Unit -5
Web service, Configuration & Deployment

Overview of XML

XML stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions −

- XML is extensible − XML allows you to create your own self-descriptive tags, or language, that suits your application.
- XML carries the data, does not present it − XML allows you to store the data irrespective of how it will be presented.
- XML is a public standard − XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

Introduction to XML in .Net XML is a cross-platform, hardware and software independent, text based markup language, which enables you to store data in a structured format by using meaningful tags. XML stores structured data in XML documents that are similar to databases. Notice that unlike Databases, XML documents store data in the form of plain text, which can be used across platforms. In an XML document, you specify the structure of the data by creating a DTD or an XML schema. When you include a DTD in an XML document, the software checks the structure of the XML document against the DTD. This process of checking the structure of the XML document is called validating. The software performing the task of validating is called a validating parser.

XML Usage

A short list of XML usage says it all −

- XML can work behind the scenes to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

What is Markup?

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. So what exactly is a markup language? Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

Following example shows how XML markup looks, when embedded in a piece of text −

```
<message>
  <text>Hello, world!</text>
</message>
```

Creating /Reading/Deleting XML Files :

https://www.youtube.com/watch?v=Qu3E-oncF3g

Web Services

- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

To summarize, a complete web service is, therefore, any service that −

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

Components of Web Services

The basic web services platform is XML + HTTP. All the standard web services work using the following components −

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of −

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

Example:

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

The steps to perform this operation are as follows −

- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

What is a Web.Config File?

A configuration file (web.config) is used to manage various settings that define a website. The settings are stored in XML files that are separate from your application code. In this way you can configure settings independently from your code. Generally a website contains a single Web.config file stored inside the application root directory. However there can be many configuration files that manage settings at various levels within an application.

Usage of configuration file:

ASP.NET Configuration system is used to describe the properties and behaviors of various aspects of ASP.NET applications. Configuration files help you to manage the many settings related to your website. Each file is an XML file (with the extension .config) that contains a set of configuration elements. Configuration information is stored in XML-based text files.

Benefits of XML-based Configuration files :

ASP.NET Configuration system is extensible and application specific information can be stored and retrieved easily. It is human readable.

- You need not restart the web server when the settings are changed in the configuration file. ASP.NET automatically detects the changes and applies them to the running ASP.NET application.
- You can use any standard text editor or XML parser to create and edit ASP.NET configuration files.

What Web.config file contains?

There are a number of important settings that can be stored in the configuration file. Some of the most frequently used configurations, stored conveniently inside Web.config file are:

- Database connections
- Caching settings
- Session States
- Error Handling
- Security

Different types of Configuration files

- Machine.config - Server or machine-wide configuration file
- Web.config - Application configuration files which deal with a single application

Machine.config File

Configuration files are applied to an executing site based on a hierarchy. There is a global configuration file for all sites in a given machine which is called Machine.config. This file is typically found in the C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG directory.

The Machine.config file contains settings for all sites running on the machine provided another .config further up the chain does not override any of these settings. Although Machine.config provides a global configuration option, you can use .config files inside individual website directories to provide more granular control. Between these two poles you can set a number of other .config files with varying degrees of applicable scope.

Application Configuration file (Web.config)

Each and Every ASP.NET application has its own copy of configuration settings stored in a file called Web.config. If the web application spans multiple folders, each sub folder has its own Web.config file that inherits or overrides the parent's file settings.

Global.asax File

- The Global.asax is also known as the ASP.NET application file and is used to serve application-level and session-level events.
- It allows us to write code that responds to global application events raised by ASP.NET or by HttpModules.
- These events fire at various points during the lifetime of a web application, including when the application domain is first created.
- The Global.asax file resides in the root directory of an ASP.NET-based application
- At run time, global.asax is parsed and compiled into a dynamically generated .NET Framework class derived from the HttpApplication base classThe Global.asax file is optional. If you do not define the file, the ASP.NET page framework assumes that you have not defined any application or session event handlers.

Basic Application Events:

.	**Method**	**Description**
1	Application_Start()	Application_Start() event occurs when the application starts, which is the first time it receives a request from any user. It doesn’t occur on subsequent requests. This event is commonly used to create or cache some initial information that will be reused later.

2	Application_End()	Application_End() event occurs when the application is shutting down, generally because the web server has restarted. You can insert cleanup code here.
3	Application_BeginRequest()	Application_BeginRequest() event occurs with each request the application receives, just before the page code is executed.
4	Application_EndRequest()	Application_EndRequest() event occurs with each request the application receives, just after the page code is executed.
5	Session_Start()	Session_Start() event occurs whenever a new user request is received and a session is started.
6	Session_End()	Session_End() event occurs when a session times out or is programmatically ended. This event is only raised if you are using in-process session state storage (the InProc mode, not the StateServer or SQLServer modes).
7	Application_Error()	Application_Error() event occurs in response to an un-handled error.

ASP.NET tracing enables you to view diagnostic information about a single request for an ASP.NET page. ASP.NET tracing enables you to follow a page's execution path, display diagnostic information at run time, and debug your application. ASP.NET tracing can be integrated with system-level tracing to provide multiple levels of tracing output in distributed and multi-tier applications.
There are two ways:

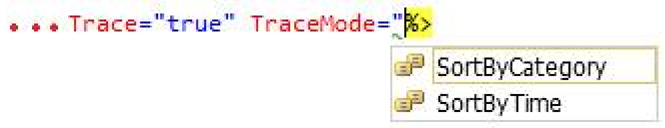
1. Page Level Tracing
2. Application Level Tracing

Page Level Tracing
We can control whether tracing is enabled or disabled for individual pages. If tracing is enabled, when the page is requested, ASP.NET appends to the page a series of tables containing execution details about the page request. Tracing is disabled by default in an ASP.NET application.
To enable Page Level Tracing follow the steps:

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Default.aspx.cs"Inherits="chat_Default" Trace="true"%>

Look at the above code, I'll be using Trace=true at the end.

ii) Optionally, we can use TraceMode attribute in above <%@ Page Title=""
Language="C#"...%> directive, for example:



```
... Trace="true" TraceMode=""%>
```

SortByCategory: Set TraceMode to SortByTime to sort trace messages in the order in which they are processed.

SortByTime: Set TraceMode to SortByCategory to sort trace messages by the categories.
iii) Now press F5 to run the application, you will see the immediate trace record on an existing page that you have set Trace and TraceMode.

Application Level Tracing

Instead of enabling tracing for individual pages, you can enable it for your entire application. In that case, every page in your application displays trace information. Application tracing is useful when you are developing an application because you can easily enable it and disable it without editing individual pages. When your application is complete, you can turn off tracing for all pages at once.

When you enable tracing for an application, ASP.NET collects trace information for each request to the application, up to the maximum number of requests you specify. The default number of requests is 10. You can view trace information with the trace viewer.
By default, when the trace viewer reaches its request limit, the application stops storing trace requests. However, you can configure application-level tracing to always store the most recent tracing data, discarding the oldest data when the maximum number of requests is reached.

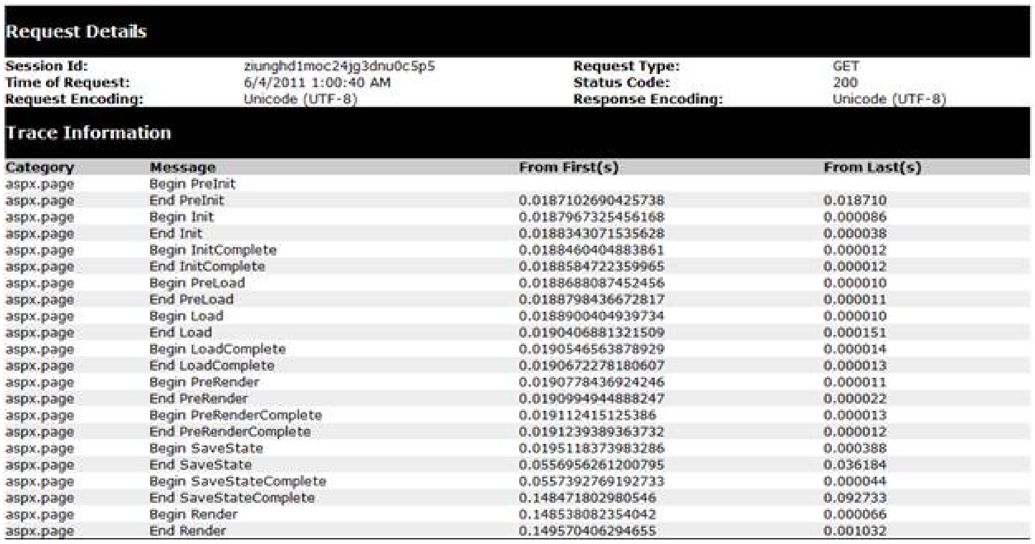
<u>To enable Application Level Tracing follow the steps:</u>

i) Delete your Page Level Tracking for better results.

