

## **Creation of REST API using C# Web API in MVC Architecture which is useful for Library Applications**

This process is basically the implementation of REST API. The Steps that are followed in creation of this project is as follows.

1. Creation of a Web API project in MVC Architecture.
2. Connecting the BookInfo table and AuthorInfo table from the SQL database using entity framework.
3. Basic methods like GET, POST, PUT, DELETE are implemented in this project and verified by using Postman. End points are created for both Author and Book Information.
4. Basic Authentication is performed for an endpoint to GET the books list.
5. Verification of the results for the given endpoints

### **1. Creation of a Web API project in MVC Architecture.**

A new project is created using MVC Architecture in Entity framework in Visual Studio.

### **2. Connecting the BookInfo table and AuthorInfo table from the SQL database using entityframework.**

Tables named BookInfo and AuthorInfo are created in SQL database using SQL Server Management Studio. They consists of all the information about the list of Books and Authors with their respective IDs and details.

### **3. Basic methods like GET, POST, PUT, DELETE are implemented in this project and verified by using Postman. End points are created for both Author and Book Information.**

Methods to get the book information and Author Information from the database as a list, GET element by ID is implemented. The Post, Put, Delete operations are also performed. This operations are verified by using Postman.

### **4. Basic Authentication is performed for an endpoint to GET the books list.**

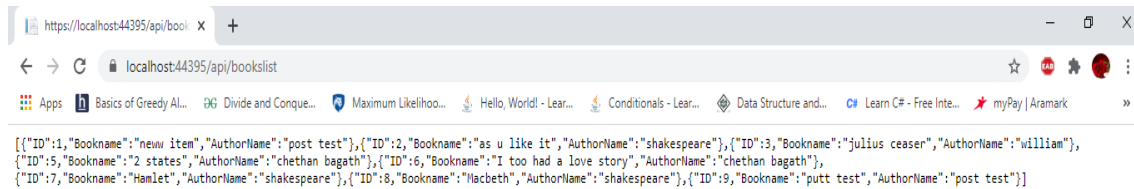
Basic Authentication is implemented when we want to get a list of Books from the database. So that the difference can be observed with and without Basic Authentication for other methods. The user gets the information if he gives the username and password in the form of base64 encoded string of username:password. Here the username and password both are given the Author Name such that the when the user makes a call. The books information written by that particular author is attained. This operation is also tested using postman.

### **5. Verification of the results for the given endpoints.**

Endpoints are created and tested for different CRUD (Create, Read, Update, Delete) operations on Books and Author related data. The results are shown below:

- i. Implementation of Get Operation to display all the books list

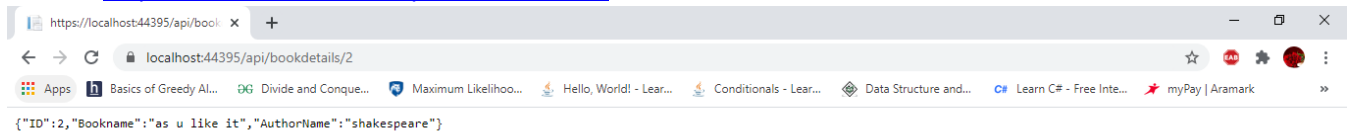
<https://localhost:44395/api/bookslist>



```
[{"ID":1,"Bookname":"new item","AuthorName":"post test"}, {"ID":2,"Bookname":"as u like it","AuthorName":"shakespeare"}, {"ID":3,"Bookname":"julius ceaser","AuthorName":"william"}, {"ID":5,"Bookname":"2 states","AuthorName":"chethan bagath"}, {"ID":6,"Bookname":"I too had a love story","AuthorName":"chethan bagath"}, {"ID":7,"Bookname":"Hamlet","AuthorName":"shakespeare"}, {"ID":8,"Bookname":"Macbeth","AuthorName":"shakespeare"}, {"ID":9,"Bookname":"putt test","AuthorName":"post test"}]
```

- ii. Implementation of Get Operation by ID to display the book details based on the book ID.

