How to work with connections, commands and data readers

1. Execute the script that creates a MvcMusicStore database. You can find this script on blackboard.
2. Create a new solution called **MusicStore**, with a WPF project called **MusicStore**. The WPF project is the presentation layer and contains the GUI.
3. Add a project called **MusicStore.Data** to the solution. This project will be our data layer. In this project:
    - Create (domain) classes for the following tables: Album, Artist and Genre
    - Create a *ConnectionFactory* class which contains one static method *CreateSqlConnection*, which returns a connection object to the MusicStore database
    - Create for each domain class a database access class (*AlbumRepository, ArtistRepository, GenreRepository*)
        1. *GenreRepository* contains a static *GetGenres* method which receives no parameters. (Genres have to be shown in the ComboBox)
        2. *AlbumRepository* contains a static *GetAlbumsByGenre* method which receives a *genreId* parameter (Only the albums of the selected Genre have to be listed in the datagrid)
        3. *ArtistRepository* contains a static *GetArtistNameById* method which receives a *artistId* property (Instead of the *ArtistId*, the *artistName* has to be shown in the datagrid)
4. Add a project called **MusicStore.Business** to the solution. This project will be our business layer. In this project:
    - Create a class called *AlbumSummary* with string properties (Title, Artist, Price).
    - Create a class called *AlbumSummaryService* with a *GetAlbumSummariesByGenre* method. This class is responsible for retrieving the albums of a genre from the repository and then transform the list of albums to a list of album summaries.
      You'll need the *ArtistRepository* to get the name of the artist.
      Make sure the price is formatted as a currency.
5. In the GUI project:
    - Use a ComboBox to display a list of the genres. Use the *DisplayMemberPath* and *SelectedValuePath* properties to bind each item to an instance of *Genre*.
    - Use a DataGrid control to display the album summaries.

```
<DataGrid x:Name="albumDataGrid" AutoGenerateColumns="False"
          HorizontalAlignment="Left" Margin="70,130,20,20"
          VerticalAlignment="Top" MinHeight="200">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Title" Width="150" Binding="{Binding Path=Title}"></DataGridTextColumn>
        <DataGridTextColumn Header="Artist" Width="200" Binding="{Binding Path=Artist}"></DataGridTextColumn>
        <DataGridTextColumn Header="Price" Width="100" Binding="{Binding Path=Price}"></DataGridTextColumn>
    </DataGrid.Columns>
</DataGrid>
```

Eventually the application should look like this: