## **Objectives**

* Explain about ASP.NET Web API PUT method
* Explain how EF Database First Approach and ASP.NET Web API can be used to modify data which is stored in the SQL Server

In this hands-on lab, you will learn how to:

* Implement an ASP.NET Web API service that modifies data that is stored in the SQL Server using EF6 Database First Approach
* Implement the PUT method in ASP.NET Web API
* Use Postman to issue Put request to the Web API service

**Prerequisites**

The following is required to complete this hands-on lab:

* Visual Studio 2017 Community Edition
* Web API 2.x
* SQL Server 2014
* Postman

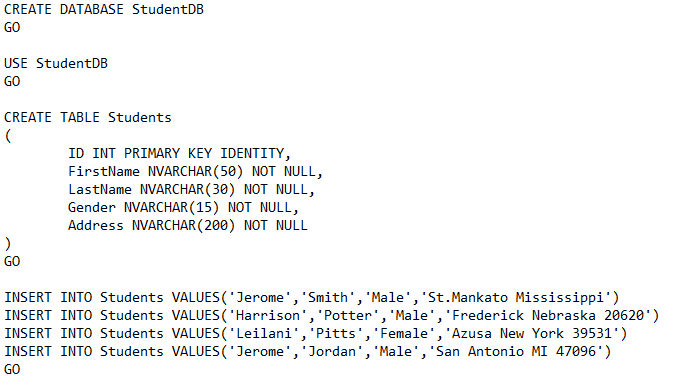
## **Notes**

Estimated time to complete this lab: **90 minutes.**

**Task 1 – Create database table to perform update operation**

**Note:** Re-use the database schema that you created for the previous exercise if exists, else follow the below steps for creating a new one.

1. Create a database called **StudentDB** in SQL Server
2. Creates the Students table and populate it with some test data as shown in the below image.

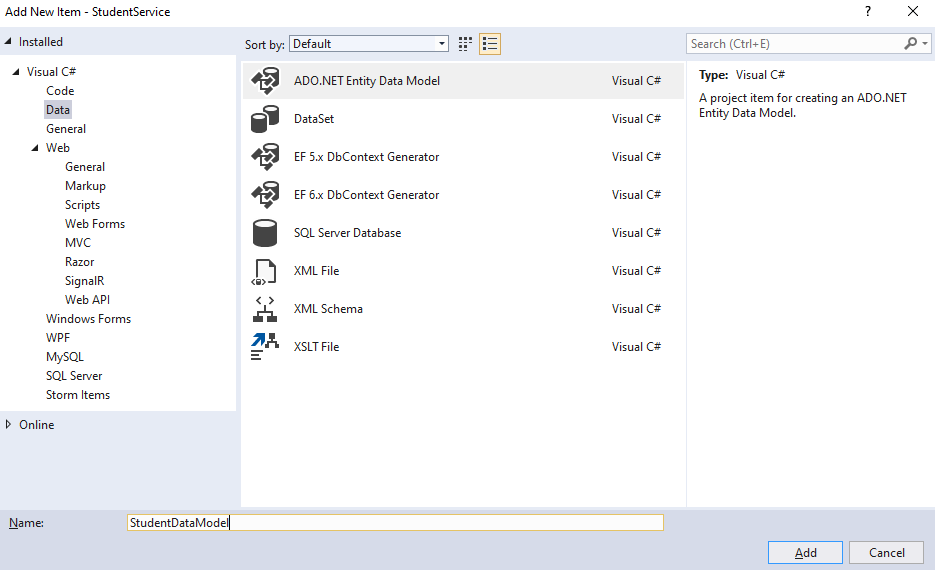


**Task 2-Creating a new ASP.NET Web API Project**

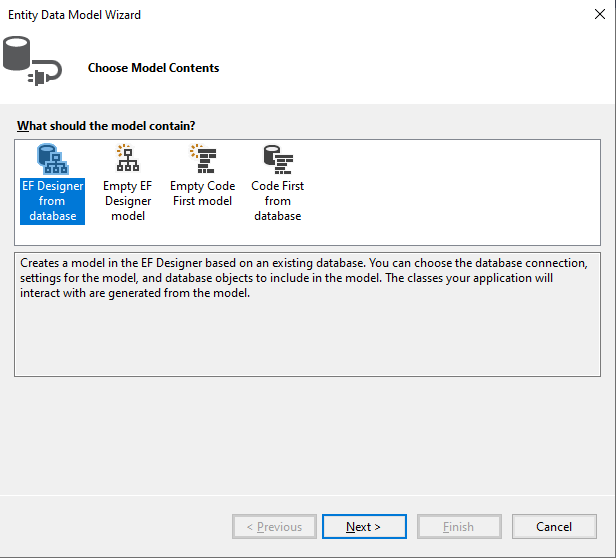
1. Refer previous exercises for creating a new ASP.NET Web API project.
2. Name your Web API project **StudentService**

**Task 3-Adding ADO.NET Entity Data Model to update data into database**

1. Right-click on the Models folder and then select **Add – New Item** option which will open the Add New Item window and from the **“Add New Item”** window select the **“Data”** option from the left pane and from the middle pane select **ADO.NET Entity Data Model**. In the Name text box, type **StudentDataModel** and finally click the **Add** button as shown in the below image.