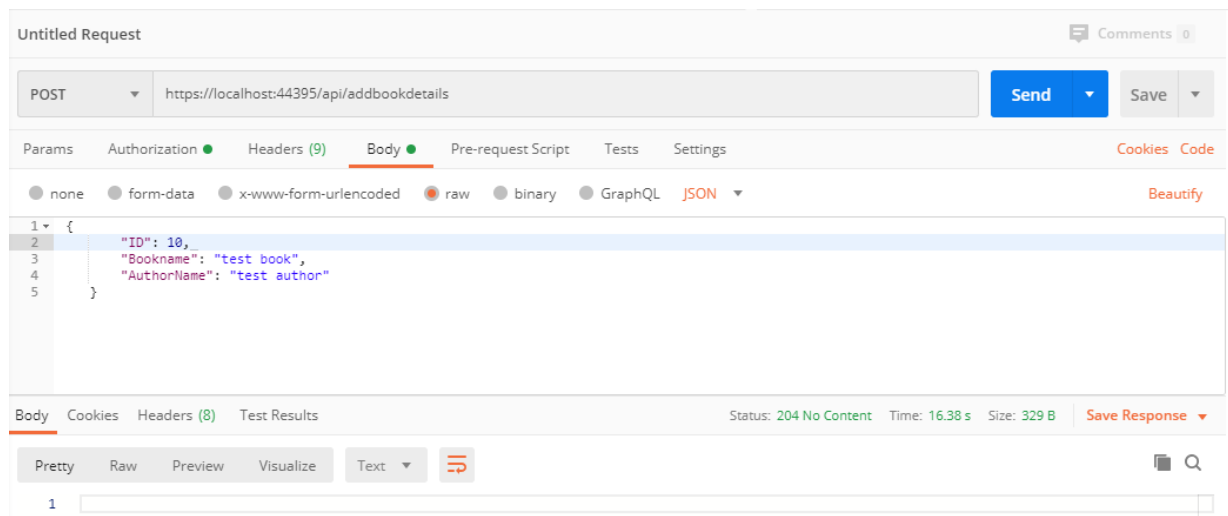
<https://localhost:44395/api/bookdetails/2>



```
{"ID":2,"Bookname":"as u like it","AuthorName":"shakespeare"}
```

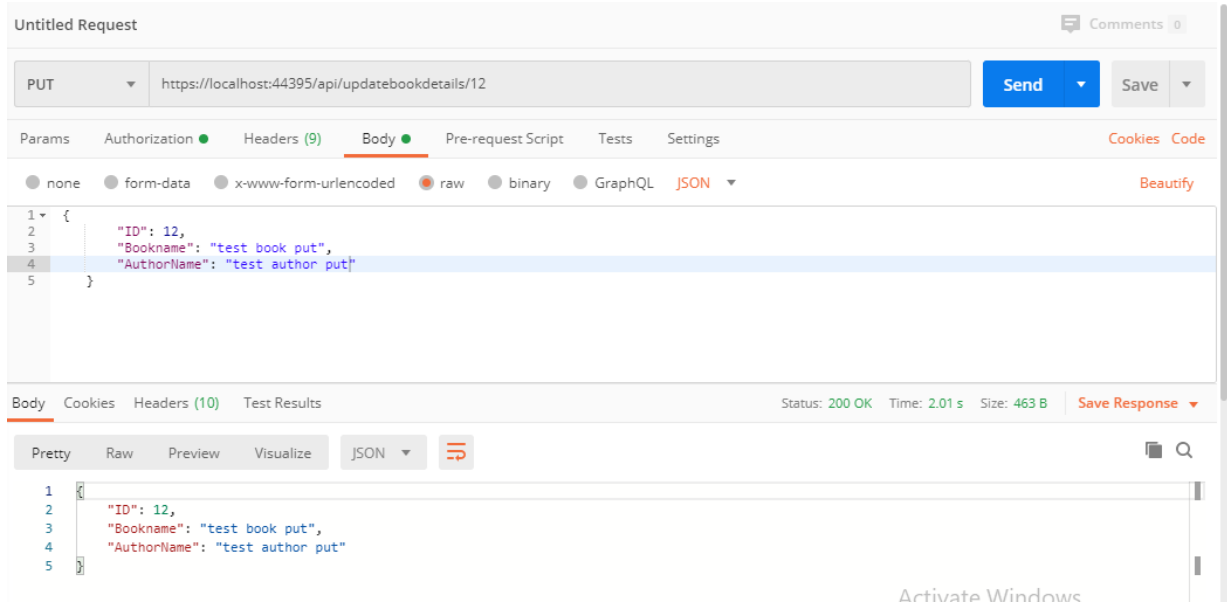
- iii. Implementation of Post Operation for adding a new book details in postman.

