14. Agregar código para inicializar la base de datos con datos de prueba

Entity Framework creará una base de datos vacía para usted. En esta sección, se escribe un método que es ejecutado después de que se crea la base de datos, el método llena con datos de prueba la base de datos.

Aquí se usará el método `EnsureCreated` para crear automáticamente la base de datos. En los videos siguientes, se revisará cómo manejar los cambios de modelo mediante el uso de las primeras migraciones de código para cambiar el esquema de la base de datos en lugar de eliminar y volver a crear la base de datos.

En la carpeta Data, cree un archivo de clase nuevo denominado *DbInitializer.cs* y reemplace el código de plantilla por el código siguiente, lo que hace que una base de datos se cree cuando sea necesario y carga los datos de prueba en la nueva base de datos.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Sistema.Models;

namespace Sistema.Data
{
    public class DbInitializer
    {
        public static void Initialize(SistemaContexto context)
        {
            context.Database.EnsureCreated();
            // Busca si existe categorias en la BD.
            if (context.Categorias.Any())
            {
                return;   // Si existe registros la BD Ha sido creada
            }

            var categorias = new Categoria[]
            {
            new Categoria{Nombre="Programación",Descripcion="Cursos de programación y
base de datos"}
            };
            foreach (Categoria c in categorias)
            {
                context.Categorias.Add(c);
            }
            context.SaveChanges();
        }
    }
}
```

El código comprueba si hay categorias en la base de datos, y si no, se supone que la base de datos es nueva y necesita ser creada con datos de prueba.

En Startup.cs, modifique el método `Configure` para llamar a este método al inicio de la aplicación. Primero, agregue el contexto a la firma del método para que la inyección de dependencia de ASP.NET pueda proporcionarla a su clase `DbInitializer`.

A continuación, llame a su método `DbInitializer`. Agrengando el contexto como parametro al final del método `configure`.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory,SistemaContexto context)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.UseStaticFiles();

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Home}/{action=Index}/{id?}");
            });
              DbInitializer.Initialize(context);
        }
```

Ahora, la primera vez que ejecute la aplicación, la base de datos se creará y se insertarán datos de prueba. Cada vez que cambie su modelo de datos, puede eliminar la base de datos, actualizar su método y empezar de nuevo con una nueva base de datos de la misma manera. En los videos siguientes verá cómo modificar la base de datos cuando cambia el modelo de datos, sin eliminarlo ni volver a crearlo utilizando migraciones.