INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

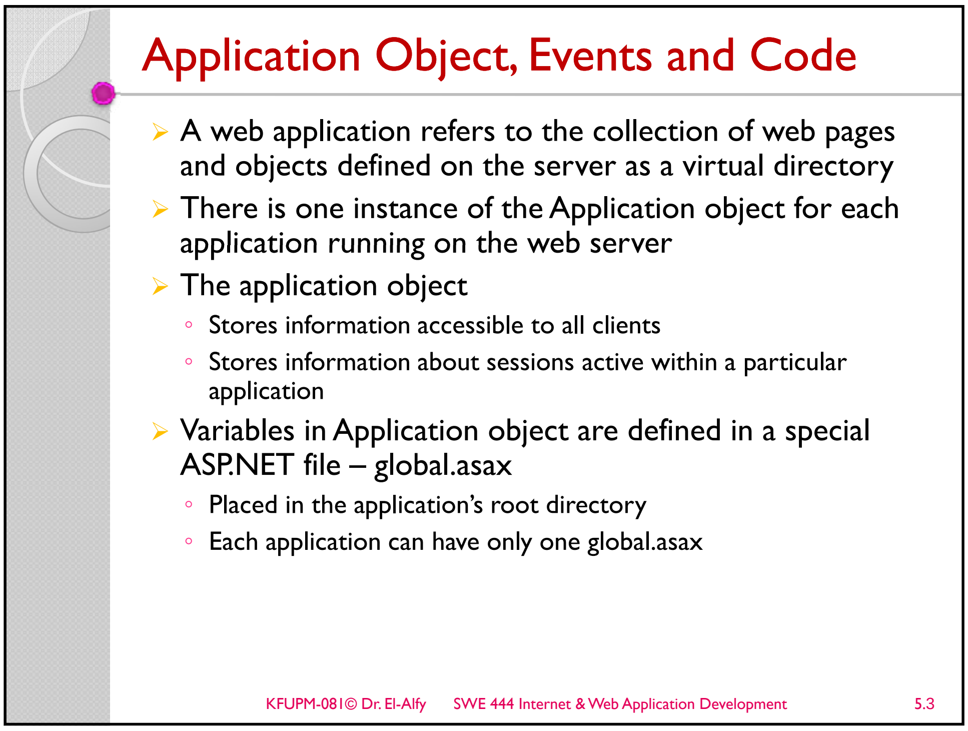
Module 5.5: More About ASP.NET

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

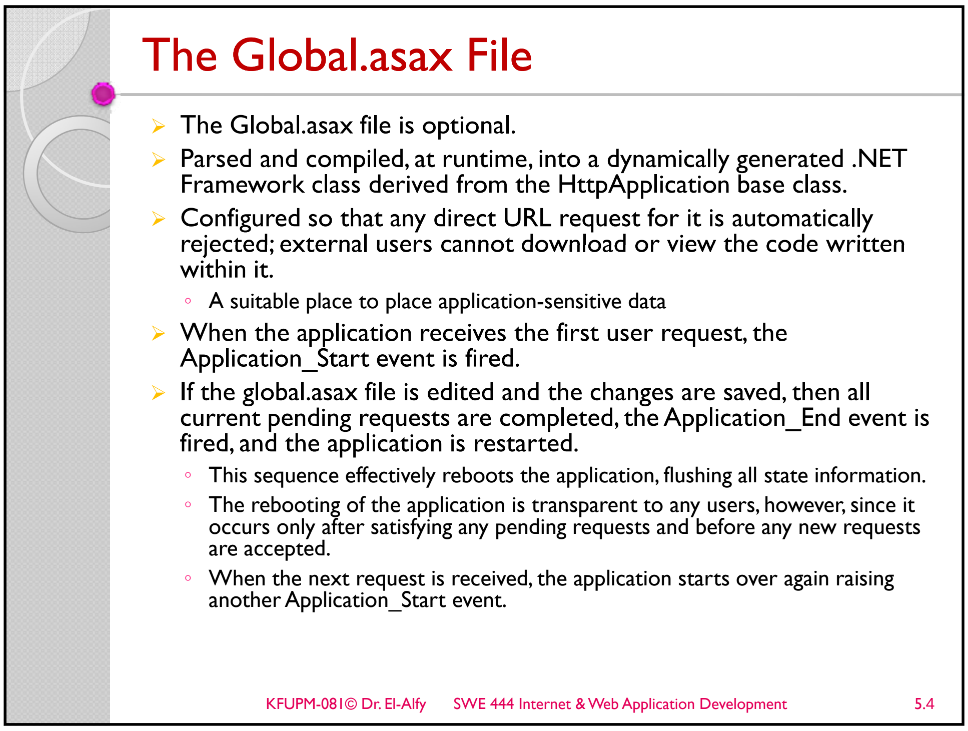
- Objectives
  - Learn how to use Application object
  - Learn how to make external configurations
  - Use authentication to control access to the application