<https://localhost:44395/api/addbookdetails>

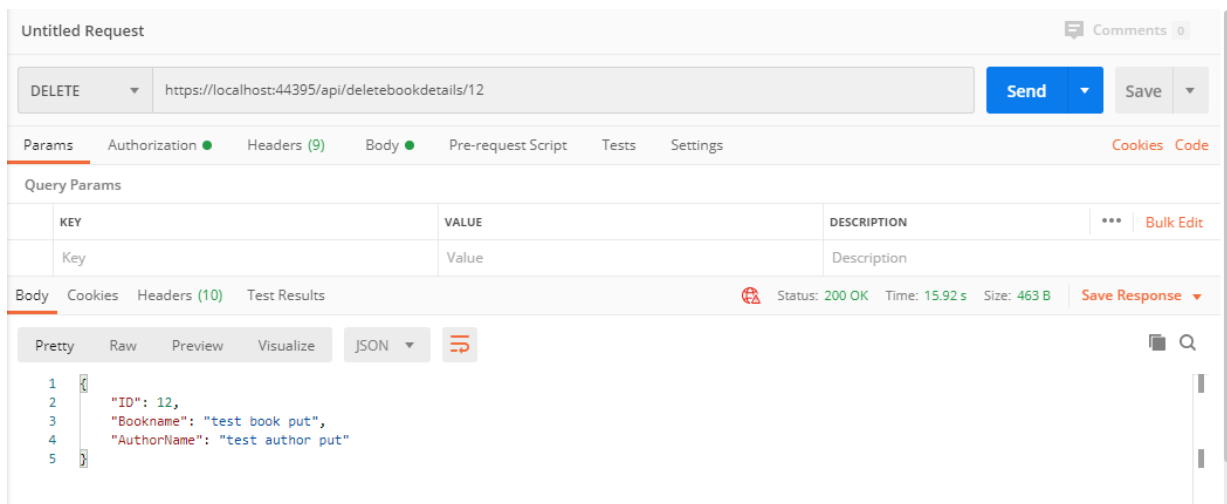


- iv. Implementation of Put Operation for updating the existing book details in postman based on the book ID.

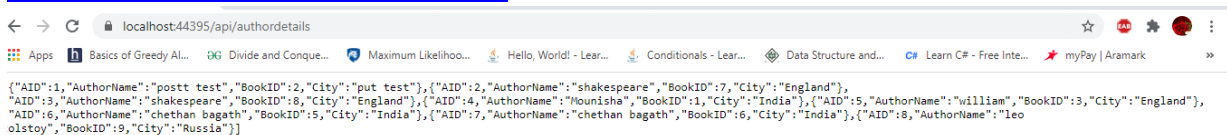
<https://localhost:44395/api/updatebookdetails/12>



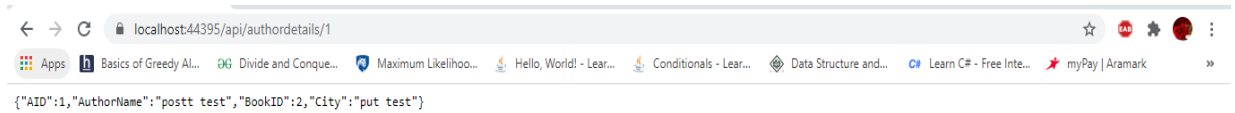
- v. Implementation of delete Operation for deleting a book information in postman by using its ID.  
<https://localhost:44395/api/deletebookdetails/12>



- vi. Implementation of Get Operation to display all the Authors details  
<https://localhost:44395/api/authordetails>

