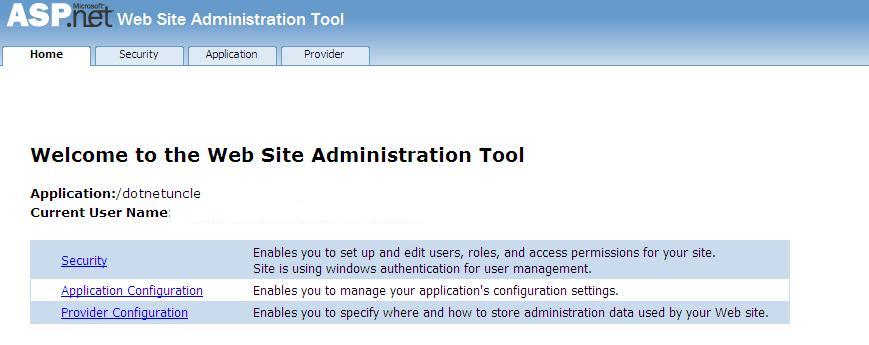
**Describe the security authentication flow and process in ASP.NET?**

When a user requests a web page, there exists a process of security too, so that every anonymous user is checked for authentication before gaining access to the webpage. The following points are followed in the sequence for authentication when a client attempts a page request:  
  
\* A .aspx web page residing on an IIS web server is requested by an end user  
\* IIS checks for the user's credentials  
\* Authentication is done by IIS. If authenticated, a token is passed to the ASP.NET worker process along with the request  
\* Based on the authentication token from IIS, and on the web.config settings for the requested resource, ASP.NET impersonates the end user to the request thread. For impersonation, the web.config **impersonate** attribute's value is checked.

**What is the Website Administrative Tool in ASP.NET 2.0 ?**

In ASP.NET 2.0, while using Visual Studio 2005 Express Edition or above, the development IDE provides an interface for editing the web.config rather than manually editing the web.config.   
  
In the IDE, click on "Website" and then on "ASP.NET Configuration". This shall open the Website configuration tool. Note that the Web Site Administration Tool is a set of prebuilt ASP.NET 2.0 webpages and resources that are located within the C:\Inetpub\wwwroot\aspnet\_webadmin\2\_0\_40607 directory.   
  
Website Administration Tool image below   


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | **What is Authentication? What are the different types of Authentication?**  In a client-server environment, there are plenty of cases where the server has to interact and identify the client that sends a request to the server. **Authentication** is the process of determining and confirming the identity of the client.   If a client is not successfully identified, it is said to be anonymous.  **Types of Authentication**  Windows Authentication Forms Authentication Passport Authentication  Essentially the Windows Authentication and Forms Authentication are the famous ones, as Passport Authentication is related to a few websites (like microsoft.com, hotmail.com, msn.com etc. only).  **Windows Authentication** is implemented mostly in Intranet scenarios. When a browser (client) sends a **Request** to a server where in windows authentication has been implemented, the initial request is anonymous in nature. The server sends back a **Response** with a message in HTTP Header. This Prompts a Window to display a Modal Dialog Box on the browser, where the end user may enter the "User name" and "Password".   The end user enters the credentials, which are then validated against the User Store on the Windows server. Note that each user who access the Web Application in a Windows Authentication environment needs to have a Windows Account in the company network.  **How to avoid or disable the modal dialog box in a Windows Authentication environment?**  By enabling the Windows Integrated Authentication checkbox for the web application through settings in IIS.   **Forms Authentication** is used in Internet based scenarios, where its not practical to provide a Windows based account to each and every user to the Web Server. In a Forms Authentication environment, the user enters credentials, usually a User Name and a corresponding Password, which is validated against a User Information Store, ideally a database table.   **Forms Authentication Ticket** is the cookie stored on the user's computer, when a user is authenticated. This helps in automatically logging in a user when he/she re-visits the website. When a Forms Authentication ticket is created, when a user re-visits a website, the Forms Authentication Ticket information is sent to the Web Server along with the HTTP Request. What is Authorization in ASP.NET? **Authorization**, in simple words means "which user can access which resource on a web server". Authentication of users may be set in the web.config file. See web.config snippet below...  <authorization>    <deny users="?" /> <!-- Allow all users -->      <!-- <allow users="[user list separated by commas]"      roles="[role list separated by commas]"/>      <deny users="[user list separated by commas]"      roles="[role list separated by commas]"/>    --> </authorization>  Based on the user and the role, access to different folders across the website may be controlled using the authorization feature of ASP.NET  **What is IIS Metabase? How to edit IIS metabase? How to backup IIS metabase file?**  **IIS Metabase** - sounds like geek stuff right! What is IIS Metabase??? In the simplest words, IIS metabase is the repository of the configuration values that are set in the Internet Information Server (IIS). The IIS metabase in an XML file. It may be controlled through program or manually too.   In order to edit IIS metabase entries, the user needs to have administrative rights on the system. To do this, in run window, type "inetmgr". Browse to "Local Computer" and right click it. Click on "Properties". Select the "Enable Direct Metabase Edit" check box.  Many times, due to the existence of multiple versions of .NET framework, some settings in the IIS metabase may get affected and cause your program not to run. For such scenarios, you may take a backup of the IIS metabase XML file and recover it. To create a portable backup, open run window and type "inetmgr". Next browse to "Local Computer" and go to "All Tasks". Next, click on Backup/Restore Configuration. Next, click on "Create Backup". In the textbod box for Configuration Backup name, type a name for your backup file. Also select the encrypt backup option and type a password. To finish the process, click "OK".   Search for Metabase.xml file on your IIS Web Server. Describe the Provider Model and Personalization in ASP.NET 2.0? The Provider model in ASP.NET 2.0 is based on the Provider Design Pattern that was created in the year 2002 and later implemented in the .NET Framework 2.0.  The Provider Model supports **automatic** creation of users and their respective roles by creating entries of them directly in the SQL Server (May even use MS Access and other custom data sources). So actually, this model also supports automatically creating the user table's schema.   The Provider model has 2 security providers in it: **Membership provider and Role Provider**. The membership provider saves inside it the user name (id) and corresponding passwords, whereas the Role provider stores the Roles of the users.   For SQL Server, the SqlMembershipProvider is used, while for MS Access, the AccessMembershipProvider is used. The Security settings may be set using the website adminstration tool. Automatically, the AccessMembershipProvider creates a Microsoft Access database file named aspnetdb.mdb inside the application's App\_Data folder. This contains 10 tables.   **ASP.NET 2.0 Personalization** - Personalization allows information about visitors to be persisted on a data store so that the information can be useful to the visitor when they visit the site again. In ASP.NET 2.0, this is controlled by a Personalization API. Before the Personalization Model came into existence, the prior versions of ASP.NET used of the old **Session** object to take care of re-visits. Now comes the **Profile object**.   In order to use a Profile object, some settings need to be done in web.config. The example below shall explain how to use a profile object:  //Add this to System.Web in web.config  <profile> <properties>   <add name="FirstName" type="System.String"/>    <add name="LastName" type="System.String"/> </properties> </profile>  'In Page\_Load event, add the following...   If Profile.FirstName <> "" Then     Panel1.Visible = False     Response.Write("Welcome Back Dear :, " & Profile.FirstName & ", " & Profile.LastName)  Else     Panel1.Visible = True End If  'Here is the code how to save the profile properties in an event to save it Profile.FirstName = txtFirstName.Text  Profile.LastName = txtLastName.Text |  |  |  |  |  | | --- | --- | --- | |  |  |  | |  |