ii) Open the Web.config file and add the following information to it; if there is not a Web.config file available then add a new one in the root.

```
<system.web>
   <trace enabled="true" pageOutput="true" requestLimit="40" localOnly="false"/>
 </system.web>
</configuration>
```

iii) The above code has many attributes used, find the detailed information about them below.
Enabled: Set it true to enable tracing for the application; otherwise, false. The default is false. You can override this setting for individual pages by setting the Trace attribute in the @ Page directive of a page to true or false.
PageOutput: Set it true to display trace both in pages and in the trace viewer (trace.axd); otherwise, false. The default is false.
RequestLimit: The number of trace requests to store on the server. The default is 10.
LocalOnly: Set it true to make the trace viewer (trace.axd) available only on the host Web server; otherwise, false. The default is true.



Request Details

Session Id:	ziunghd1moc24jg3dnu0c5p5	**Request Type:**	GET
Time of Request:	6/4/2011 1:00:40 AM	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

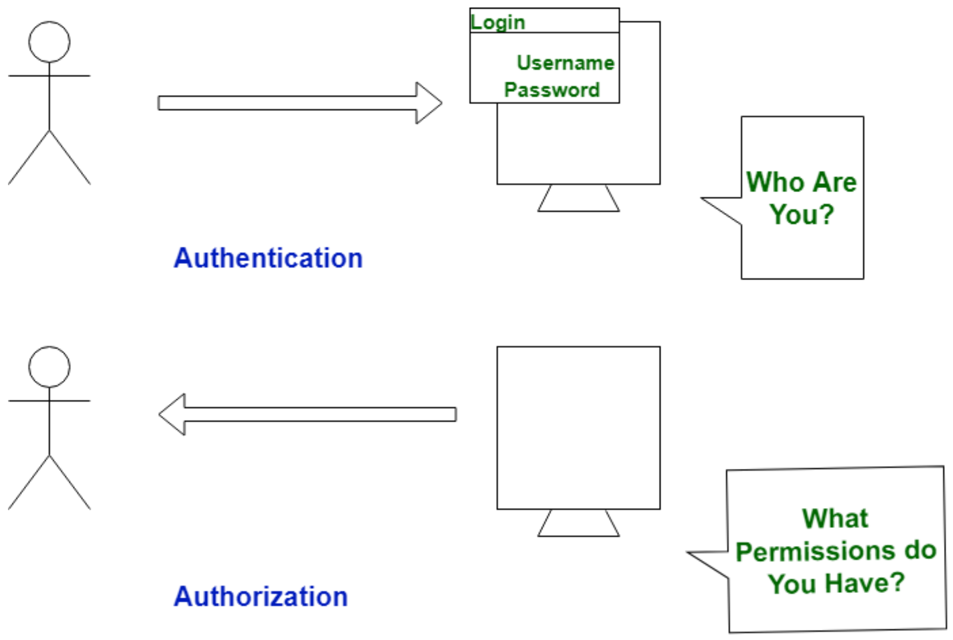
Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.0187102690425738	0.018710
aspx.page	Begin Init	0.0187967325456168	0.000086
aspx.page	End Init	0.0188343071535628	0.000038
aspx.page	Begin InitComplete	0.0188460404883861	0.000012
aspx.page	End InitComplete	0.0188584722359965	0.000012
aspx.page	Begin PreLoad	0.0188688087452456	0.000010
aspx.page	End PreLoad	0.0188798436672817	0.000011
aspx.page	Begin Load	0.0188900404939734	0.000010
aspx.page	End Load	0.0190406881321509	0.000151
aspx.page	Begin LoadComplete	0.0190546563878929	0.000014
aspx.page	End LoadComplete	0.0190672278180607	0.000013
aspx.page	Begin PreRender	0.0190778436924246	0.000011
aspx.page	End PreRender	0.0190994944888247	0.000022
aspx.page	Begin PreRenderComplete	0.019112415125386	0.000013
aspx.page	End PreRenderComplete	0.0191239389363732	0.000012
aspx.page	Begin SaveState	0.0195118373983286	0.000388
aspx.page	End SaveState	0.0556956261200795	0.036184
aspx.page	Begin SaveStateComplete	0.0557392769192733	0.000044
aspx.page	End SaveStateComplete	0.148471802980546	0.092733
aspx.page	Begin Render	0.148538082354042	0.000066
aspx.page	End Render	0.149570406294655	0.001032

