This topic covers how Entity Framework discovers which database connection to use, and how you can change it. Models created with Code First and the EF Designer are both covered in this topic.

Typically an Entity Framework application uses a class derived from DbContext. This derived class will call one of the constructors on the base DbContext class to control:

- How the context will connect to a database—i.e. how a connection string is found/used
- Whether the context will use calculate a model using Code First or load a model created with the EF Designer
- Additional advanced options

The following fragments show some of the ways the DbContext constructors can be used.

Use Code First with connection by convention

If you have not done any other configuration in your application, then calling the parameterless constructor on DbContext will cause DbContext to run in Code First mode with a database connection created by convention. For example:

```
namespace Demo.EF
{
    public class BloggingContext : DbContext
    {
        public BloggingContext()
        // C# will call base class parameterless constructor by default
        {
        }
    }
}
```

In this example DbContext uses the namespace qualified name of your derived context class—Demo.EF.BloggingContext—as the database name and creates a connection string for this database using either SQL Express or LocalDb. If both are installed, SQL Express will be used.

Visual Studio 2010 includes SQL Express by default and Visual Studio 2012 includes LocalDb. During installation, the EntityFramework NuGet package checks which database server is available. The NuGet package will then update the configuration file by setting the default database server that Code First uses when creating a connection by convention. If SQL Express is running, it will be used. If SQL Express is not available then LocalDb will be registered as the default instead. No changes are made to the configuration file if it already contains a setting for the default connection factory.

Use Code First with connection by convention and specified database name

If you have not done any other configuration in your application, then calling the string constructor on DbContext with the database name you want to use will cause DbContext to run in Code First mode with a database connection created by convention to the database of that name. For example:

```
public class BloggingContext : DbContext
{
    public BloggingContext()
        : base("BloggingDatabase")
    {
    }
}
```

In this example DbContext uses "BloggingDatabase" as the database name and creates a connection string for this database using either SQL Express (installed with Visual Studio 2010) or LocalDb (installed with Visual Studio 2012). If both are installed, SQL Express will be used.

Use Code First with connection string in app.config/web.config file

You may choose to put a connection string in your app.config or web.config file. For example:

```
<configuration>
  <connectionStrings>
```

```
    <add name="BloggingCompactDatabase"
         providerName="System.Data.SqlServerCe.4.0"
         connectionString="Data Source=Blogging.sdf"/>
  </connectionStrings>
</configuration>
```

This is an easy way to tell DbContext to use a database server other than SQL Express or LocalDb — the example above specifies a SQL Server Compact Edition database.

If the name of the connection string matches the name of your context (either with or without namespace qualification) then it will be found by DbContext when the parameterless constructor is used. If the connection string name is different from the name of your context then you can tell DbContext to use this connection in Code First mode by passing the connection string name to the DbContext constructor. For example:

```
public class BloggingContext : DbContext
{
    public BloggingContext()
        : base("BloggingCompactDatabase")
    {
    }
}
```

Alternatively, you can use the form "name=<connection string name>" for the string passed to the DbContext constructor. For example:

```
public class BloggingContext : DbContext
{
    public BloggingContext()
        : base("name=BloggingCompactDatabase")
    {
    }
}
```

This form makes it explicit that you expect the connection string to be found in your config file. An exception will be thrown if a connection string with the given name is not found.

Database/Model First with connection string in app.config/web.config file

Models created with the EF Designer are different from Code First in that your model already exists and is not generated from code when the application runs. The model typically exists as an EDMX file in your project.

The designer will add an EF connection string to your app.config or web.config file. This connection string is special in that it contains information about how to find the information in your EDMX file. For example:

```
<configuration>
  <connectionStrings>
    <add name="Northwind_Entities"
         connectionString="metadata=res://*/Northwind.csdl|
                                    res://*/Northwind.ssdl|
                                    res://*/Northwind.msl;
                           provider=System.Data.SqlClient;
                           provider connection string=
                               &quot;Data Source=.\sqlexpress;
                                     Initial Catalog=Northwind;
                                     Integrated Security=True;
                                     MultipleActiveResultSets=True&quot;"
         providerName="System.Data.EntityClient"/>
  </connectionStrings>
</configuration>
```

The EF Designer will also generate code that tells DbContext to use this connection by passing the connection string name to the DbContext constructor. For example:

```
public class NorthwindContext : DbContext
{
    public NorthwindContext()
        : base("name=Northwind_Entities")
    {
    }
}
```