1. Once you click on the **Add** button, it will open the **Entity Data Model Wizard** and from that wizard select “**EF Designer from database”** option and click on the “**Next”** button as shown below.



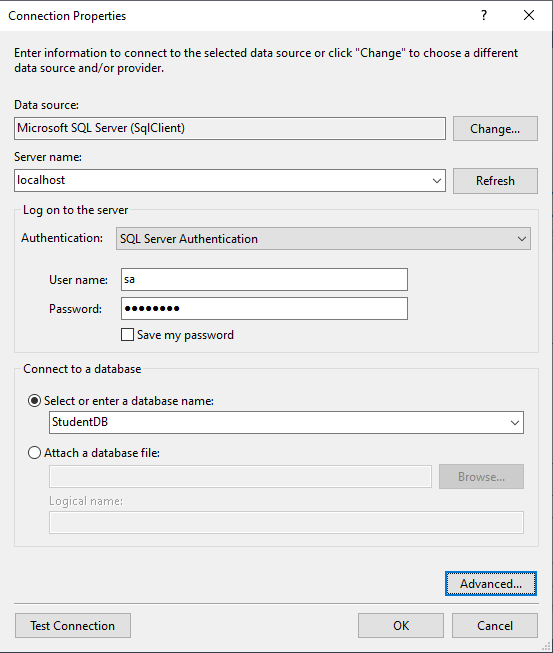
1. On the next screen, click on the “**New Connection**” button
2. Once you click on the **New Connection** Button it will open the **Connection Properties** window. On “**Connection Properties**” window, set

**Server Name =** <your\_server\_name>

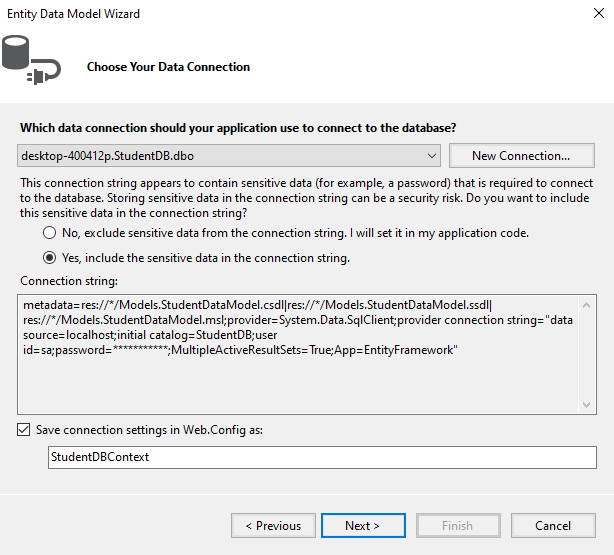
**Authentication =** Select the authentication type

**Select or enter a database name =** StudentDB

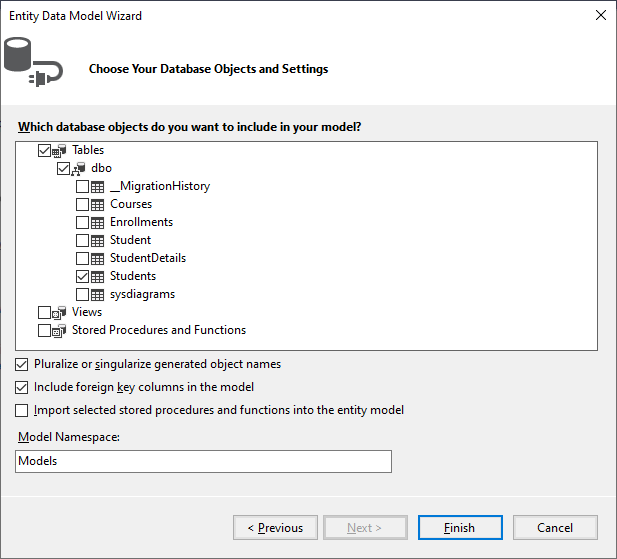
Click the **OK** button as shown below.