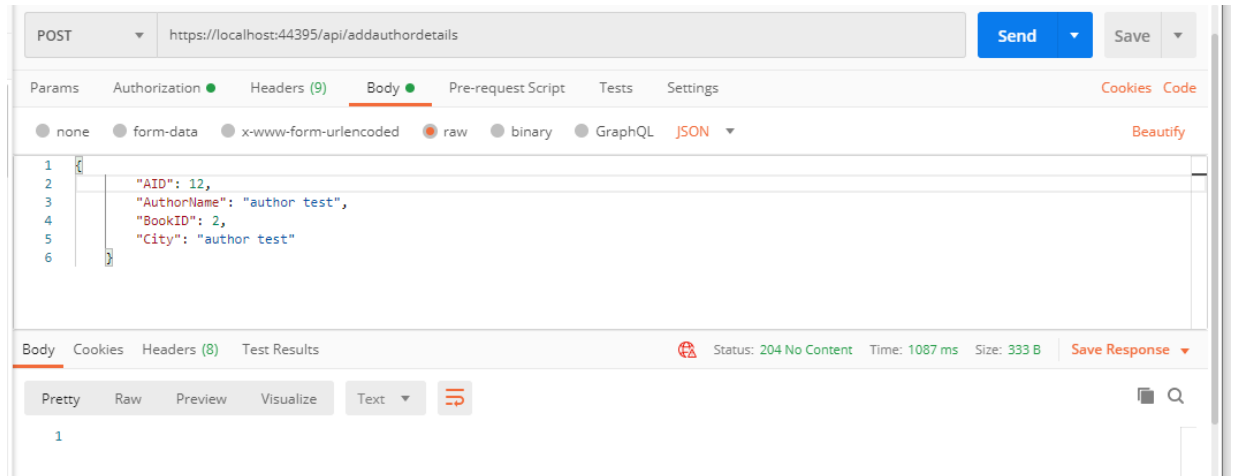


- vii. Implementation of Get Operation by ID to display the Authors details based on the Author ID.  
<https://localhost:44395/api/authordetails/1>



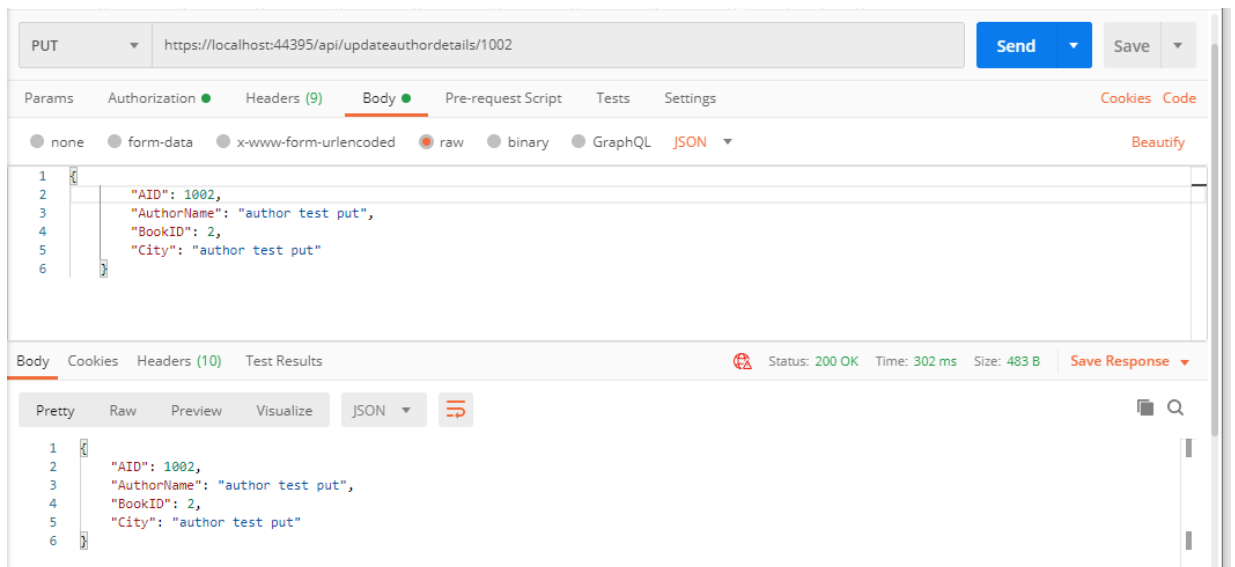
- viii. Implementation of Post Operation for adding a new Author details in postman.

