Using SQL Server and ASP.Net on Sandstorm

This exercise introduces you to the Microsoft's SQL Server, ADO.NET and ASP.NET web services.
For this exercise we will use the SQL Server `on sandstorm.cs.ut.ee`

Microsoft SQL Server Management Studio

This exercise can be completed by opening SQL Server Management Studio on `sandstorm.cs.ut.ee`.

- Start SQL Server Management Studio and connect to:

```
Server type: Database Engine
Server name: sandstorm.cs.ut.ee
Authentication: SQL Server Authentication
Login: sg1
Password: SG1_pass
```

If you wish, you can change your password as follows:
- Select "New Query" from the toolbar and enter:

```
sp_password @old='SG1_pass', @new='newpassword'
```

where *newpassword* is your new password.

A database has already been created on the server for your use. The name of this database is ST1. When you connect to SQL Server using the above procedure, this database will automatically be opened when you connect, so you can immediately start creating tables on this database. Creating a table can be done as follows:
- Right click on the Tables tab of your database and select "`New Table…`".
- Create a `Customer` table as follows and populate it with some sample data.

```
CREATE TABLE Customer (
        Name VARCHAR(80) NOT NULL,
        Street VARCHAR(100) NOT NULL,
        Postcode INTEGER NOT NULL,
        PRIMARY KEY (Name))
        ;
```

Next we will write a simple .NET application that accesses this database using the ADO.NET database API.
- Start Visual Studio.NET 2005 and create a new C# Console Application Project.
  (You can use VB.NET if you prefer, but you will need to adjust the syntax).

- In the main method, add the following code:

First we need to create and open an ADO.NET connection to the `sandstorm` server:

```
SqlConnection connection = new SqlConnection(
"Data Source=sandstorm.cs.ut.ee;Initial Catalog=ST1;User
ID=sg1;Password=SG1_pass");

connection.Open();
```

You can either fully qualify these class names or simply `import the namespace:`

```
using System.Data.SqlClient;
```

Next we use ADO.NET to create an SQL command:

```
SqlCommand command = new SqlCommand("SELECT Name, Street, Postcode FROM
Customer", connection);
```

There are many things you can do with an SQL command, but in this case we will use it to obtain an ADO.NET data reader:

```
SqlDataReader reader = command.ExecuteReader();
```

We can iterate through the rows of the result set by calling the data reader's `next` method until it returns false.

```
while (reader.Read())
     ...
```

For each row we will print out the 3 columns:

```
    System.Console.WriteLine(reader.GetString(0));
    System.Console.WriteLine(reader.GetString(1));
    System.Console.WriteLine(reader.GetInt32(2));
```

Finally we need to close the connection and data reader:

```
    reader.Close();
    connection.Close();
```

- Now try to compile and run your application.

ASP.NET Web Services

Next we'll convert our simple .NET application into a Web Service implemented using ASP.NET.

- In Visual Studio 2005, create a new "Web Site…" (Empty Web Site) in File System Location C:\Temp\Tutorial8
- Right click on Tutorial8 and Add New Item... a Web Service called CustomerService.asmx
- In the CustomerService class, replace the HelloWorld web method with:

```
public Customer[] GetCustomers()
```

- Create a new class called Customer that contains public fields for Name, Street and Postcode of type `string, string` and `int` respectively. Add a default constructor as well as a constructor that takes three initialization parameters.

- Copy the ADO.NET code from your console application to your new GetCustomers method and modify it so that it creates and returns an array of Customers rather than writing them to the console. Hint: use a `System.Collections.Generic.List<Customer>` object to accumulate the list of Customers as you read them and convert it to an array at the end using its `ToArray` method.

- Build and then Start (without debugging) your application.

- Click on "Service Description" to view the WSDL.

- Click on "`GetCustomers`" to test your web service.

Deploying ASP.NET Web Services

Next we'll deploy our ASP.NET web service to the `sandstorm` server.

- In the Address bar of Windows Explorer type `\\Inetpub\`***`sg1`***

- Copy your local Tutorial8 application directory into the above server directory.

- Use your browser (preferably use IE as a browser for this exercise because IE allows you to test your Web services more easily) to access the `asmx` file that you want to test on the sandstorm server. The application should be accessible here: http://sandstorm.cs.ut.ee:2345/CustomerService.asmx

Creating a Web Service Client to a Remote Web Service

- Use Visual Studio to create a console application (C# or VB.NET)
  - Add a web reference to http://terraservice.net/TerraService2.asmx?WSDL
  - In your main method, create an instance of the `TerraServer` proxy class.
  - Create a `Place` object called `NY` and set its properties to "New York", "New York", "United States".
  - Invoke the `GetPlaceFacts` method of the service to retrieve a `PlaceFacts` object about New York.
  - Write out the type of place that it is and the longitude and latitude its centre.
  - Compile and test your application.

- Invoke the `GetTileMetaFromLonLatPt` method of the service, passing the centre of New York, a theme of `1` and a scale of `8` metres.
  - If the tile metadata indicates that the tile `exists`, invoke the `GetTile` method of the service to retrieve the `tileId` contained in the metadata.
  - Convert the byte array into a `System.Drawing.Image` using:
    `Image.FromStream(new System.IO.MemoryStream(tile))`
  - Invoke the image's `save` method to write the image to a file.
  - Compile and test your application.

- Try out some other scales (1m – 512m), themes (1 or 2) and cities (North America only).

Creating a Web Service (using WSDL first approach)

- Open a Visual Studio command prompt
  - Use the `wsdl` tool to generate a server interface for the WSDL file located at ITempConverter.wsdl

- Use Visual Studio to create a new Web Site containing an ASP.NET Web Service called `TemperatureConverter`.
  - Add the `ITempConverterserviceInterfaces.cs` file to the `App_Code` folder of the Visual Studio project
  - Modify the implementation of the Service:
    - Remove the constructor and `HelloWorld` method
    - Remove the attributes from the class.
    - Change the base class to `IITempConverterbinding`
    - Create `CtoF` and `FtoC` methods as defined in the interface.
  - Implement the `CtoF` and `FtoC` methods:
    - F = C * 9/5 + 32
    - C = (F – 32) * 5/9
  - Compile and "test" your web service.
    - Do not create a client application – just test locally via web browser

You can now deploy the Web service on the Sandstorm server by dropping the entire directory containing your Web site into `\\Inetpub\`***`sg1`*** `on` Sandstorm, and then check it by pointing your browser to: http://sandstorm.cs.ut.ee:2345/TemperatureConverter.asmx