

**CP5046 ICT PROJECT 1**

**Collectibles Auctions System on EBay Marketplace (CASE)**

**System Preparation Guide   
(version 1.0)**

Team NAME : TEAM AWESOME  
PROJECT TEAM :

Karimullah Bin Syed Ahmed (12749720)   
Vimal Adithann (12741594)   
Aswath Baskaran (12686276)

**Jan 2013**

Table of Contents

[1 Introduction 3](#_Toc345456523)

[2 Preparation for Implementation 4](#_Toc345456524)

[2.1 Installing the Integrated Development Environment 4](#_Toc345456525)

[2.2 Installing the MySQL/.NET Connector 5](#_Toc345456526)

[2.3 Installing the ODBC Connector 7](#_Toc345456527)

[2.4 Installing the MySQL Database Server 9](#_Toc345456528)

[2.5 Installing the MySQL Front End Tool 11](#_Toc345456529)

[2.6 Obtaining EBay API Keys 13](#_Toc345456530)

[2.7 Creating a Working Environment 14](#_Toc345456531)

[2.8 Adding Web References to eBay API 16](#_Toc345456532)

[2.9 Adding the Trading API 18](#_Toc345456533)

[2.10 Allowing access to make API Calls 19](#_Toc345456534)

[2.11 Allowing Access to Database Server 19](#_Toc345456535)

[3 Populating Tables using SQL Command Line 20](#_Toc345456536)

# 

# Introduction

This document discusses the installation and configuration required to setup the environment and parameters required to start the coding proper. The steps and be broken down into the following:

**Installation Phase**

* Installing the Visual Studio Express 2010 IDE
* Installing MySQL/.Net connector
* Installing MySQL/ODBC connector
* Installing the Database Server
* Installing the MySQL Front End Tool

**Configuration Phase**

* Obtaining EBay API Keys
* Referencing EBay APIs in the Visual Studio Express 2010 IDE
* Configuring IDE to make API Calls to EBay
* Configuring IDE to make connections to the Database Server
* Configuring ASP.Net Membership and Role Providers to work with system

**Database Preparation**

* Creating the MySQL Database schema
* Populating the tables with required baseline data.

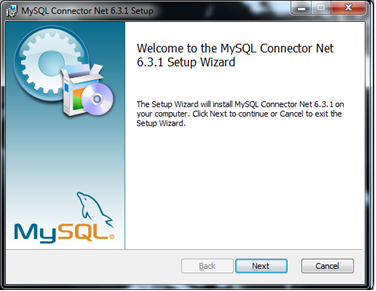
# Preparation for Implementation

## Installing the Integrated Development Environment

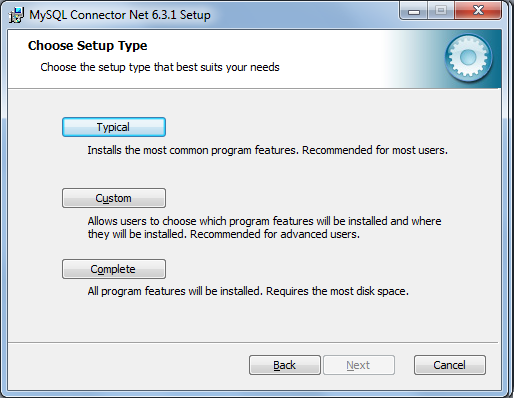
1. Download Visual Studio 2010 express
2. Install the Integrated Development Environment (IDE) at the default location with the default components.
3. Restart the machine to register all components with the operating system registry.

## Installing the MySQL/.NET Connector

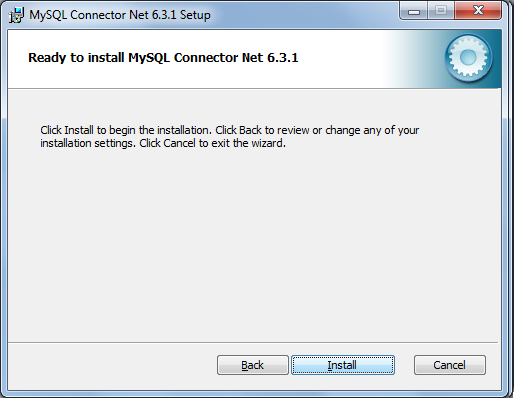
1. The MySQL/.Net connector serves as an interface between the MySQL Database and the IDE to allow transactions to take place. By default, only Microsoft SQL Server instances are allowed direct access from the IDE. Download the component at <http://download.softagency.net/MySQL/Downloads/Connector-Net/mysql-connector-net-6.3.1.zip>
2. Launch the installer for the connector and press next.