- Outline
  - Application Object
  - Application Conf. Settings
  - Forms Authentication
  - Stored Procedures



Application Object, Events and Code

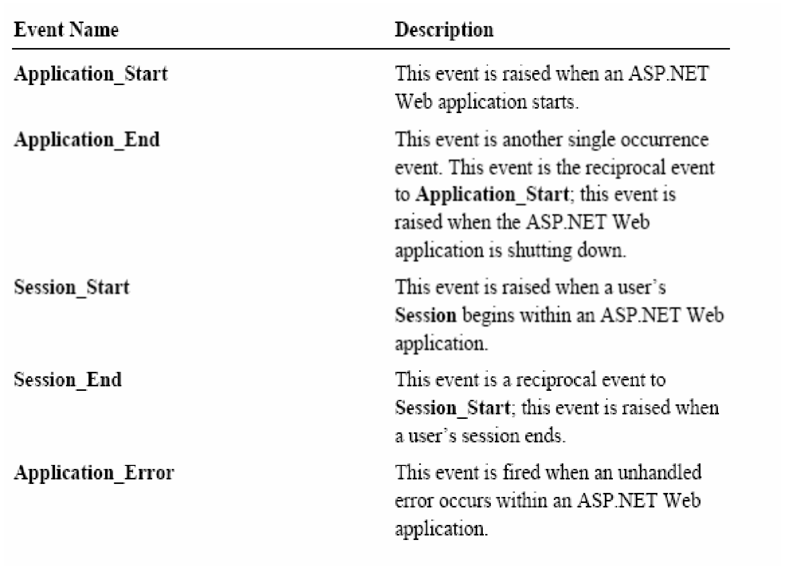
- A web application refers to the collection of web pages and objects defined on the server as a virtual directory
- There is one instance of the Application object for each application running on the web server
- The application object
  - Stores information accessible to all clients
  - Stores information about sessions active within a particular application
- Variables in Application object are defined in a special ASP.NET file – global.asax
  - Placed in the application's root directory
  - Each application can have only one global.asax



The Global.asax File

- The Global.asax file is optional.
- Parsed and compiled, at runtime, into a dynamically generated .NET Framework class derived from the HttpApplication base class.
- Configured so that any direct URL request for it is automatically rejected; external users cannot download or view the code written within it.
  - A suitable place to place application-sensitive data
- When the application receives the first user request, the Application_Start event is fired.
- If the global.asax file is edited and the changes are saved, then all current pending requests are completed, the Application_End event is fired, and the application is restarted.
  - This sequence effectively reboots the application, flushing all state information.
  - The rebooting of the application is transparent to any users, however, since it occurs only after satisfying any pending requests and before any new requests are accepted.
  - When the next request is received, the application starts over again raising another Application_Start event.

Application Events



Event Name	Description
Application_Start	This event is raised when an ASP.NET Web application starts.
Application_End	This event is another single occurrence event. This event is the reciprocal event to **Application_Start**; this event is raised when the ASP.NET Web application is shutting down.
Session_Start	This event is raised when a user's **Session** begins within an ASP.NET Web application.
Session_End	This event is a reciprocal event to **Session_Start**; this event is raised when a user's session ends.
Application_Error	This event is fired when an unhandled error occurs within an ASP.NET Web application.

Application Code – global.asax

```
<script language="c#" runat="server" >
 void Application_OnStart(Object obj,
  EventArgs e){
      Application[“timeKeeper"]= "";
      Application[“visitCounter"]= 0;
 }
</script>
```

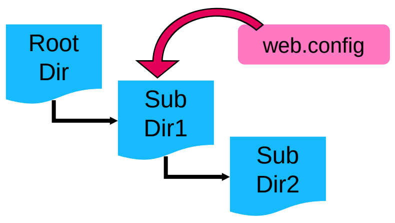
Application Configuration Settings

- In classic ASP all Web site related information was stored in the metadata of IIS.
  - Disadvantage: remote Web developers couldn't easily make Web-site configuration changes.
- Such configuration changes need to be done through the IIS admin tool
  - Your Web host will likely charge you a fee to do this for you.
- With ASP.NET, these settings are directly under developer control
  - Placed into an XML-formatted text file (Web.config) that resides in the Web site's root directory.
- Goal of ASP.NET configuration (web.config):
  - Provide extensible configuration for admins & developers to hierarchically apply settings for an application