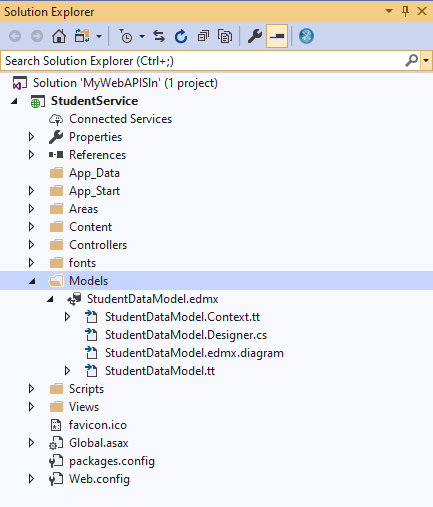
1. Once you click on the **OK** button it will navigate back to the Choose Your Data Connection wizard. Here Modify the Connection String as **StudentDBContext** and click on the Next Button as shown in the below image.



1. On the next screen, make sure that “Entity Framework 6.x” is selected.
2. On Choose Your Database Objects and Settings screen, select the **“Students”** table, provide the model namespace name and click on Finish button as shown below.



1. Once you click on the Finish Button the following edmx file will be generated within the Models folder as shown below.

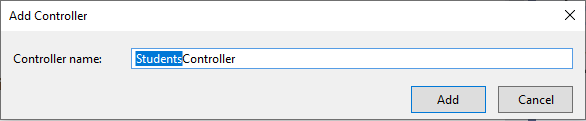


##### Task 4 - ****Adding Web API Controller****

##### Right-click on the Controllers folder and select****Add – Controller****option and then select “****Web API 2 Controller – Empty****” and click on the “****Add****” button as shown below.

##### 

1. On the next screen set, the Controller Name as **StudentsController** and click on the **Add** button as shown in the below image.



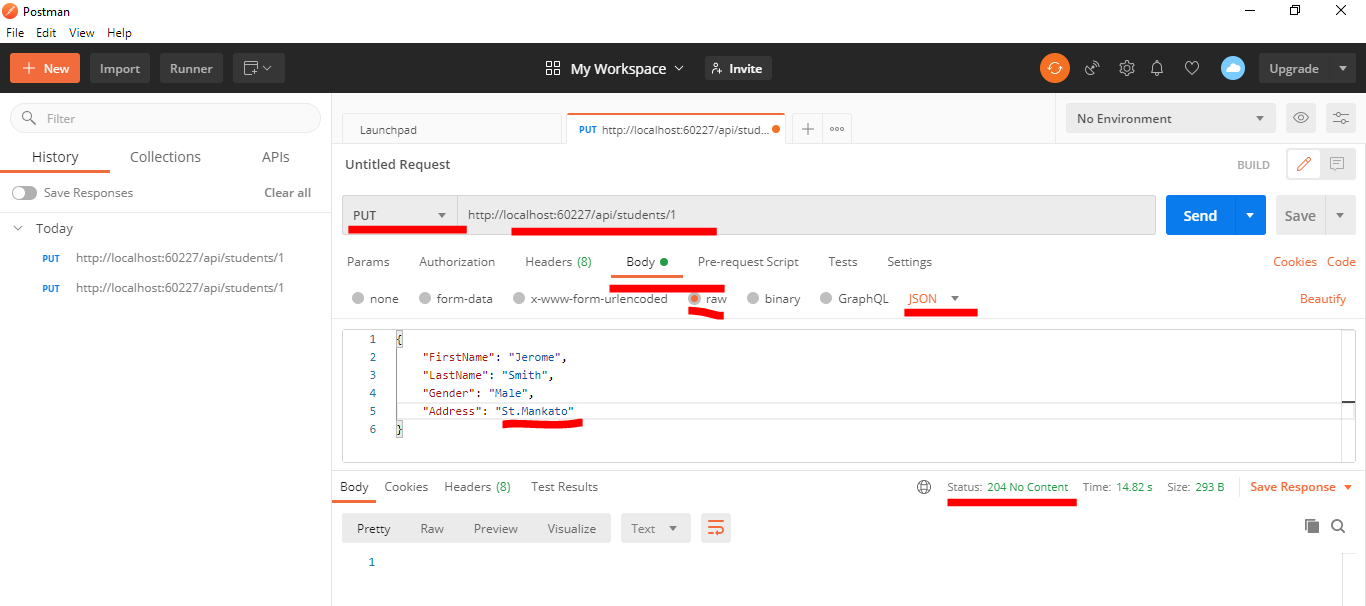
##### Task 5 - ****Implementing the PUT method in ASP.NET Web API****