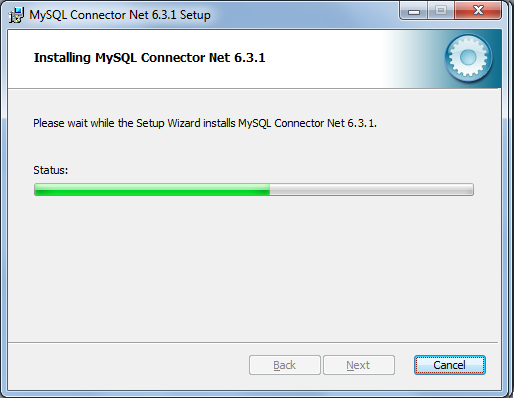
1. Select Typical and press next.



1. Press Install to start the installation process.



1. Wait for the installation to complete.

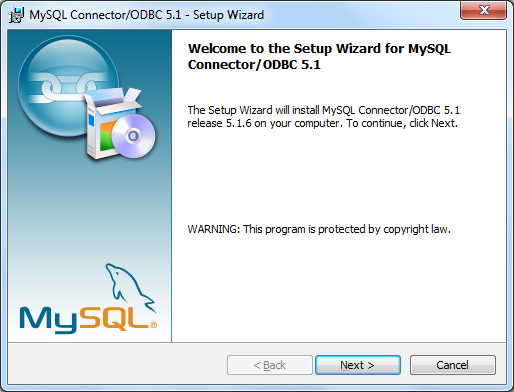


1. The MySQL Connector for .Net is now complete.

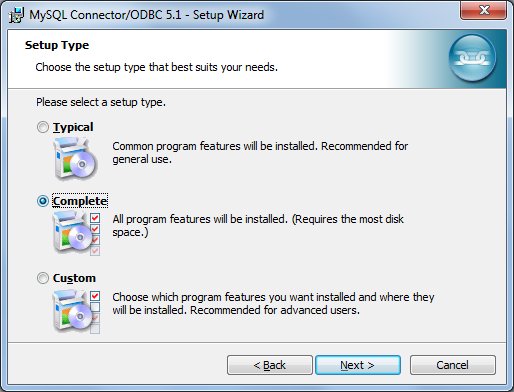
## Installing the ODBC Connector

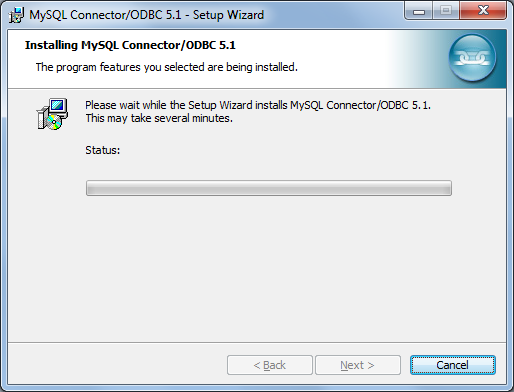
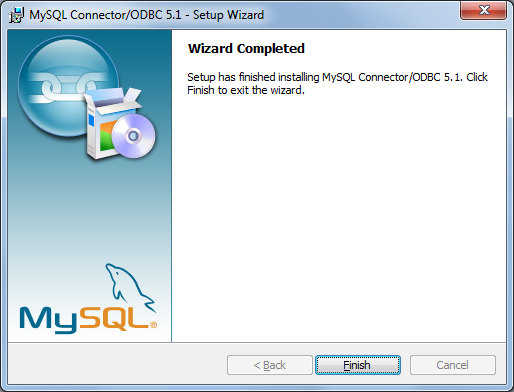
The MySQL Connector/ODBC is the name for the family of MySQL ODBC drivers (previously called MyODBC drivers) that provide access to a MySQL database using the industry standard Open Database Connectivity (ODBC) API. Typically, you need to install Connector/ODBC only on Windows machines.

1. Launch the ODBC connector installer



1. Select Complete and press Next

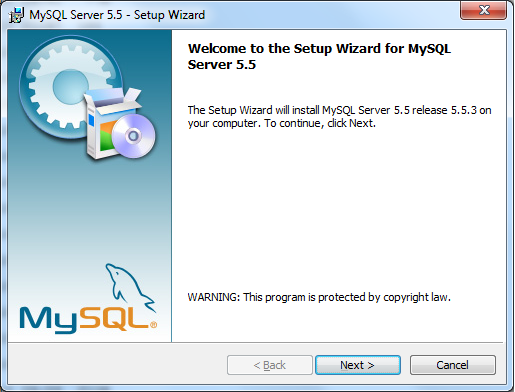


1. Select next to start the installation process.  
   
2. Press finish to complete the installation process.  
    

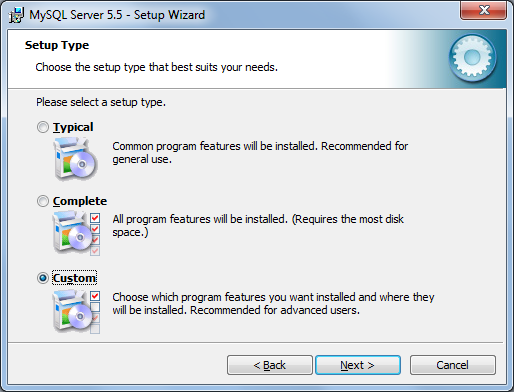
## Installing the MySQL Database Server

MySQL community edition database server can be found free of charge at http://download.softagency.net/MySQL/Downloads/

