

Guía Instalación de la Prueba Programada – Súper Zapatos GAP


Descarga Fuentes


Tanto la aplicación web como el API de servicios REST se encuentran en un repositorio en GIT, por lo que lo pueden descargar de la siguiente dirección:


https://github.com/BernalF/GAP_Test/archive/master.zip


Creación de la Base de Datos y aplicación de Scripts


Dentro los fuentes descargos, hay una carpeta que se llama Databases; aplicar los scripts siguiendo el orden numérico del nombre de los archivos.


 01 Database Objects - SShoesDB.sql


 02 uspAddArticles.sql

 03 uspGetArticles.sql

 04 uspDelArticles.sql

 05 uspAddOrModifyStores.sql

 06 uspGetStores.sql

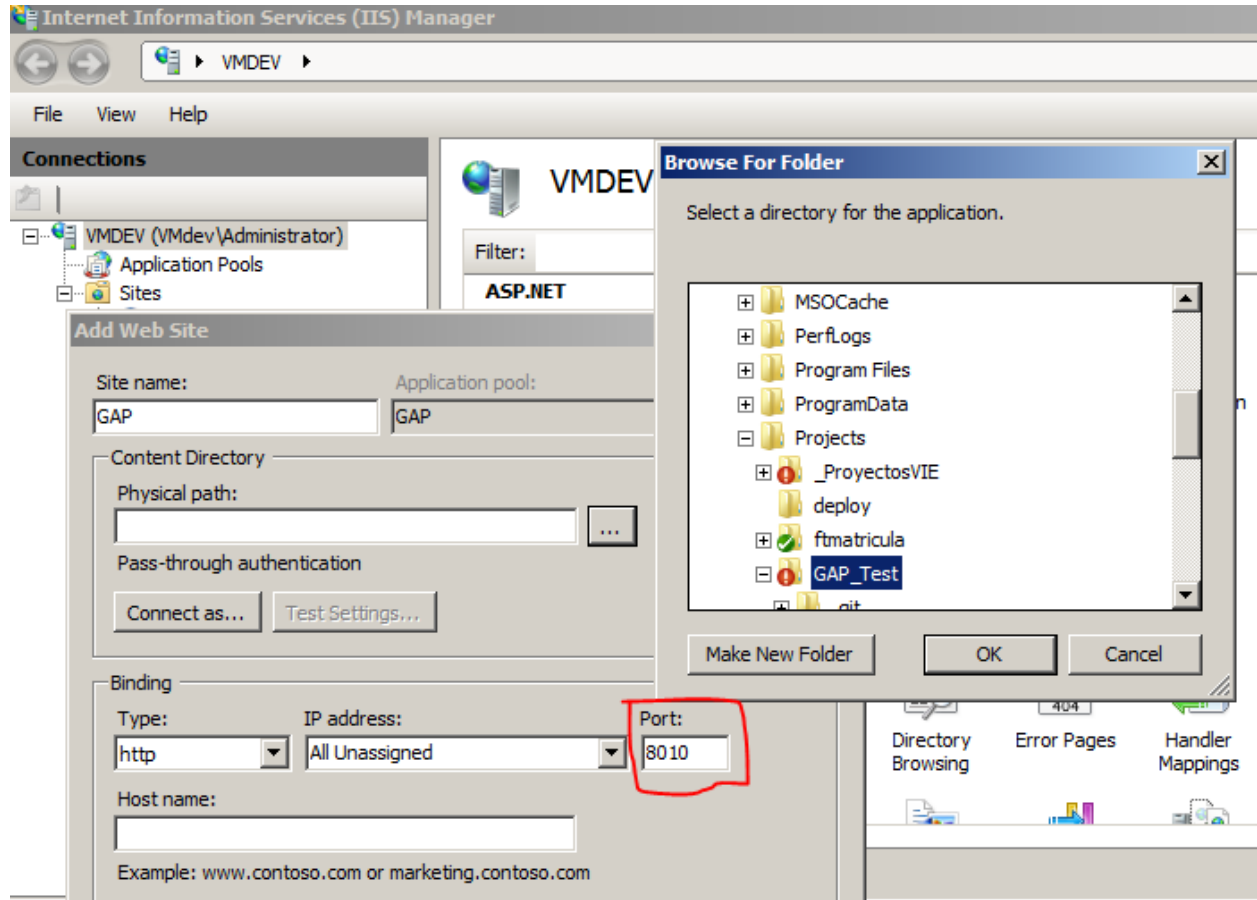
 07 uspDelStores.sql

 08 uspModifyArticles.sql

Como Nota importante es necesario crear la base de datos manual con el nombre SShoesDB

