<https://lms.thesoftwareguild.com/courses/368/assignments/4314?module_item_id=58698>

**Assignment: DVD Library Database & Web API**

* Due No Due Date
* Points 100
* Submitting a website url

**Overview**

The purpose of this assessment is to show your proficiency in developing database objects and interacting with them using the C# database access techniques of ADO .NET and Entity Framework.  You will expose your database interactions through an ASP .NET Web API project.

**Requirements**

Your assignment is to create some database build scripts, an ASP .NET Web API project, and utilize the previous assessment's DVD Library REST client HTML/JavaScript/CSS to connect to it. Your Web API must provide all the functionality necessary to successfully run the application.

**Database Build Scripts**

* Your Web API must provide the same functionality as the provided JAR application from the DVD Library Rest Client Lab.
* You must provide repeatable scripts to create thedatabase,it’sobjects, and sample data. Please submit a .zip file named DvdScripts.zip containing these files:
  + DvdLibraryCreate.sql – Should create the database and its tables
  + DvdLibrarySprocs.sql – Should create the stored procedures.
  + DvdLibrarySampleData.sql – Should create sample data
  + DvdLibrarySecurity.sql – Should create an application account.

**Web Application Guidance**

You should not make any changes to your previous HTML/CSS/JavaScript application except to update the URL to point to your Web API application. Additionally:

* + The Web API application must be configured to use CORS. Allow all sites, origins, and methods (\*).
  + The Web API application should contain an IDvdRepository interface. There should be three implementations of this interface:
    - DvdRepositoryMock – Returns in-memory sample data.
    - DvdRepositoryEF – Uses Entity Framework code-first to store and retrieve data from SQL Server.
    - DvdRepositoryADO – Uses ADO .NET to store and retrieve data from SQL Server.
  + A factory class should instantiate the appropriate implementation of the interface. It should be driven by a web.config <appSettings> key called “Mode”.  Values of “Mode” are:
    - SampleData – Selects DvdRepositoryMock
    - EntityFramework – Selects DvdRepositoryEF
    - ADO – Selects DvdRepository ADO
  + The DvdLibrarySecurity.sql file must perform the following actions.
    - Create a server login named ‘DvdLibraryApp’ with a password of ‘testing123’.
    - Create a database account for ‘DvdLibraryApp’.
    - Grant Execute on all used stored procedures to ‘DvdLibraryApp’
    - Grant SELECT, INSERT, UPDATE, and DELETE on all used tables to ‘DvdLibraryApp’

Hint – For search functionality you should use the LIKE SQL keyword in the WHERE clause for the stored procedure and the Contains() string method in Entity Framework.

**DVD Web API Specification**

You must provide the following functionality:

| **Retrieving a Dvd by Id** | |
| --- | --- |
| HTTP Method: | GET |
| URL: | /dvd/{id} |
| Path Variable: | {id} – Id of the requested Dvd |
| Request Body: | None |
| Response Body: | JSON representation of a Dvd:  {   “dvdId”: 0,   “title”: “A Great Tale”,   “realeaseYear”: 2015,   “director”: “Sam Jones”,   “rating”: “PG”,   “notes”: “This really is a great tale!” } |

| **Retrieving all Dvds** | |
| --- | --- |
| HTTP Method: | GET |
| URL: | /dvds |
| Path Variable: | None |
| Request Body: | None |
| Response Body: | JSON array of Dvds:  [   {     "dvdId": 0,     "title": "A Great Tale",     "realeaseYear": 2015,     "director": "Sam Jones",     "rating": "PG",     "notes": "This really is a great tale!"   },   {     "dvdId": 1,     "title": "A Good Tale",     "realeaseYear": 2012,     "director": "Joe Smith",     "rating": "PG-13",     "notes": "This is a good tale!"   } ] |