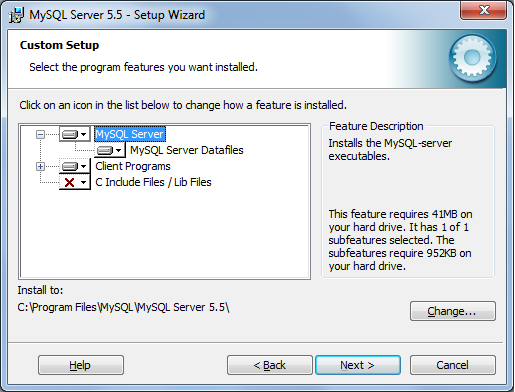
1. Click on the downloaded file to start the installation process and press next



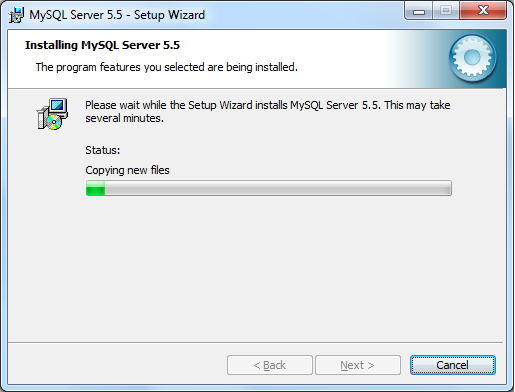
1. Select Custom and Press Next



1. Select the Components as shown and press next.



1. Wait for the installation to complete.

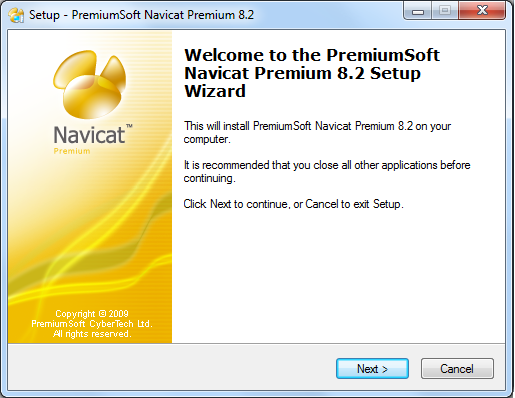


1. You have successfully installed MySQL Database Server

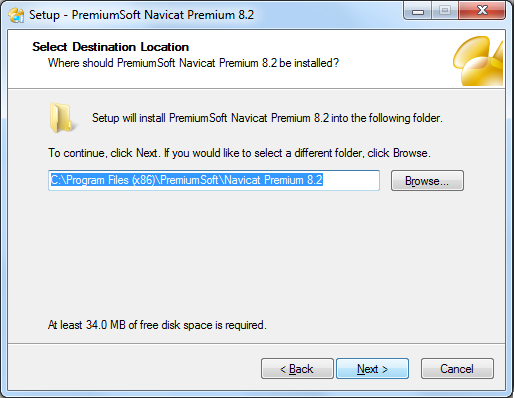
## Installing the MySQL Front End Tool

The Front end tools allow the provision for a developer to access the DBMS backend without the need for the use of the command line environment. For this project the NaviCat MySQL Client has been chosen. The application can be found at <http://download.navicat.com/download/navicat9_premium_en.exe>

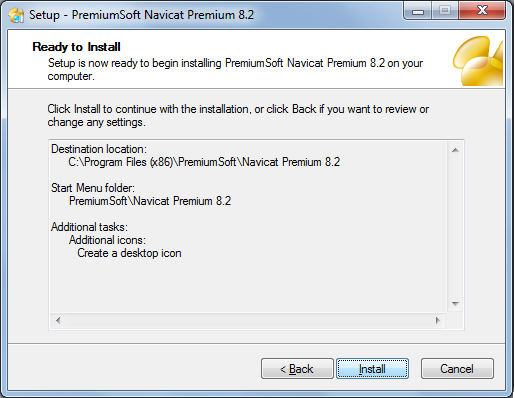
1. Start the installation process and click next



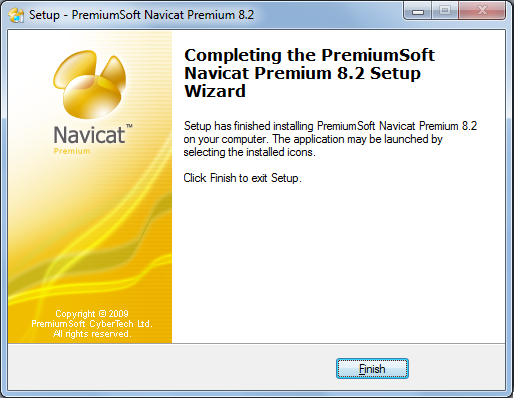
1. Select the installation location and press next



1. Select install to start the installation process.



1. Press finish to end the installation process



## Obtaining EBay API Keys

The EBay API requires keys authentication before it can be used to make system calls. EBay allows anyone with a developer account to generate keys.

1. Create a Developer account at  
   <https://developer.ebay.com/DevZone/account/Default.aspx>
2. Generate the Sandbox and Production Keys and the appropriate user token.

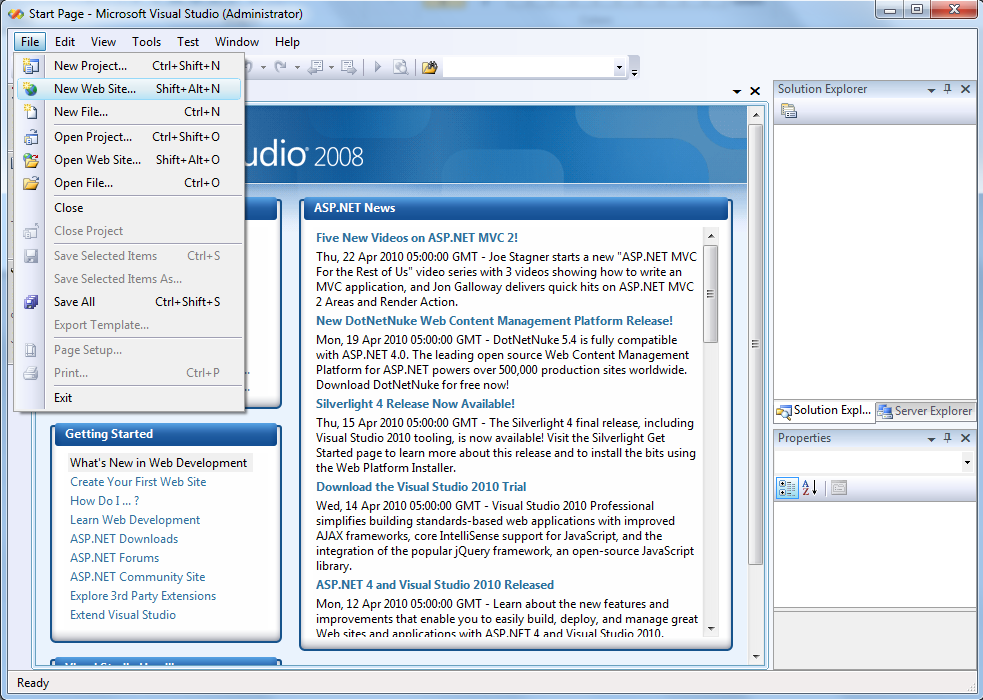
The following is a sample of the EBay keys:

* DEVID: 29049a4e-a486-4218-b9dd-b9aea3bf3c2
* AppID: KarimSye-8875-4be9-94ad-04f3217733c
* CertID: 3168e132-b1bc-4295-9021-af6e6ea56025
* User Token: AgAAAA\*\*AQAAAA\*\*aAAAAA\*\*1lK

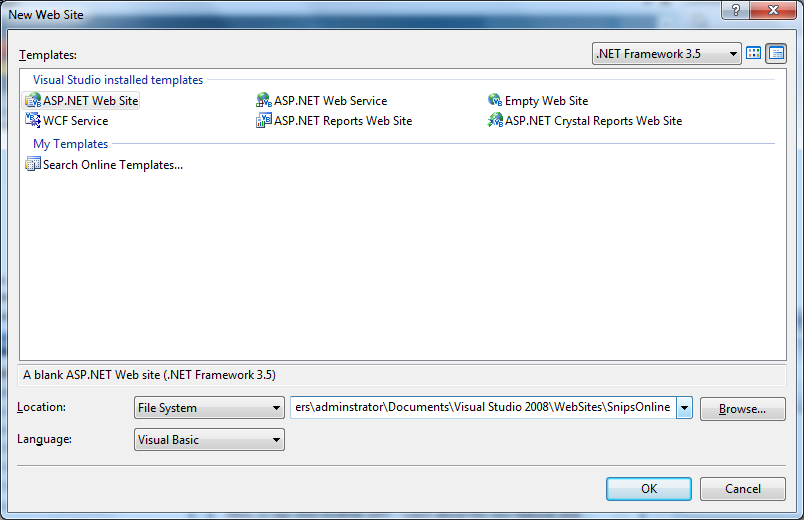
## Creating a Working Environment

The IDE, visual studio 2010 express also has to be configured for it to work with the database and EBay API. The following are the steps undertaken to integrate the IDE with the database server and the EBay API.

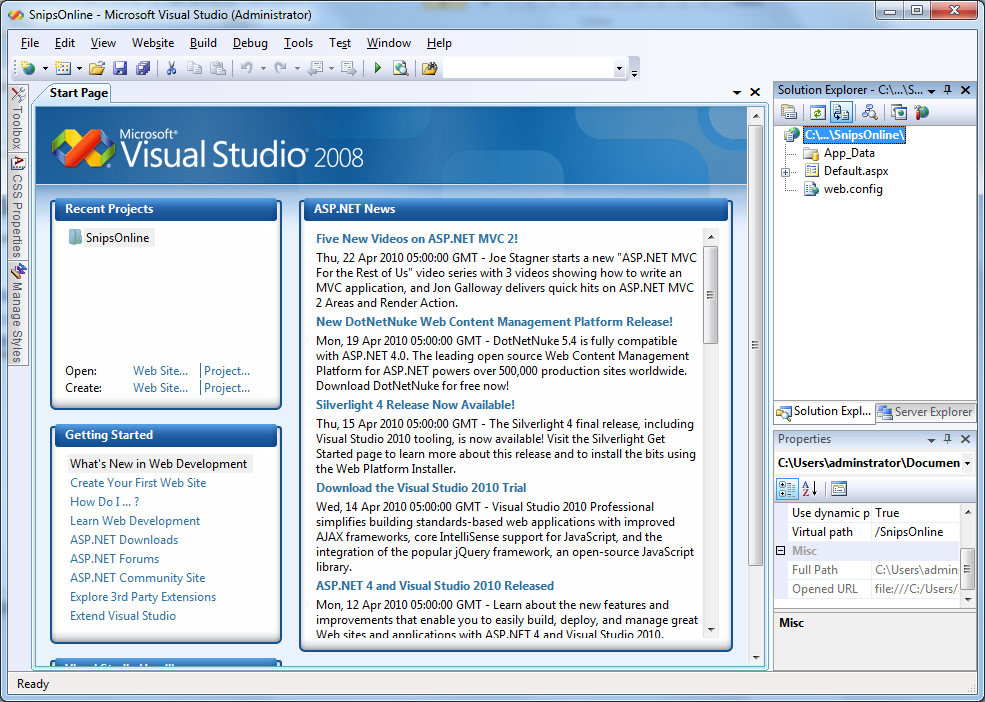
1. Open the visual studio 2010 ide and select a new website



1. Select “ASP.Net Web Site” and press next.



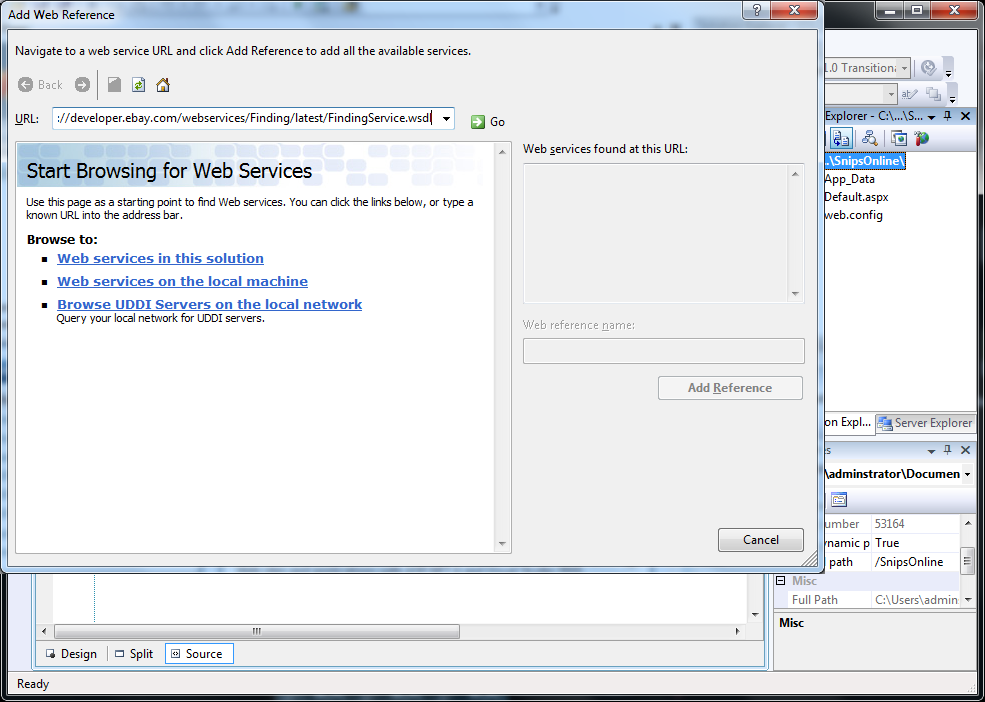
1. A new working profile will be created.