##### Add the following code in StudentsController.cs

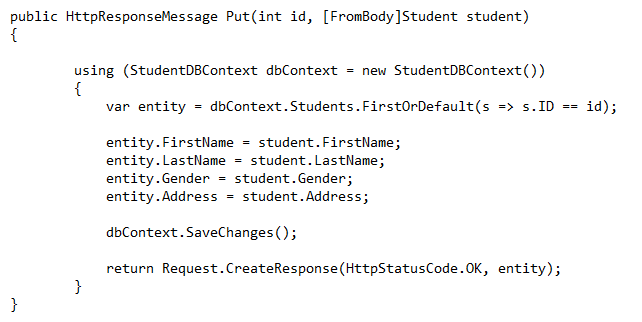
##### 

**Task 6 – Run your application and observe the result using a Web API client tool like Postman**

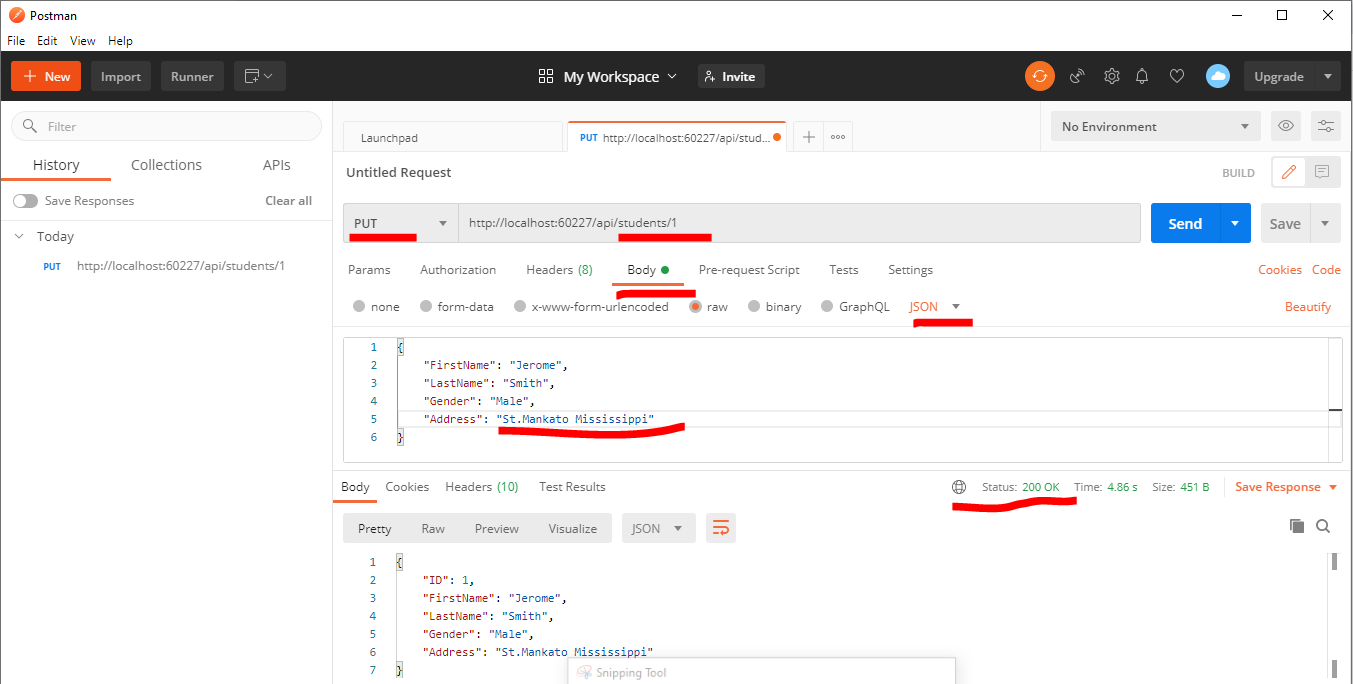
1. Set the HTTP verb to **PUT**
2. In the Request **Body**, include the updated student object with which you want to update. Try updating the address field
3. Choose **JSON** as the format
4. Finally, click on the **Send** button as shown below