Configuraciones

Al tratarse de dos sitios web, todo se creó para ser manejado en un sitio web aparte, por lo que primeramente se debe abrir el IIS y agregar un web site nuevo que apunte al puerto 8010



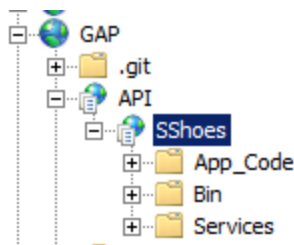
Antes de proseguir verificar si existe algún Application pool como el indicado en la siguiente imagen. El nombre del Application pool no es Importante, pero si debe estar configurado con .NET framework 4.0 o superior y Managed Pipeline Mode: Integrated.



En caso de no existir ninguno con esas características se debe crear uno

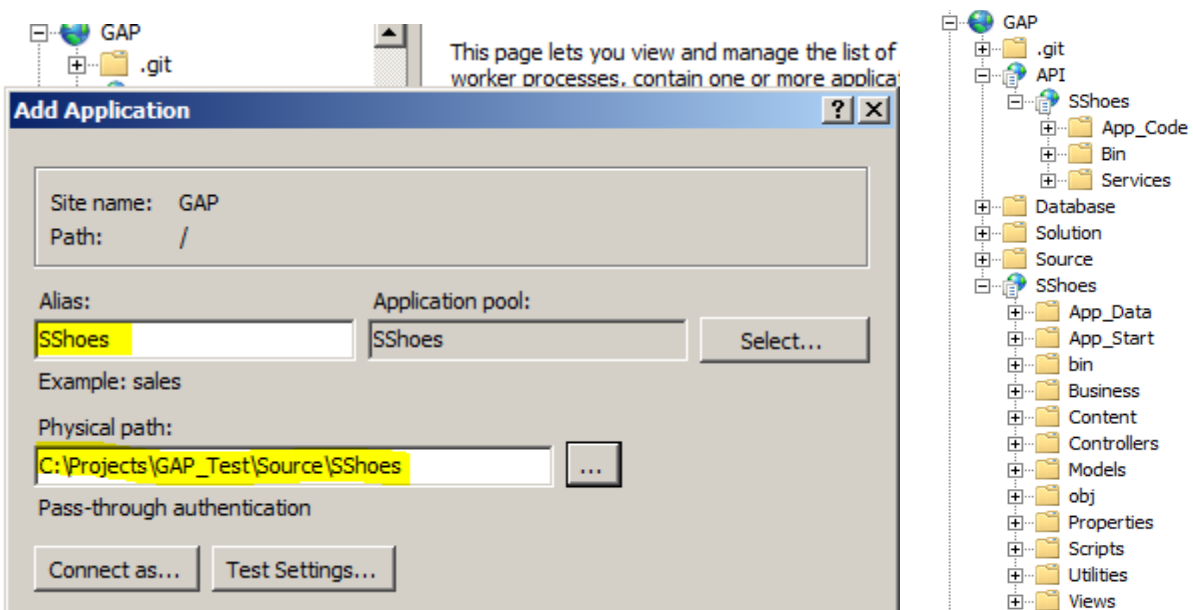
Configuración Servicios REST

Dentro del nuevo sitio web creado, convertir tanto la carpeta API como la de SShoes como aplicaciones web



Configuración del sitio “Super Shoes “

Sobre el nuevo site creado, le damos clic derecho -> Add Application, le ponemos SShoes y lo ponemos a apuntar a la carpeta SShoes dentro de Source de los fuentes descargados.