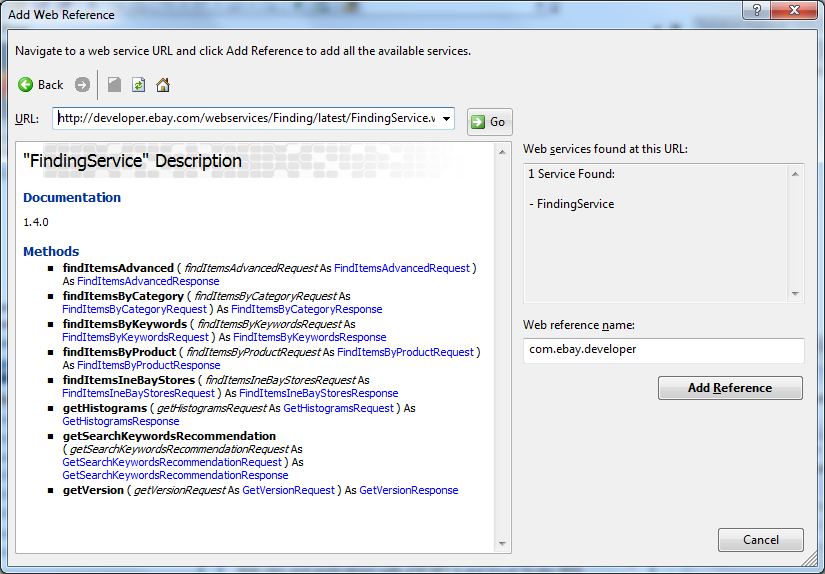
## Adding Web References to eBay API

The shopping api has to be added as a web reference since the .Net SDK only provides access to the trading api.

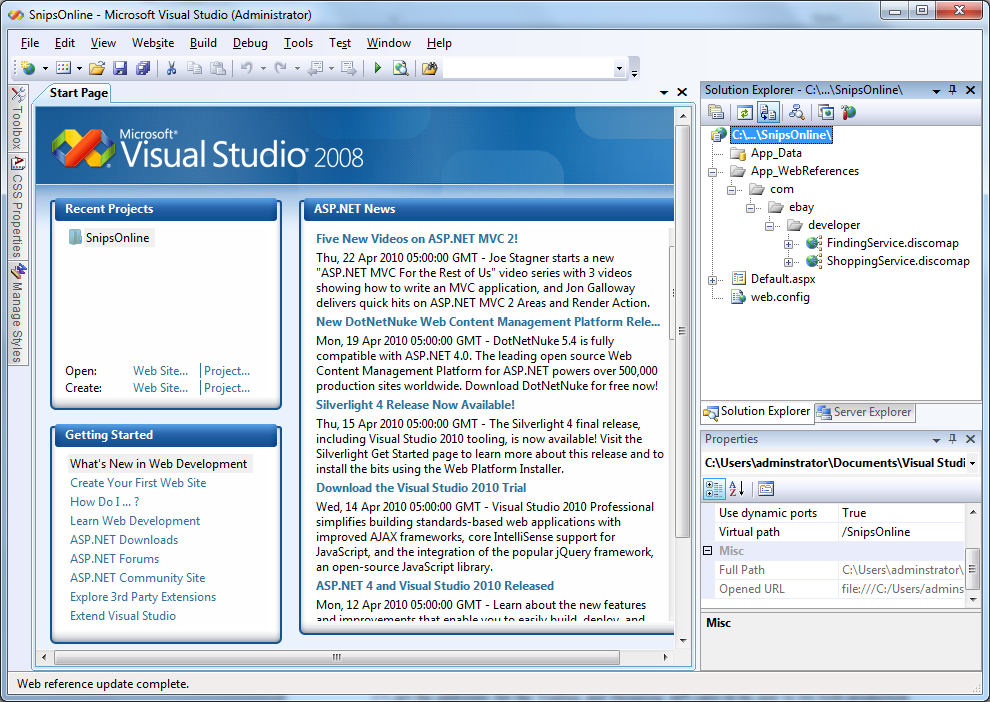
1. Right click on the project name and select “Add Web Reference”.
2. Add [http://developer.ebay.com/webservices/latest/ShoppingService*.*wsdl](http://developer.ebay.com/webservices/latest/ShoppingService.wsdl) and press go.