<https://localhost:44395/api/addauthordetails>



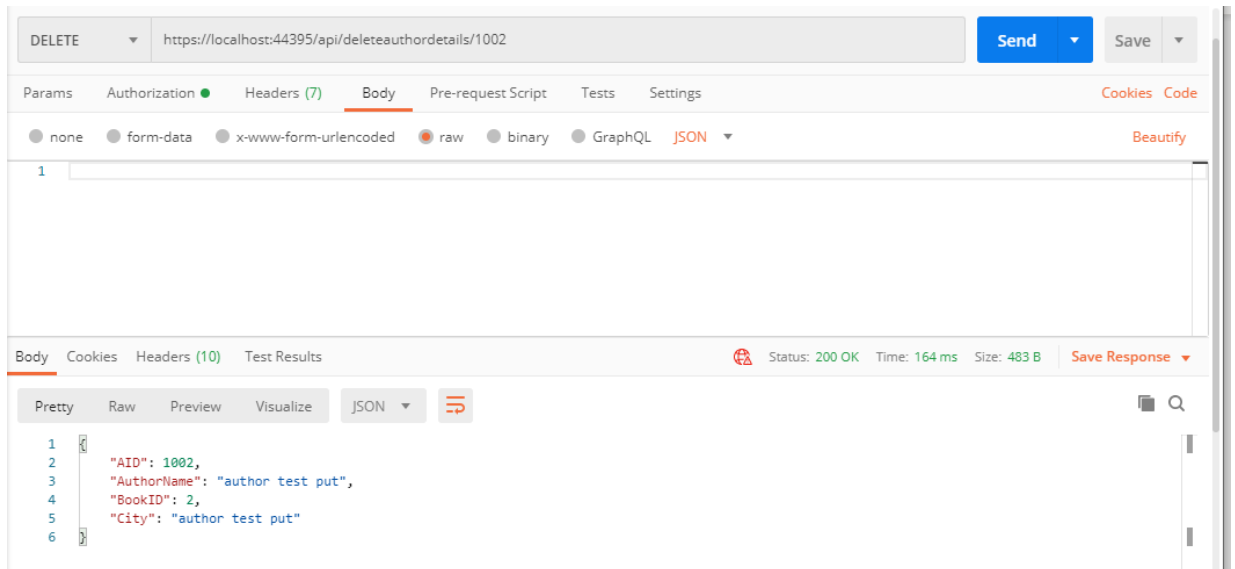
- ix. Implementation of Put Operation for updating the existing Author details in postman based on the Author ID.

<https://localhost:44395/api/updateauthordetails/1002>



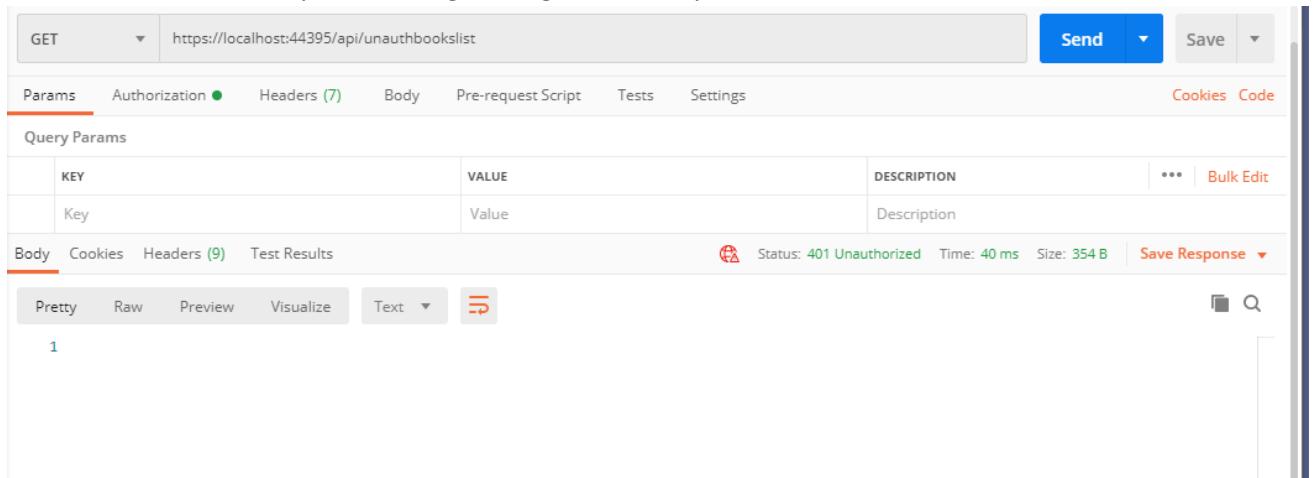
- x. Implementation of delete Operation for deleting a Author information in postman by using its ID.

<https://localhost:44395/api/deleteauthordetails/1002>



- xi. Implementation of Basic Authentication for getting a list of books written by an author.  
<https://localhost:44395/api/unauthbookslist>

When no username and password is given it gives the output as 401 unauthorized error



When username and password is given as William in the header then it gives the list of books written by author William

Untitled Request

Comments 0

GEThttps://localhost:44395/api/unauthbookslist

SendSave

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

william

Password

\*\*\*\*\*

☐ Show Password

BodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 52 msSize: 452 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "ID": 3,
4     "Bookname": "julius ceaser",
5     "AuthorName": "william"
6   }
7 ]
```