What is trace.axd?

Simply, it is a viewer which has some features for displaying trace information for us. This file is not so necessary when you set PageOutput to true. To add this file to the application simply navigate to add a new file and name it trace.axd; it will automatically add the required code for the viewer.

Authentication and authorization

Both Authentication and Authorization area units are utilized in respect of knowledge security that permits the safety of an automatic data system. Each area unit terribly crucial topics usually related to the online as key items of its service infrastructure. However, each of the terms area units is completely different with altogether different ideas. whereas indeed, they're usually employed in an equivalent context with an equivalent tool, they're utterly distinct from one another. In the authentication process, the identity of users is checked for providing access to the system. While in the authorization process, a person's or user's authorities are checked for accessing the resources. Authentication is done before the authorization process, whereas the authorization process is done after the authentication process.



Authentication	**Authorization**
In the authentication process, the identity of users are checked for providing access to the system.	While in the authorization process, the person’s or user’s authorities are checked for accessing the resources.
In the authentication process, users or persons are verified.	While in this process, users or persons are validated.
It is done before the authorization process.	While this process is done after the authentication process.
It usually needs the user’s login details.	While it needs the user’s privilege or security levels.
Authentication determines whether the person is a user or not.	While it determines What permission does the user have?
Generally, transmit information through an ID Token.	Generally, transmit information through an Access Token.
The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of the user authentication process.	The OAuth 2.0 protocol governs the overall system of user authorization process.
Popular Authentication Techniques- ● Password-Based Authentication ● Passwordless Authentication ● 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) ● Single sign-on (SSO) ● Social authentication	Popular Authorization Techniques- ● Role-Based Access Controls (RBAC) ● JSON web token (JWT) Authorization ● SAML Authorization ● OpenID Authorization ● OAuth 2.0 Authorization

The authentication credentials can be changed in part as and when required by the user.	The authorization permissions cannot be changed by the user as these are granted by the owner of the system and only he/she has the access to change it.
The user authentication is visible at the user end.	The user authorization is not visible at the user end.
The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc.	The user authorization is carried out through the access rights to resources by using roles that have been pre-defined.
Example: Employees in a company are required to authenticate through the network before accessing their company email.	Example: After an employee successfully authenticates, the system determines what information the employees are allowed to access.

Error handling in asp.net

- Tracing - tracing the program execution at page level or application level.
- Error handling - handling standard errors or custom errors at page level or application level.
- Debugging - stepping through the program, setting break points to analyze the code

Customizing Error Page

To customize the default error page, one will have to change the default configuration settings of the application.

There are three error modes in which an ASP.Net application can work:

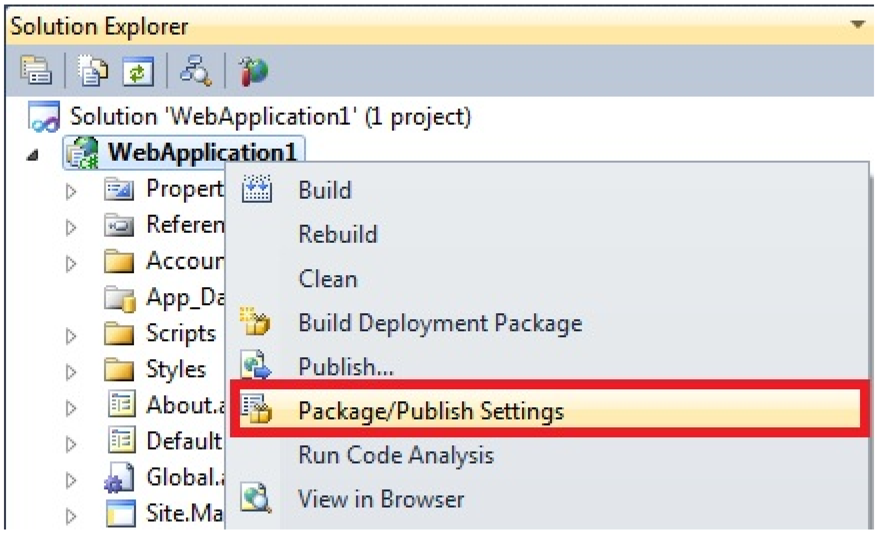
1. **Off Mode**
2. **On Mode**
3. **RemoteOnly Mode**