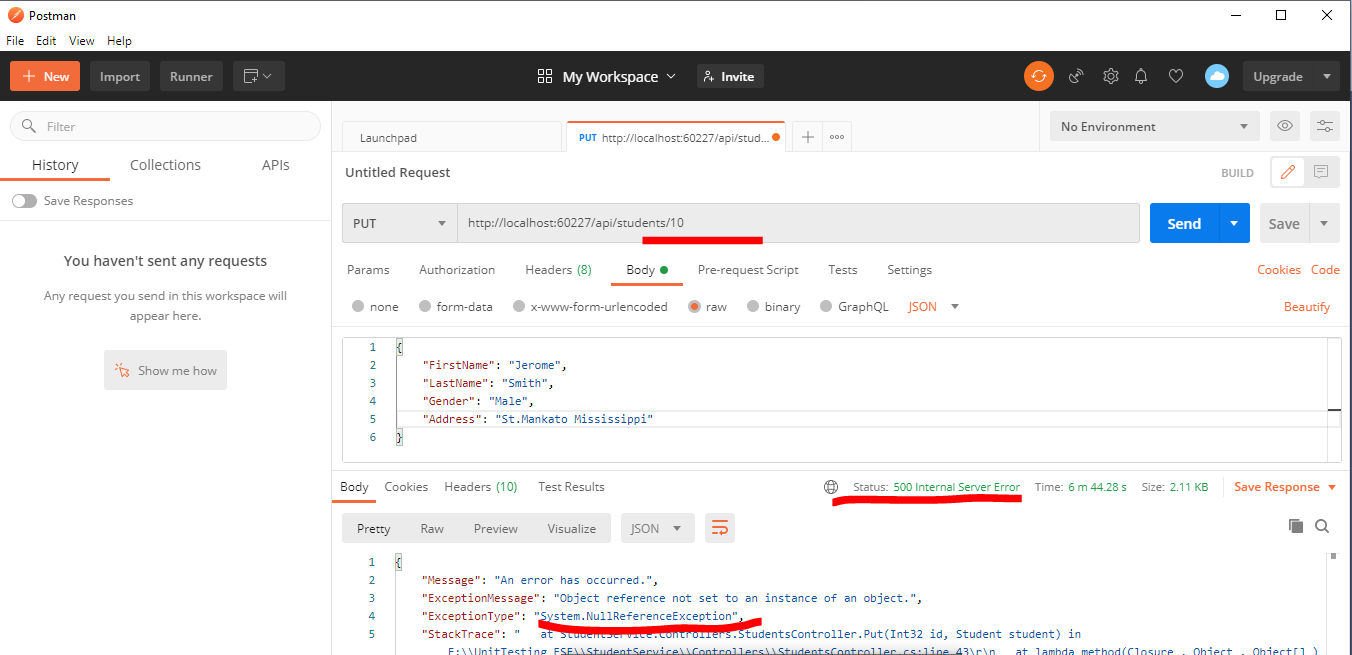
1. It will be working fine and updates the respective student record in the database as expected. Since the return type of the Put method is void, we get status **code 204 No Content**. When the update is successful, we want to return status code **200 OK** indicating that the update is successful.
2. Modify your Put action as shown below in order to achieve the above goal.



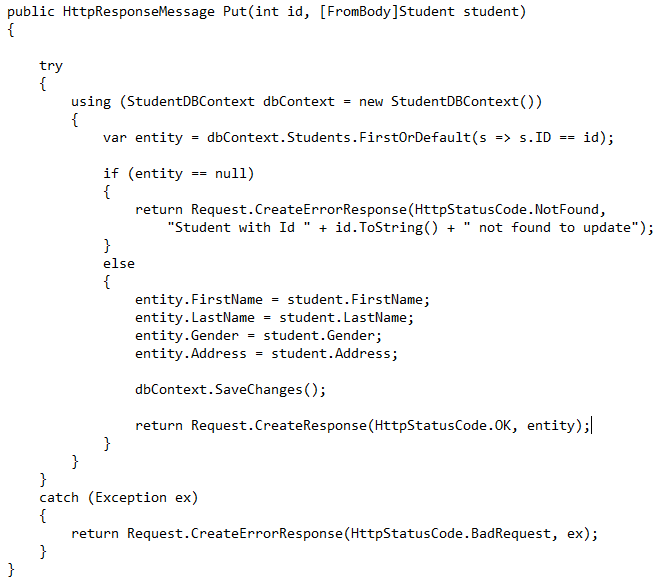
1. After modifying the code, rebuild your API Project and execute again.
2. Verify whether you are getting the response as expected as shown in the below picture.



1. Next, try updating a student record in the database with an invalid id. You will get the following response. We HTTP status code **500** **Internal Server Error**, because of a NULL reference exception.



1. In order to fix this issue modify the code in the Put method as shown below.



1. After updating, rebuild your Web API and execute. This time you will get status code **404 Not Found** instead of 500 Internal Server Error which is more meaningful in this context.