Configuraciones en el Web.config de la aplicación SShoes

- Definir la base de datos: Buscar la sección “connectionStrings” y modificar los datos correspondientes:

Data Source=IP_SERVER_DE_BASE_DATOS;

InitialCatalog=NOMBRE_DE_BASE_DATOS;

User ID=USUARIO_BD;

Password=PASSWORD_USUARIO_BD

```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data Source=localhost;Initial Catalog=SShoesDB;
  User ID=sa;Password=sax" providerName="System.Data.SqlClient" />
</connectionStrings>
```

- Definir la dirección ip a la cual se hospeda el API de servicios: Buscar el key dentro la sección de Application keys y modificar el valor de dicho key

```
<add key="apiServer" value="http://localhost:8010/" />
</appSettings>
```

La aplicación esta correctamente configurada, pero antes de proseguir un par de notas:

- Hay dos tipos de proyectos en la carpeta sources; el cual es el proyecto publicado, es decir, solo vistas y dll, por lo que si cuando creamos la aplicación lo ponemos a apuntar a esa carpeta; en teoría el proyecto le cargaría a la primera
- Si de lo contrario ponen a apuntar SShoes a como lo mostraban las imágenes anteriores, es necesario abrir la solución y compilar el proyecto para que se generen los dlls.

Tomando en cuenta esas notas podemos ingresar a la URL <http://localhost:8010/SShoes/>

The screenshot shows the 'Super Shoes' application interface. At the top, there's a navigation bar with 'Super Shoes', 'Stores', and 'Articles'. On the left, there's a sidebar with 'INVENTORY CONTROL' and sub-items 'Stores' and 'Articles'. The main content area is titled 'Articles List' and features a table with columns: Name, Description, Price, Total in self, Total in vault, and Store Name. There are two rows of data: 'Articulo 1' and 'Aires I'. Each row has an 'Edit' button. An 'Add Articles' button is located at the top right of the table.

Name	Description	Price	Total in self	Total in vault	Store Name	
Articulo 1	Articulo 1	\$130.00	10	10	Tienda 1	Edit
Aires I	Aires I	\$300.00	3	4	Tienda 2	Edit

© 2013 - by BernalF

Retrospectiva

Algunas cosas importantes para observar a nivel de desarrollo:

- La aplicación se desarrollo en MVC 4 con Razor, jQuery, C# y SQL Server 2008
- Todo el código tiene documentación interna de manera light.
- Casi responsive, se podría amoldar en un browser de un teléfono con android.
- Se hicieron las cosas de varias maneras para así mostrar que las cosas se pueden hacer de distintas formas, como por ejemplo, a nivel de base de datos, se maneja en unos caso un solo store procedure para agregar y modificar (stores), aunque luego se creó uno para cada uno (articles), también sucede con el retrieve de data que se creó el retrieve de todos los elementos y de un elemento específico por ID en el mismo store procedure.
- Para el consumo de los servicios REST se consumieron con jquery por medio de ajax y desde c# con HttpWebRequest, para mostrar los dos escenarios y las dos formas de hacerlos.
- El mantenimiento de scores se hizo de otra manera, explotando a lo máximo el modelo de MVC, para eso está la clase dentro de la carpeta Business que llega a ser nuestra capa lógica.
- La clase GeneralCollections se utiliza para devolver un Enumerable para hacer binding de los DropDowns.
- Los 3 servicios REST creados son los de Add, Update y Delete Articles, de los cuales los dos primeros van por POST y con jQuery y el ultimo por GET y desde el controller (esto también para ejemplificar las dos formas de consumirlos)
- Los servicios REST manejan su propio namespace, removiendo el famoso y default "tempuri"
- La capa de datos se maneja de manera implícita con un dll de mi propiedad creado por un colega y mi persona llamado BBCorporation.DataAccess.
- Para la interfaz grafica, se utilizo twitter bootstrap.
- El javascript se manejo como un modulo en un archivo js aparte.

En general el ejercicio estuvo fácil e interesante cuando intente consumir un servicio REST desde el controller (c#), digamos que tiene su toque, por lo demás fue un bonito ejercicio.