The Error mode attribute determines whether or not an ASP.Net error message is displayed. By default, the mode value is set to "RemoteOnly".

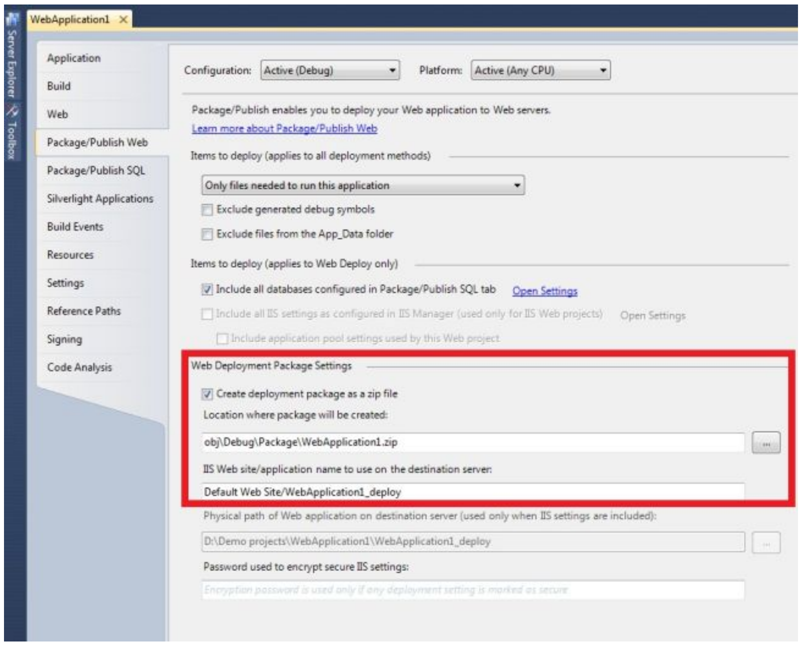
- **Off Mode**
  When the error attribute is set to "Off", ASP.Net uses its default error page for both local and remote users in case of an error.
- **On Mode**
  In case of "On" Mode, ASP.Net uses a user-defined custom error page instead of its default error page for both local and remote users. If a custom error page is not specified, ASP.Net shows the error page describing how to enable remote viewing of errors.
- **RemoteOnly**
  The ASP.Net error page is shown only to local users. Remote requests will first check the configuration settings for the custom error page or finally show an IIS error.

Deploying Application on Web Server

1, Right-click you asp.net web application project and select "package/Publish Setting".



2, Provide a few basic data in the configuration form (you may not want to change any setting in the first attempt). Select the project and click on properties.



Configuration: Active (Debug) Platform: Active (Any CPU)

Package/Publish enables you to deploy your Web application to Web servers.
Learn more about Package/Publish Web

Items to deploy (applies to all deployment methods)

Only files needed to run this application

- [ ] Exclude generated debug symbols
- [ ] Exclude files from the App_Data folder

Items to deploy (applies to Web Deploy only)

- [x] Include all databases configured in Package/Publish SQL tab Open Settings
- [ ] Include all IIS settings as configured in IIS Manager (used only for IIS Web projects) Open Settings
  - [ ] Include application pool settings used by this Web project