1. Click on add reference to add the Shopping api to the project.

**

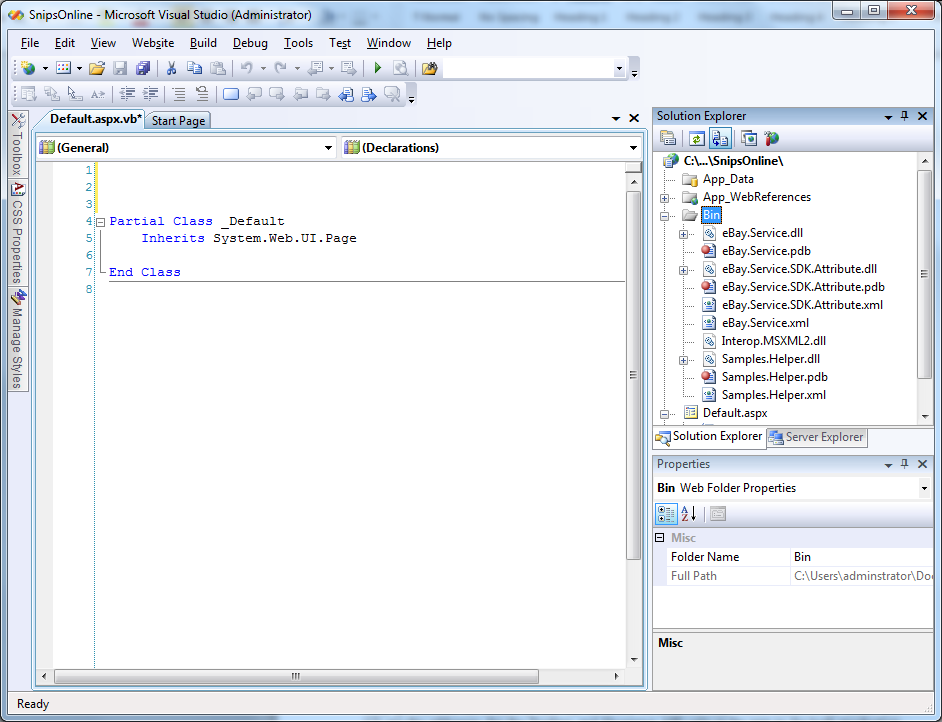
1. The web references will now appear in the com.ebay.developer folder in App\_WebReferences.



## Adding the Trading API

The Trading API is available for .Net environment via the SDK available on eBay’s developer’s portal. The SDK can be downloaded for free at [http://developer.ebay.com/DevZone/codebase/dotnetsdk/eBaySDKDotNET661Full2\_0.zip](http://developer.ebay.com/DevZone/codebase/dotnetsdk/eBaySDKDotNET661Full2_0.zip%20)

1. Drag the contents of the folder to the Bin folder of the project profile in the IDE.



## Allowing access to make API Calls

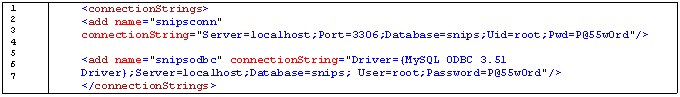
The web.config file has to be modified as such to enable the eBay api to make calls.

**Fig A8.1 – Editing the web.config file for eBay access**

Lines (3-6) depict the key and token pair that was allocated for the eBay sandbox environment. Calls made using these keys will only work in eBay sandbox. Lines (8-11) depict keys for the eBay production environment and can naturally be used for real world transactions. Lines (12-17) are the addresses for the Trading and Shopping API calls to be sent to for both production and the sandbox.

## Allowing Access to Database Server

To enable the website to bind itself to the correct database, the server’s connection strings must be supplied to the web server. Open the web.config file in your website and add connection strings.

**Fig A8.2 – Editing the web.config file for database connections to work**

Lines (2-3) depict the connection string to a database called “snips” and also the username and password. Lines (5-6) depict the same but for making ODBC connections. Both strings can be used interchangeably depending on the ASP.net component which makes the connection.

# Populating Tables using SQL Command Line

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | CREATE TABLE `ebay\_cat\_ver` (  `id` int(1) NOT NULL,  `version` int(5) NOT NULL,    PRIMARY KEY (`id`) ) |

**Fig 3.1 – The SQL Schema for creation of Ebay\_Cat\_Ver Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | CREATE TABLE `ebay\_parent\_cat` (  `CategoryID` varchar(8) NOT NULL,  `CategoryName` varchar(125) NOT NULL,  PRIMARY KEY (`CategoryID`) ) |

**Fig 3.2 – The SQL Schema for creation of Ebay\_Parent\_Cat Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 | CREATE TABLE `ebaychildcat` (  `CategoryID` varchar(8) NOT NULL,  `CategoryName` varchar(125) NOT NULL,  `parentCat` varchar(8) NOT NULL,  PRIMARY KEY (`CategoryID`),  KEY `ebaychildcat\_ibfk\_2` (`parentCat`),CONSTRAINT `ebaychildcat\_ibfk\_2` FOREIGN KEY   (`parentCat`) REFERENCES `ebayparentcat` (`CategoryID`) ON DELETE CASCADE ON UPDATE   CASCADE ) |

**Fig 3.3– The SQL Schema for creation of Ebay\_Child\_Cat Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | CREATE TABLE `shoppingcart` (  `custID` int(11) NOT NULL,  `itemID` varchar(15) NOT NULL,  `localValue` int(5) NOT NULL,  PRIMARY KEY (`custID`,`itemID`)) |