Hierarchy of .config Files

- Multiple .config files can, and typically do, exist on a single system.
- System-wide configuration settings for the .NET Framework are defined in the Machine.config file.
  - Placed in

%SystemRoot%\Microsoft.NET\Framework\%VersionNumber%\CONFIG\ folder.
- Configuration files can be stored in application folders
  - Configuration system automatically detects changes
- Hierarchical configuration architecture
  - Applies to the actual directory and all subdirectories



Creating a web.config File

- At the root level of web.config is the <configuration> tag.
- Inside this tag you can add a number of other tags
  - The most common and useful one being the system.web tag, where you will specify most of the Web site configuration parameters.
- However, to specify application-wide settings you use the <appSettings> tag.
  - Inside of this tag you can specify zero to many settings by using the <add ... /> tag.
  - For example, if we wanted to add a database connection string parameter we could have a Web.config file like:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
       <add key="connectionString"
          value="Data Source=localhost;Initial Catalog=pubs;Integrated Security=True" />
    </appSettings>
   <system.web>
         ...
    </system.web>
</configuration>
```

- Retrieve as: **string str = ConfigurationSettings.AppSettings["connectionString"];**

Forms Authentication

- Like IIS, ASP.NET has its own authentication methods
- When IIS receives a request for an ASP.NET resource, like .aspx file
  - It performs its own authentication (if the web app is configured in IIS to do so)
  - And then passes on the request and a security token to the ASP.NET runtime
- ASP.NET supports the following authentication modes
  - None – ASP.NET relies on IIS for authentication
  - Windows – treats the user identity supplied in the security token by IIS as the authenticated user
  - Forms – allows authentication via login forms of the Web Application
  - Passport – uses the Microsoft Passport system running on a separate Passport server for authentication
- Authentication mode is specified within the `authentication` element of the application's Web.config file:

```
<system.web>
   …
  <authentication mode="Windows" />
</system.web>
```

Example I

- Consider the following configuration file:

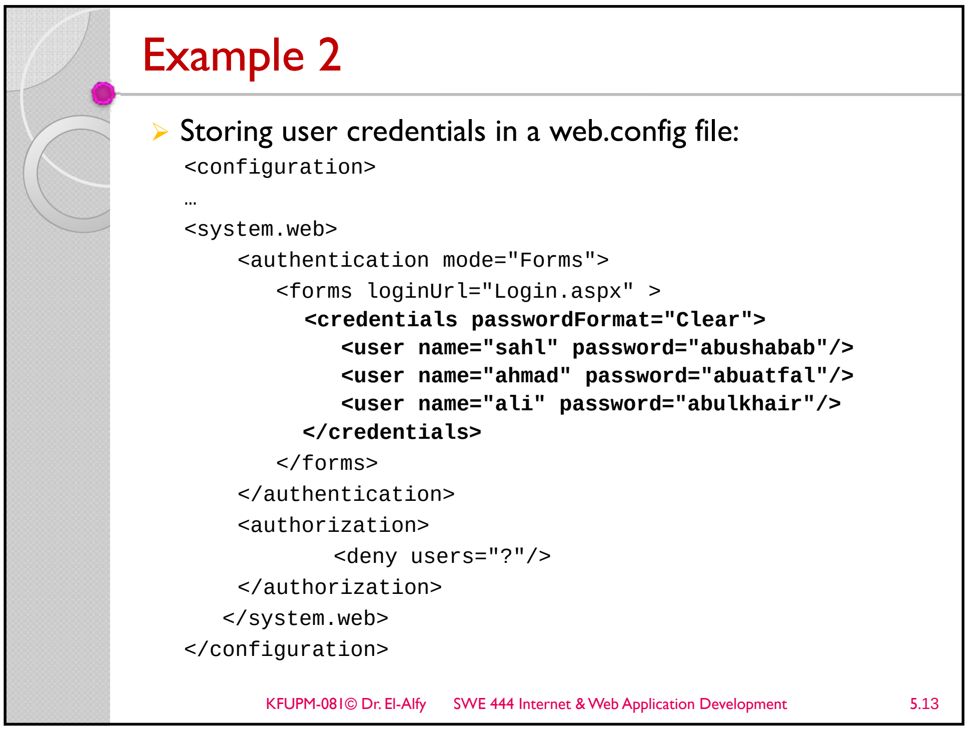
```
<configuration>
…
<system.web>
     <authentication mode="Forms">
              <forms loginUrl="Login.aspx" >
              </forms>
     </authentication>
     <authorization>
              <deny users="?"/>
     </authorization>
   </system.web>
</configuration>
```

- Suppose you place the above in the folder containing your Web application files, then
  - An attempt by a user to access any file in the Web application now will be redirected to Login.aspx automatically
  - **<deny users="?" />** specifies that all unauthenticated users are denied access to ASP.NET resources in the site
  - Users information can be hard-coded in an event handler, inside a web.config file or, more appropriately, inside a database

Example I (cont.)

- The authentication logic can be hard-coded as follows:

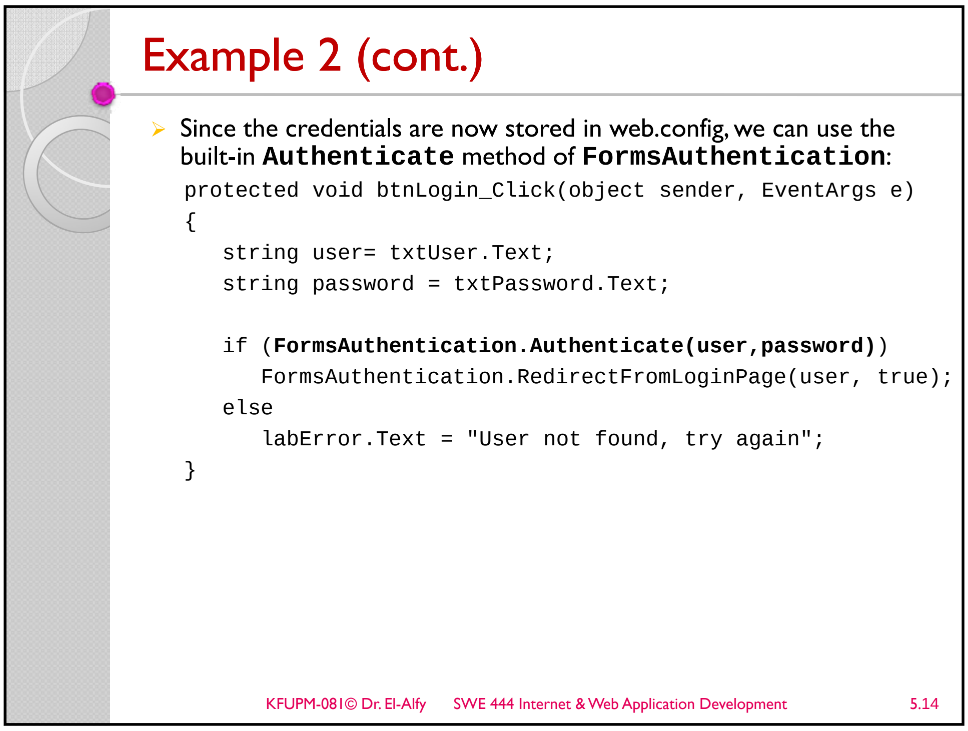
```
protected void btnLogin_Click(object sender, EventArgs e)
{
   string user= txtUser.Text;
   string password = txtPassword.Text;

   if (IsValidUser(user, password))
      FormsAuthentication.RedirectFromLoginPage(user, true);
   else
      labError.Text = "User not found, try again";
}

private bool IsValidUser(string user, string password)
   {
      if (user == "sahl" && password == "abushabab")
         return true;
      else
         return false;
   }
```



Example 2

- Storing user credentials in a web.config file:

```
<configuration>
…
<system.web>
    <authentication mode="Forms">
       <forms loginUrl="Login.aspx" >
         <credentials passwordFormat="Clear">
            <user name="sahl" password="abushabab"/>
            <user name="ahmad" password="abuatfal"/>
            <user name="ali" password="abulkhair"/>
         </credentials>
       </forms>
    </authentication>
    <authorization>
           <deny users="?"/>
    </authorization>
   </system.web>
</configuration>
```



Example 2 (cont.)

- Since the credentials are now stored in web.config, we can use the built-in **`Authenticate`** method of **`FormsAuthentication`**:

```
protected void btnLogin_Click(object sender, EventArgs e)
{
   string user= txtUser.Text;
   string password = txtPassword.Text;

   if (FormsAuthentication.Authenticate(user,password))
      FormsAuthentication.RedirectFromLoginPage(user, true);
   else
      labError.Text = "User not found, try again";
}
```

Example 3: Customizing Authentication

- Suppose we want to allow everyone access to the main folder of the application and allow access to a MembersOnly folder only to authenticated users
- We place the following in the main folder

```
<configuration>
…
<system.web>
      <authentication mode="Forms">
         <forms loginUrl="Login.aspx" > <!– can add credentials here ! -->
         </forms>
      </authentication>
      <authorization>
          <allow users="*"/>
      </authorization>
   </system.web>
</configuration>
```

- And place the following in the `MembersOnly` folder (there should not be `authentication` element here!):

```
<configuration >
   <system.web>
      <authorization>
           <deny users="?"/>
      </authorization>
    </system.web>
</configuration>
```

Example 4: Authentication using WAT

- The most versatile solution is to store user credentials in a database
- This can be done by creating authentication information using the WAT (Website Administration Tool) in Visual Studio 2005
  - Start the WAT:
    - Web Site > ASP.NET Configuration
  - Click the Security Table
  - Click the Create User link
  - Fill in the form and click the Create User button
  - Add two more users
- From the above steps, the WAT would have created an SQL server database with the information you entered added to a number of tables
  - Or better still see Chapter 13 of Randy Connolly's "Core Internet Application Development with ASP.NET 2.0", 2007

Stored Procedures

- A precompiled collection of SQL statements stored under a name and processed as a unit.
- They're stored in and deployed with the database
- They are usually written in a proprietary database language like PL/SQL for Oracle database or PL/PgSQL for PostgreSQL.
- Stored procedures are extremely similar to the constructs seen in other programming languages.
  - They accept data in the form of input parameters that are specified at execution time.
  - These input parameters (if implemented) are utilized in the execution of a series of statements that produce some result.
  - This result is returned to the calling environment through the use of a recordset, output parameters and a return code.
- Types
  - User-defined Stored Procedures
  - System Stored Procedures

Benefits of Stored Procedures

1. Precompiled execution.
   - SQL Server compiles each stored procedure once and then reutilizes the execution plan.
   - This results in tremendous performance boosts when stored procedures are called repeatedly.
2. Reduced client/server traffic.
   - Stored procedures can reduce long SQL queries to a single line thereby reducing network traffic.
3. Efficient reuse of code and programming abstraction.
   - Stored procedures can be used by multiple users and client programs.
   - Judicious use of stored procedures can reduce development time.
4. Enhanced security controls.
   - You can grant users permission to execute a stored procedure independently of underlying table permissions.

Stored Procedures: Example

- Consider the following `studentGrades` table:

ID	Name	Standing	Grades(%)
40232	Ahmad	P	50
40165	Khalid	G	50
40147	Qais	P	50
40244	Ibrahim	P	99
40284	Ali	G	84
40434	Amr	G	32

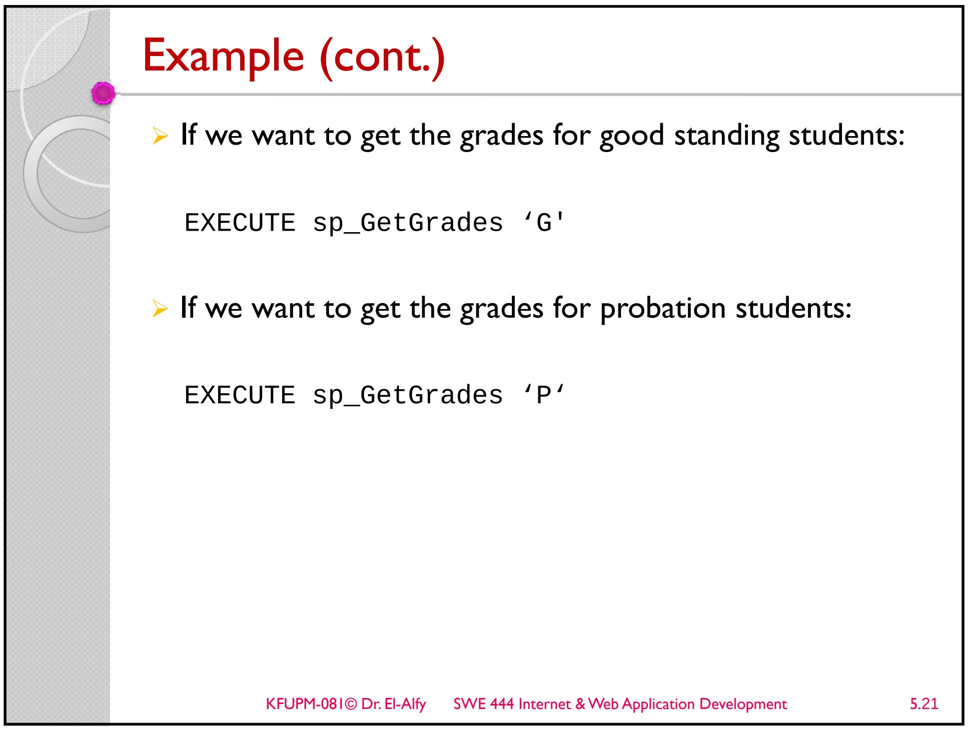
Example (cont.)

- In Query:

```
SELECT Name, Grades
FROM studentGrades
WHERE Standing = ‘G’
```

- In Stored Procedure (Visual Basic):

```
CREATE PROCEDURE sp_GetGrades
      @standing varchar(1)
      AS
      SELECT Name, Grades
      FROM studentGrades
      WHERE Standing = @standing
```



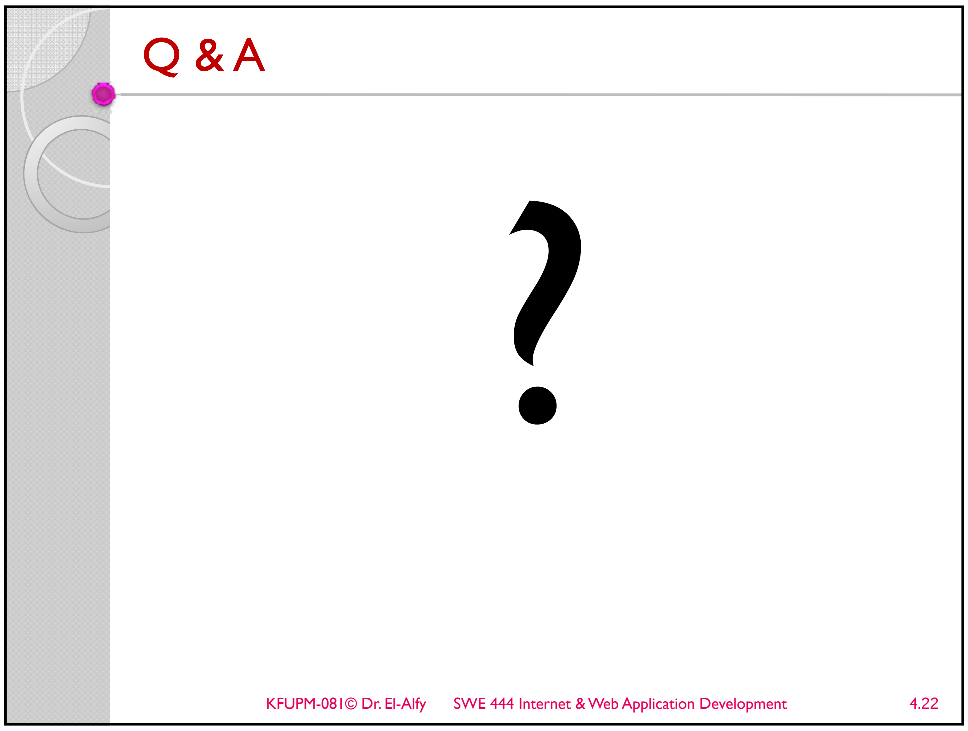
Example (cont.)

- If we want to get the grades for good standing students:

```
EXECUTE sp_GetGrades ‘G'
```

- If we want to get the grades for probation students:

```
EXECUTE sp_GetGrades ‘P‘
```



Q & A

?

References

- H. M. Deitel, P. J. Deitel, and A. B. Goldberg, *Internet and World Wide Web How to Program*, 4/e, Pearson Education Inc., 2008.
- Some useful links with examples and other resources:
  - W3C http://www.w3.org/TR/xpath
  - W3School ADO Tutorial
    - http://www.w3schools.com/asp/default.asp
  - W3School ADO Tutorial http://www.w3schools.com/ado/default.asp
  - W3School SQL Tutorial
    - http://www.w3schools.com/sql/default.asp
  - W3School PHP Tutorial
    - http://www.w3schools.com/php/default.asp