Web Deployment Package Settings

- [x] Create deployment package as a zip file

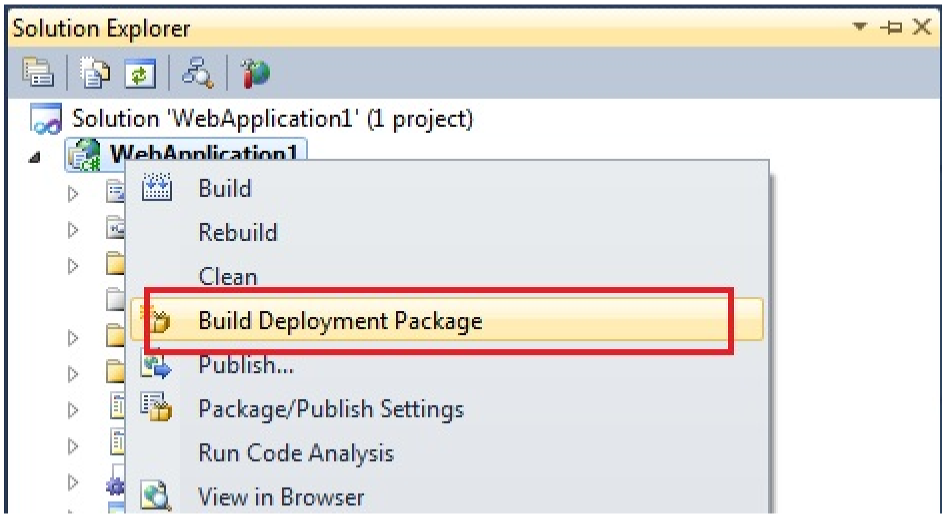
Location where package will be created:
obj\Debug\Package\WebApplication1.zip

IIS Web site/application name to use on the destination server:
Default Web Site/WebApplication1_deploy

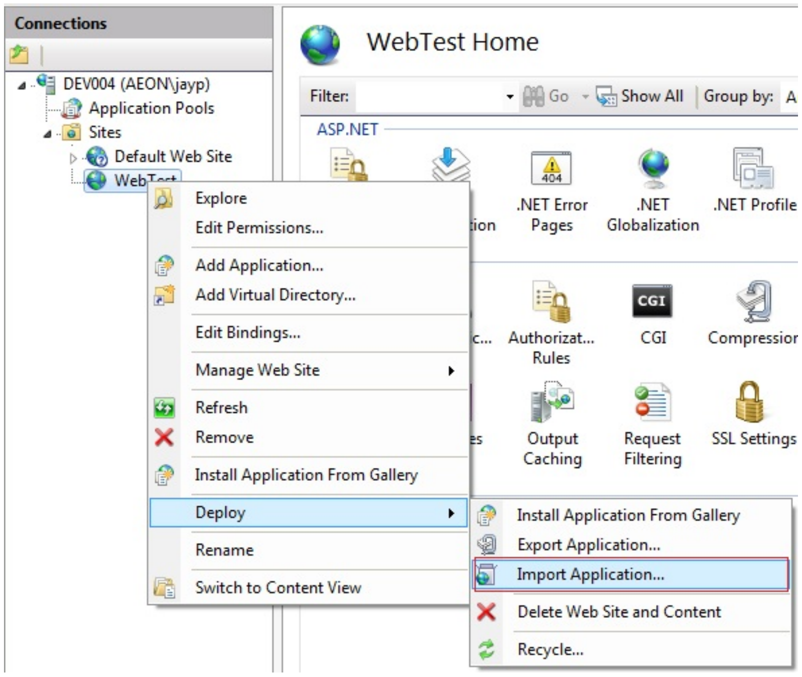
Physical path of Web application on destination server (used only when IIS settings are included):
D:\Demo projects\WebApplication1\WebApplication1_deploy

Password used to encrypt secure IIS settings:
Encryption password is used only if any deployment setting is marked as secure

3, Right-click on the project and select "Build Deployment Package". Visual Studio will build the deployment package (zip file) and will store it in the default location.



4, Open IIS manager(type "inetmgr" in Run command), Right-click on the target web site (where you want to deploy your Asp.net application), Select deploy=>Import application.

