdr.MustUnderstand := False;
  Hdr.UsernameToken := UsernameToken.Create;
  Hdr.UsernameToken.Username := Username.Create;
  Hdr.UsernameToken.Username.Text := 'admin';
```

```

Hdr.UsernameToken.Password := Password.Create;

Hdr.UsernameToken.Password.Type_ := 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText';

Hdr.UsernameToken.Password.Text := 'logo';

// addrequired data to send to the web service

Headers := Service as ISOAPHeaders;

Headers.Send(Hdr);

Result:= Service.test(Params);

if(Result <> nil) then
    ShowMessage('Done for test method.')
else
    ShowMessage('Test method returned nil!');

Params.firm := 1;
Params.period := 1;
Headers.Send(Hdr);

Result:= Service.first('MMQOItemBrowser', Params, 10, False);

if(Result <> nil) then
    ShowMessage('Done for first method.')
else
    ShowMessage('First method returned nil!');

Hdr.Free;
Params.Free;

end;

```

Bu örnekte sarıyla boyalı yerler UOD kullanıcı adı ve şifresinin doldurulduğu yerler. Özetle yukarıdaki kod, SOAP başlık (header) kısmını elle oluşturuyor, içindeki UsernameToken'ın içine gerekli bilgileri dolduruyor ve web servis istemcisindeki her metodu çağırılmadan önce bu başlık bilgisini (Hdr nesnesi) yolluyor. Böylece yollanan SOAP mesajının başlık (header) bölümünde ilgili kullanıcı adı ve şifre bilgileri yollanmış oluyor. Bu örnek kod çalıştırılınca diğer örneklerde de olduğu gibi arka arkaya iki tane mesaj penceresi gelmesi gerekli ("Done for test method" ve "Done for first method"). Bu örneğe ait tüm proje kodlarına **Delphi_WebServisOrnek.zip** dosyasından erişilebilir.