**Fig 3.4– The SQL Schema for creation of Ebay\_Child\_Cat Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 | CREATE TABLE `my\_aspnet\_membership` (  `userId` int(11) NOT NULL DEFAULT '0', `Email` varchar(128) DEFAULT NULL, `Comment` varchar(255) DEFAULT NULL, `Password` varchar(128) NOT NULL, `PasswordKey` char(32) DEFAULT NULL, `PasswordFormat` tinyint(4) DEFAULT NULL, `PasswordQuestion` varchar(255) DEFAULT NULL, `PasswordAnswer` varchar(255) DEFAULT NULL, `IsApproved` tinyint(1) DEFAULT NULL, `LastActivityDate` datetime DEFAULT NULL, `LastLoginDate` datetime DEFAULT NULL, `LastPasswordChangedDate` datetime DEFAULT NULL, `CreationDate` datetime DEFAULT NULL, `IsLockedOut` tinyint(1) DEFAULT NULL, `LastLockedOutDate` datetime DEFAULT NULL, `FailedPasswordAttemptCount` int(10) unsigned DEFAULT NULL, `FailedPasswordAttemptWindowStart` datetime DEFAULT NULL, `FailedPasswordAnswerAttemptCount` int(10) unsigned DEFAULT NULL, `FailedPasswordAnswerAttemptWindowStart` datetime DEFAULT NULL,  PRIMARY KEY (`userId`)) |

**Fig 3.5 – The SQL Schema for creation of MY\_ASPNET\_MEMBERSHIP Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | CREATE TABLE `cust\_purchase\_order` (  `itemID` varchar(15) NOT NULL, `userID` int(11) NOT NULL, `title` varchar(255) NOT NULL, `itemCost` double(10,2) NOT NULL, `shippingC` double(10,2) NOT NULL, `totalC` double(10,2) NOT NULL, `dateOrdered` date NOT NULL, `timeOrdered` time NOT NULL,  PRIMARY KEY (`itemID`),  KEY `userID` (`userID`),  CONSTRAINT `cust\_order\_ibfk\_1` FOREIGN KEY (`userID`) REFERENCES `customer` (`customerId`)  ON DELETE CASCADE ON UPDATE CASCADE ) |

**Fig 3.6 – The SQL Schema for creation of CUST\_PURCHASE\_ORDER Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | CREATE TABLE `cust\_selling\_order` (  `itemID` varchar(15) NOT NULL, `userID` int(11) NOT NULL, `title` varchar(255) NOT NULL, `subTitle` varchar(255) NOT NULL, `listPrice` double(10,2) NOT NULL, `buyItNowPRice` double(10,2) NOT NULL, `feesDeductible` double(10,2) NOT NULL, `startTime` datetime NOT NULL, `endTime` datetime NOT NULL, `itemDesc` text NOT NULL, `picURL` varchar(151) NOT NULL, `orderStatus` varchar(15) DEFAULT NULL, `salePrice` double(10,2) DEFAULT NULL,  PRIMARY KEY (`itemID`),  KEY `userID` (`userID`),  CONSTRAINT `cust\_selling\_order\_ibfk\_1` FOREIGN KEY (`userID`) REFERENCES `customer` (`customerID`)  ON DELETE CASCADE ON UPDATE CASCADE ) |

**Fig 3.7 – The SQL Schema for creation of CUST\_PURCHASE\_ORDER Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | CREATE TABLE `order\_feedback` (  `itemID` varchar(15) NOT NULL, `feedbackDesc` varchar(255) NOT NULL, `itemCost` varchar(20) NOT NULL, `itemQuality` varchar(20) NOT NULL, `itemShipping` varchar(20) NOT NULL, `positiveExperience` varchar(20) NOT NULL, `positiveExperiencePic` varchar(20) NOT NULL, `dateRated` date NOT NULL,  PRIMARY KEY (`itemID`),  CONSTRAINT `orders\_feedback\_ibfk\_1` FOREIGN KEY (`itemID`) REFERENCES `cust\_order`  (`itemID`) ON DELETE CASCADE ON UPDATE CASCADE ) |

**Fig 3.8 – The SQL Schema for creation of Order\_Feedback Table**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | CREATE TABLE `shoppingcart` (  `custID` int(11) NOT NULL,  `itemID` varchar(15) NOT NULL,  `localValue` int(5) NOT NULL,  `dateAdded` datetime NOT NULL,  PRIMARY KEY (`custID`,`itemID`),  CONSTRAINT `shoppingCartFK` FOREIGN KEY (`custID`) REFERENCES `customer` (`iD`) ON DELETE   CASCADE ON UPDATE CASCADE |

**Fig 3.9 – The SQL Schema for creation of ShoppingCart Table**

# 

THE END