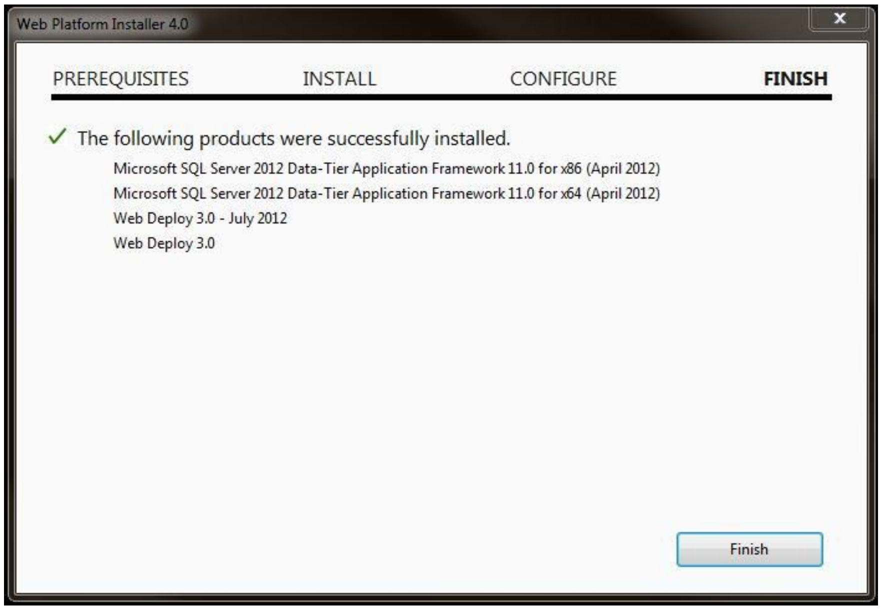


Important Note

If your IIS doesn't have this Deploy option then you need to install Extension of the IIS.

So that visits this web site:
http://www.iis.net/downloads/microsoft/web-deploy Click on Install this Extension.

install that extension and you will get this message.

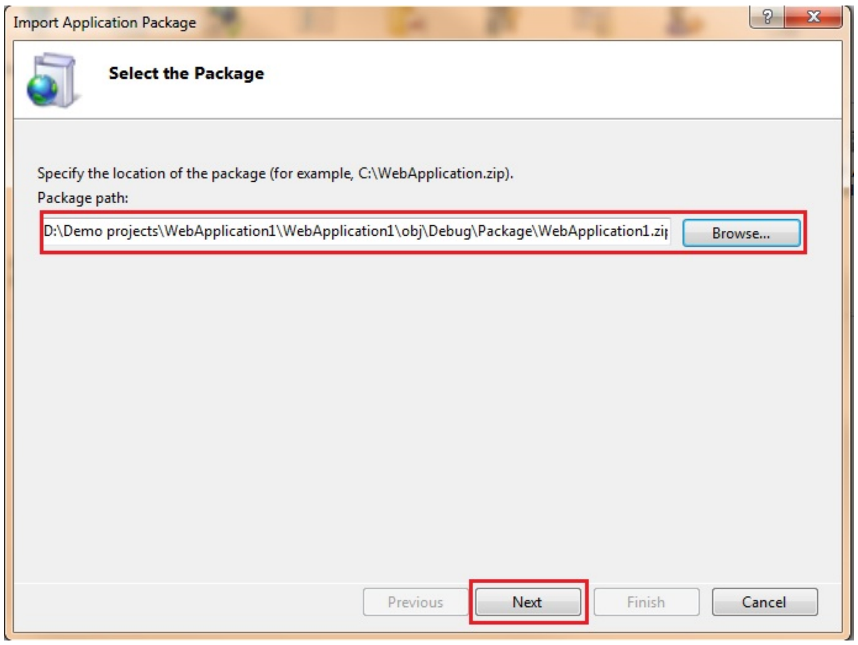


✓ The following products were successfully installed.

Microsoft SQL Server 2012 Data-Tier Application Framework 11.0 for x86 (April 2012)
Microsoft SQL Server 2012 Data-Tier Application Framework 11.0 for x64 (April 2012)
Web Deploy 3.0 - July 2012
Web Deploy 3.0

Then will be able to see the deploy option in IIS.

5, Browse the deployment package file (the Zip file) to select it and select "Next" buttons on wizard steps and click the "**finish**" button on the last step. You can alter the connection string in wizards steps.



Specify the location of the package (for example, C:\WebApplication.zip).
Package path:
D:\Demo projects\WebApplication1\WebApplication1\obj\Debug\Package\WebApplication1.zi