| **Retrieving Dvds by Title** | |
| --- | --- |
| HTTP Method: | GET |
| URL: | /dvds/title/{title} |
| Path Variable: | {title} – Title search term |
| Request Body: | None |
| Response Body: | JSON array of Dvds that match the search criteria:  [   {     "dvdId": 0,     "title": "A Super Tale",     "realeaseYear": 2015,     "director": "Sam Jones",     "rating": "PG",     "notes": "A great remake!"   },   {     "dvdId": 1,     "title": "A Super Tale",     "realeaseYear": 1985,     "director": "Joe Smith",     "rating": "PG",     "notes": "The original!"   } ] |

| **Retrieving Dvds by Release Year** | |
| --- | --- |
| HTTP Method: | GET |
| URL: | /dvds/year/{releaseYear} |
| Path Variable: | {releaseYear} – Release year search term |
| Request Body: | None |
| Response Body: | JSON array of Dvds that match the search criteria:  [   {     "dvdId": 0,     "title": "A Great Tale",     "realeaseYear": 2015,     "director": "Sam Jones",     "rating": "PG",     "notes": "This really is a great tale!"   },   {     "dvdId": 1,     "title": "A Wonderful Tale",     "realeaseYear": 2015,     "director": "Joe Smith",     "rating": "PG-13",     "notes": "Wonderful, just wonderful!"   } ] |

| **Retrieving Dvds by Director Name** | |
| --- | --- |
| HTTP Method: | GET |
| URL: | /dvds/director/{directorName} |
| Path Variable: | {directorName} – Director name search term |
| Request Body: | None |
| Response Body: | JSON array of Dvds that match the search criteria:  [   {     "dvdId": 0,     "title": "A Great Tale",     "realeaseYear": 2015,     "director": "Sam Jones",     "rating": "PG",     "notes": "This really is a great tale!"   },   {     "dvdId": 1,     "title": "A Wonderful Tale",     "realeaseYear": 2015,     "director": "Joe Smith",     "rating": "PG-13",     "notes": "Wonderful, just wonderful!"   } ] |

| **Retrieving Dvds by Rating** | |
| --- | --- |
| HTTP Method: | GET |
| URL: | /dvds/rating/{rating} |
| Path Variable: | {rating} – Rating search term |
| Request Body: | None |
| Response Body: | JSON array of Dvds that match the search criteria:  [   {     "dvdId": 0,     "title": "A Great Tale",     "realeaseYear": 2015,     "director": "Sam Jones",     "rating": "PG",     "notes": "This really is a great tale!"   },   {     "dvdId": 1,     "title": "Just A Tale",     "realeaseYear": 2015,     "director": "Joe Baker",     "rating": "PG",     "notes": "It is a tale!"   } ] |

| **Creating a new Dvd** | |
| --- | --- |
| HTTP Method: | POST |
| URL: | /dvd |
| Path Variable: | None |
| Request Body: | JSON representation of Dvd to be created (NOTE – this **does not** **include a dvdId)**:  {     "title": "A New Tale",     "realeaseYear": 2016,     "director": "Jack Jameson",     "rating": "PG-13",     "notes": "Brand spanking new!" } |
| Response Body: | JSON representation of the newly created Dvd (NOTE – this **does include the system assigned dvdId**):  {     "dvdId": 42,     "title": "A New Tale",     "realeaseYear": 2016,     "director": "Jack Jameson",     "rating": "PG-13",     "notes": "Brand spanking new!" } |

| **Updating an Existing Dvd** | |
| --- | --- |
| HTTP Method: | PUT |
| URL: | /dvd/{id} |
| Path Variable: | {id} – Id of Dvd to be updated |
| Request Body: | JSON representation of Dvd to be updated (NOTE – this **does** **include a dvdId)**:  {     "dvdId": 42,     "title": "A New Tale",     "realeaseYear": 2016,     "director": "Jack Jameson",     "rating": "PG-13",     "notes": "I’m just changing my note..." } |
| Response Body: | None |

| **Deleting a Dvd** | |
| --- | --- |
| HTTP Method: | DELETE |
| URL: | /dvd/{id} |
| Path Variable: | {id} – Id of Dvd to be deleted |
| Request Body: | None |
| Response Body: | None |