**M.Sc. Computer Science Semester-I**

**Name – Abhijeet Ashok Chauhan**

**Roll No - 527**

**Elective Paper I (**Trends in Cloud Computing**)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | | | | |
| **NO** | **DATE** | **TITLE** | **PAGE NO** | **SIGN** |
|  | | | | |
| 1 |  | SOAP Webservices |  |  |
| 2 |  | Create RESTful Services to perform CRUD operation |  |  |
| 3 |  | Create RESTful Services from Pattern |  |  |
| 4 |  | MVC application. |  |  |
| 5 |  | Service Endpoint |  |  |
| 6 |  | Restful services using WEB-API. |  |  |
| 7 |  | Web application using Azure |  |  |
| 8. |  | Install Google App Engine. Create hello world app and other simple web applications using python/java. |  |  |

Practical No 1

**Aim: Create SOAP webservices to find factorial of a number.**

1. SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.
2. SOAP is a W3C recommendation for communication between two applications.
3. SOAP is XML based protocol. It is platform independent and language independent. By using SOAP, you will be able to interact with other programming language applications.

**How does SOAP work?**

1. SOAP requests are easy to generate and process responses. First, a request for a service is generated by a client using an XML document. Next, a SOAP client sends the XML document to a SOAP server.
2. When the server receives the SOAP message, it sends the message as a service invocation to the requested server-side application.
3. A response containing the requested parameters, return values and data for the client is returned first to the SOAP request handler and then to the requesting client.
4. Both SOAP requests and responses are transported using Hypertext Transfer Protocol Secure (HTTPS) or a similar protocol like HTTP.

**Advantages of Soap Web Services**

1. **WS Security:** SOAP defines its own security known as WS Security.
2. **Language and Platform independent:** SOAP web services can be written in any programming language and executed in any platform.

**Disadvantages of Soap Web Services**

1. **Slow:** SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.
2. **WSDL dependent:** SOAP uses WSDL and doesn't have any other mechanism to discover the service.

**Source Code**

**FactorialWs.java**

package abc;

import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

@WebService(serviceName = "FactorialWs")

public class FactorialWs {

@WebMethod(operationName = "factorial")

public int factorial(@WebParam(name = "x") int x) {

int fact =1;

for(int i = 1; i <= x; i++){

fact = fact \* i;

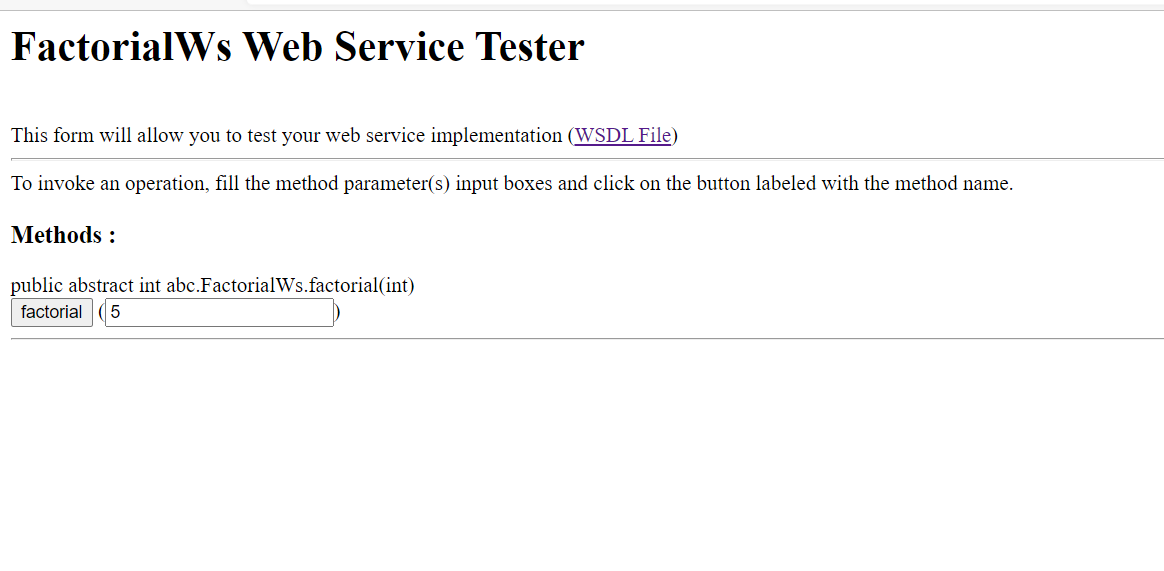
}

return fact;

}

}

**Output**



Practical No 2

**Aim: RESTful Services using Java to perform CRUD operation.**

1. REST or Representational State Transfer is an architectural style that can be applied to web services to create and enhance properties like performance, scalability, and modifiability.
2. RESTful web services are generally highly scalable, light, and maintainable and are used to create APIs for web-based applications. It exposes API from an application in a secure and stateless manner to the client.
3. The protocol for REST is HTTP.
4. In this architecture style, clients and servers use a standardized interface and protocol to exchange representation of resources.
5. REST emerged as the predominant Web service design model just a couple of years after its launch, measured by the number of Web services that use it.
6. Owing to its more straightforward style, it has mostly displaced SOAP and WSDL-based interface design.

**Advantages of RESTful web services**

1. **Speed:** As there is no strict specification, RESTful web services are faster as compared to SOAP. It also consumes fewer resources and bandwidth.
2. **Compatible with SOAP:** RESTful web services are compatible with SOAP, which can be used as the implementation.
3. **Language and Platform Independency:** RESTful web services can be written in any programming language and can be used on any platform.
4. **Supports Various Data Formats:** It permits the use of several data formats like HTML, XML, Plain Text, JSON, etc.

**CRUD**

1. CRUD refers to the four basic operations a software application should be able to perform – Create, Read, Update, and Delete.
2. In such apps, users must be able to create data, have access to the data in the UI by reading the data, update or edit the data, and delete the data.
3. In full-fledged applications, CRUD apps consist of 3 parts: an API (or server), a database, and a user interface (UI).
4. The API contains the code and methods, the database stores and helps the user retrieve the information, while the user interface helps users interact with the app.
5. You can make a CRUD app with any of the programming languages out there. And the app doesn’t have to be full stack – you can make a CRUD app with client-side JavaScript.
6. In fact, the app with which I will be showing you how create, read, update and delete operations work is made with client-side JavaScript.
7. Each letter in the CRUD acronym has a corresponding HTTP request method.

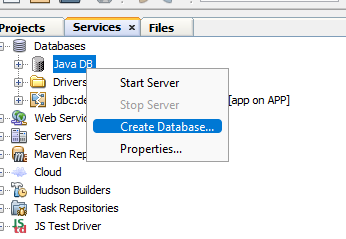
**Creating Restful Webservice**

**Step 1: Create a New Java Web Application Project**

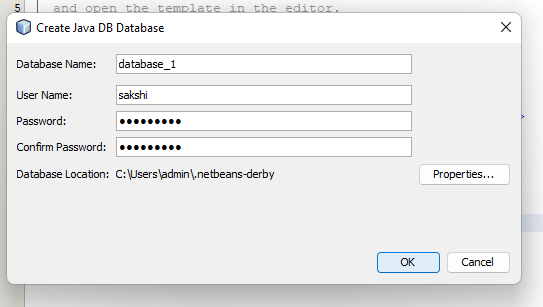
1. Open NetBeans IDE.
2. Click on "File" > "New Project..."
3. Choose "Java Web" > "Web Application" and click "Next."
4. Give your project a name and choose a location. Click "Next."
5. Select "JavaServer Faces" as the framework and "GlassFish" as the server. Click "Finish."

**Step 2: Set Up the Database**

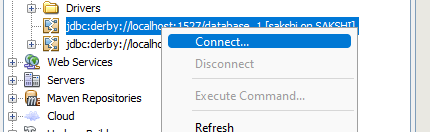
1. Services >> Java DB >> create Database.



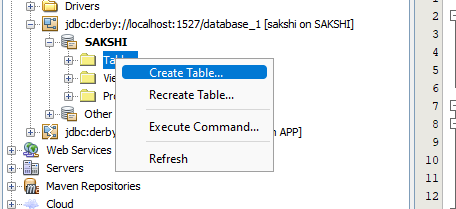
1. Provide a Database name and valid username and password, and then create Database.

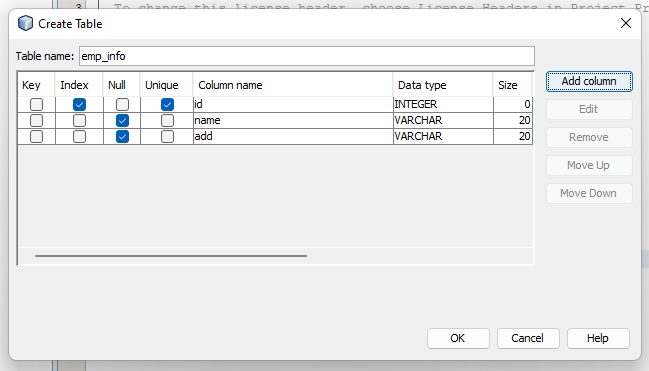


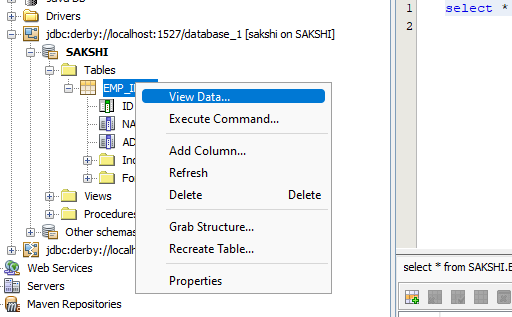
1. Connect the Database

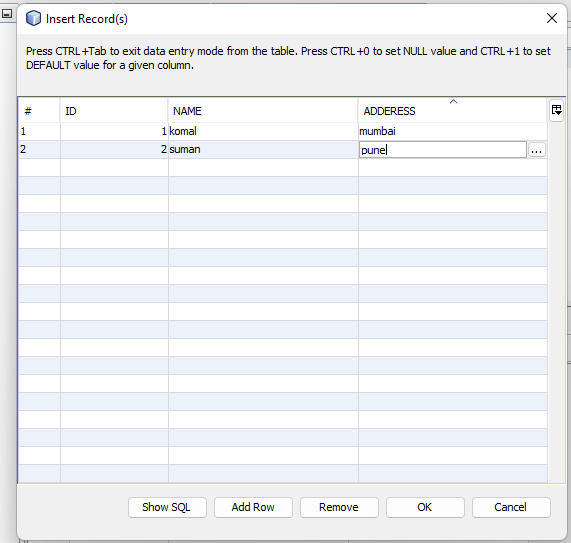


**Step 3: Create table in My Database**



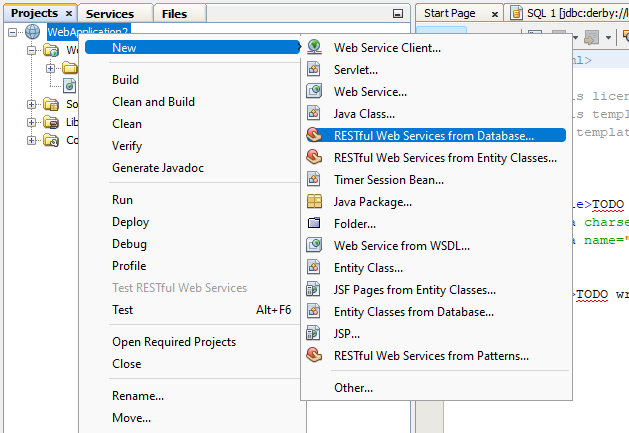


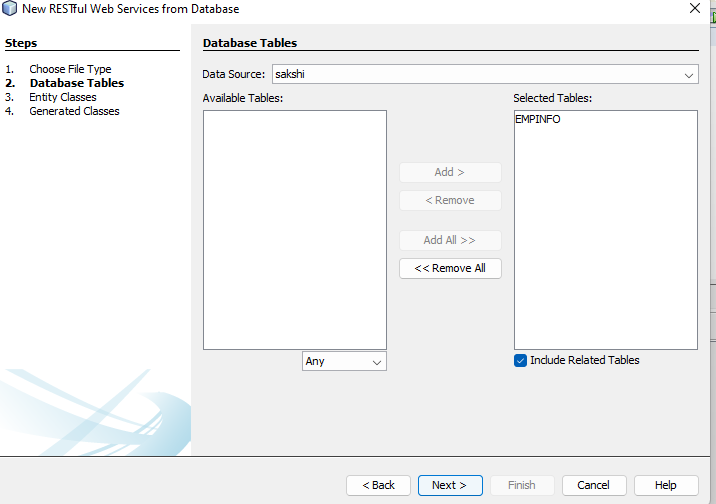




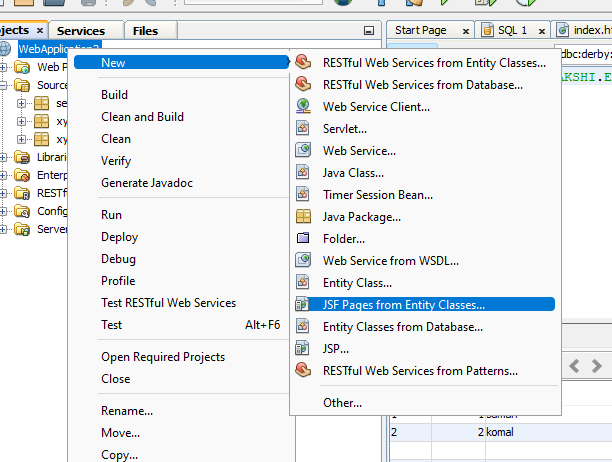


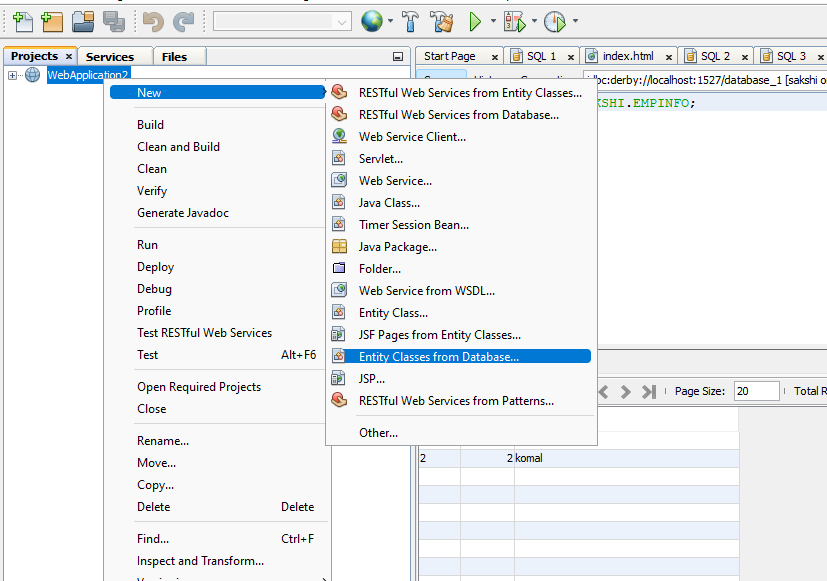
**Step 4: Under project name >> New >> Select RESTful Webservices from Database**



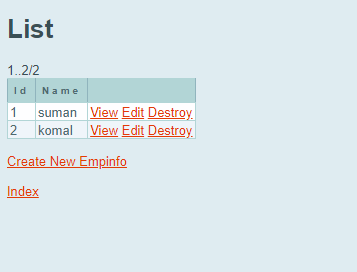




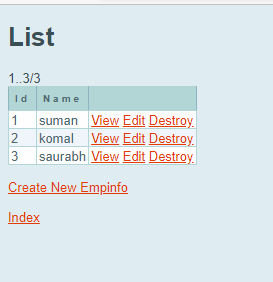




Then Clean and Build >> Deploy >> Run



Add 1 more entry.



Practical No 3

**Aim: Create Restful Services from Pattern**

1. REST is an architectural style for designing networked applications, and it defines a set of constraints to be used when creating web services.
2. RESTful web services are generally highly scalable, light, and maintainable and are used to create APIs for web-based applications. It exposes API from an application in a secure and stateless manner to the client.
3. Below, I'll elaborate on RESTful services.

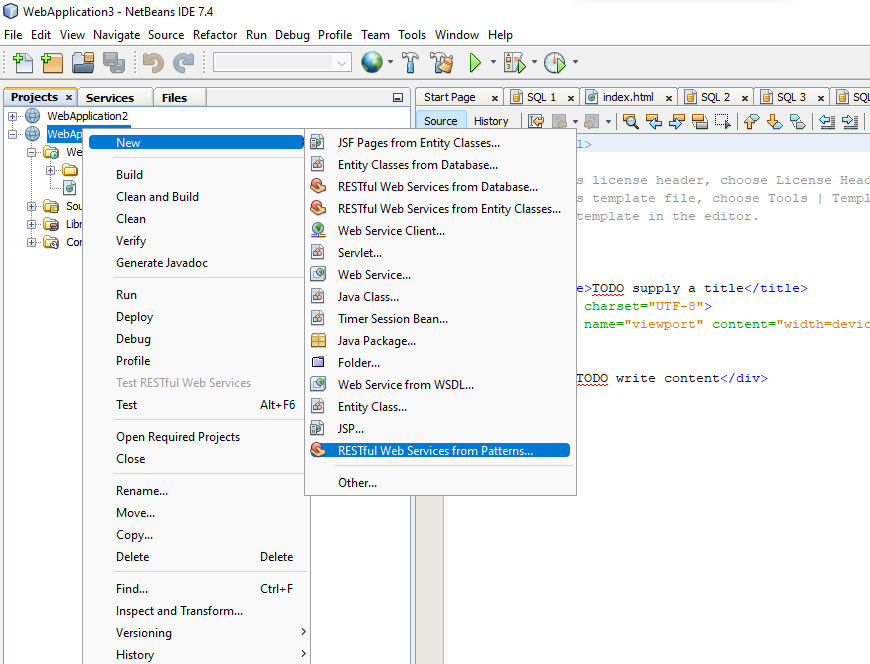
**HTTP Methods (CRUD Operations):**

* Create (POST): Used to create a new resource.
* Read (GET): Used to retrieve a representation of a resource.
* Update (PUT or PATCH): Used to modify an existing resource.
* Delete (DELETE): Used to remove a resource.

**Step 1: Create a New Java Web Application Project**

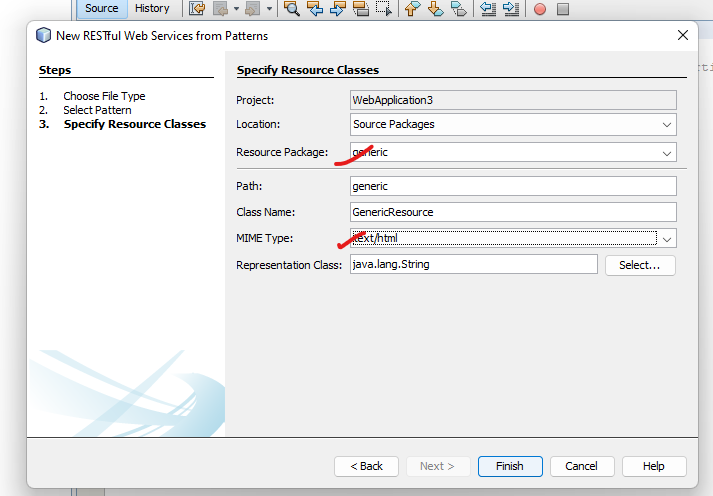
1. Open NetBeans IDE.
2. Click on "File" > "New Project..."
3. Choose "Java Web" > "Web Application" and click "Next."
4. Give your project a name and choose a location. Click “Next."
5. Select "JavaServer Faces" as the framework and "GlassFish" as the server. Click "Finish."

**Step 2: Create web application and add restful web services from pattern.**

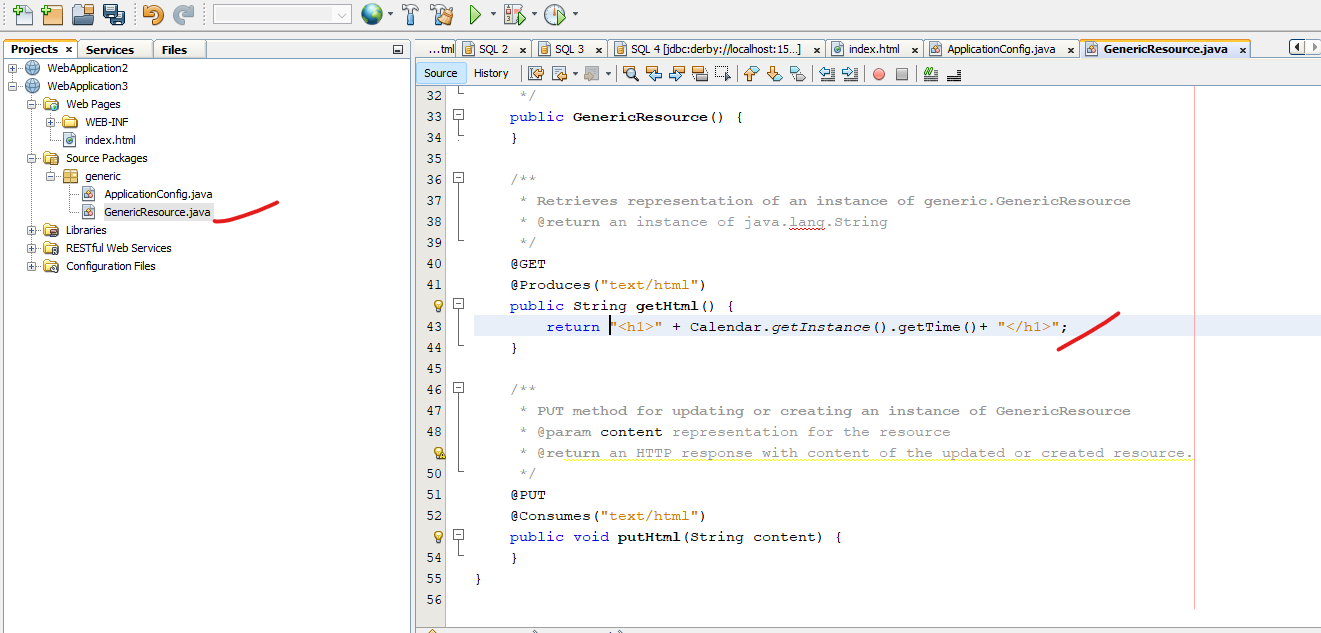


**Step 3: New Restful Web Service from Pattern**

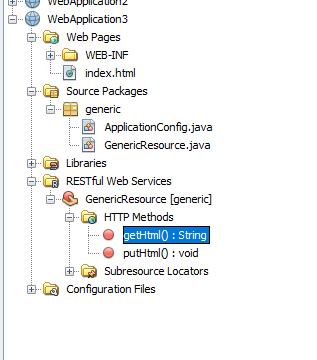
1. Fill resource package name
2. Add MIME Type – text/html



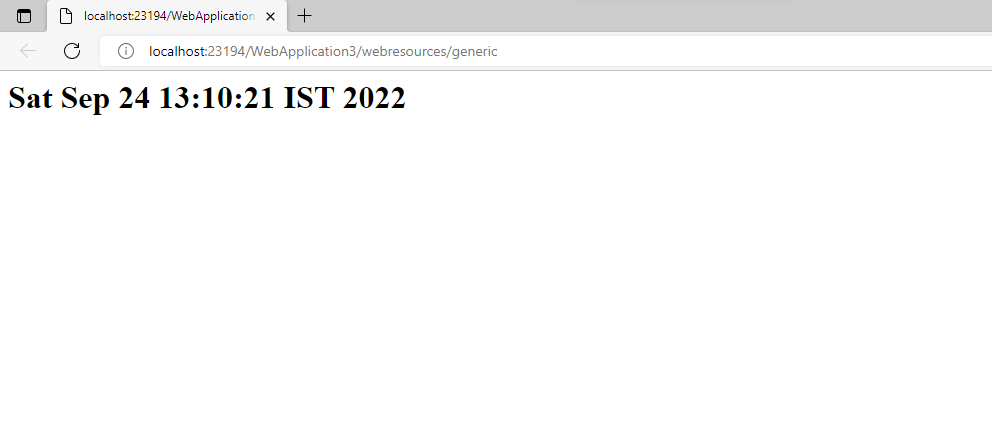
**Write code in generic source…**



**After writing code clean and build web application->deploy->Test “getHtml():String”**



**Output**



Practical No 4

**Aim: To Create an MVC application.**

1. Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements.
2. These elements are the internal representations of information (the Model), the interface (the View) that presents information to and accepts it from the user, and the Controller software linking the two.
3. Traditionally used for desktop graphical user interfaces (GUIs), this pattern became popular for designing web applications.
4. Popular programming languages have MVC frameworks that facilitate the implementation of the pattern.

**Features of MVC**

1. It provides a clear separation of business logic, Ul logic, and input logic.
2. It offers full control over your HTML and URLs which makes it easy to design web application architecture.
3. It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs.
4. It supports Test Driven Development (TDD).

**Components of MVC**

The MVC framework includes the following 3 components:

1. Controller
2. Model
3. View

**Controller**

1. The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary.
2. The controller doesn’t have to worry about handling data logic, it just tells the model what to do.
3. It processes all the business logic and incoming requests, manipulates data using the Model component, and interact with the View to render the final output.

**View**

1. The View component is used for all the UI logic of the application.
2. It generates a user interface for the user.
3. Views are created by the data which is collected by the model component but these data aren’t taken directly but through the controller. It only interacts with the controller.

**Model**

1. The Model component corresponds to all the data-related logic that the user works with.
2. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.
3. It can add or retrieve data from the database. It responds to the controller’s request because the controller can’t interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

**Advantages of MVC**

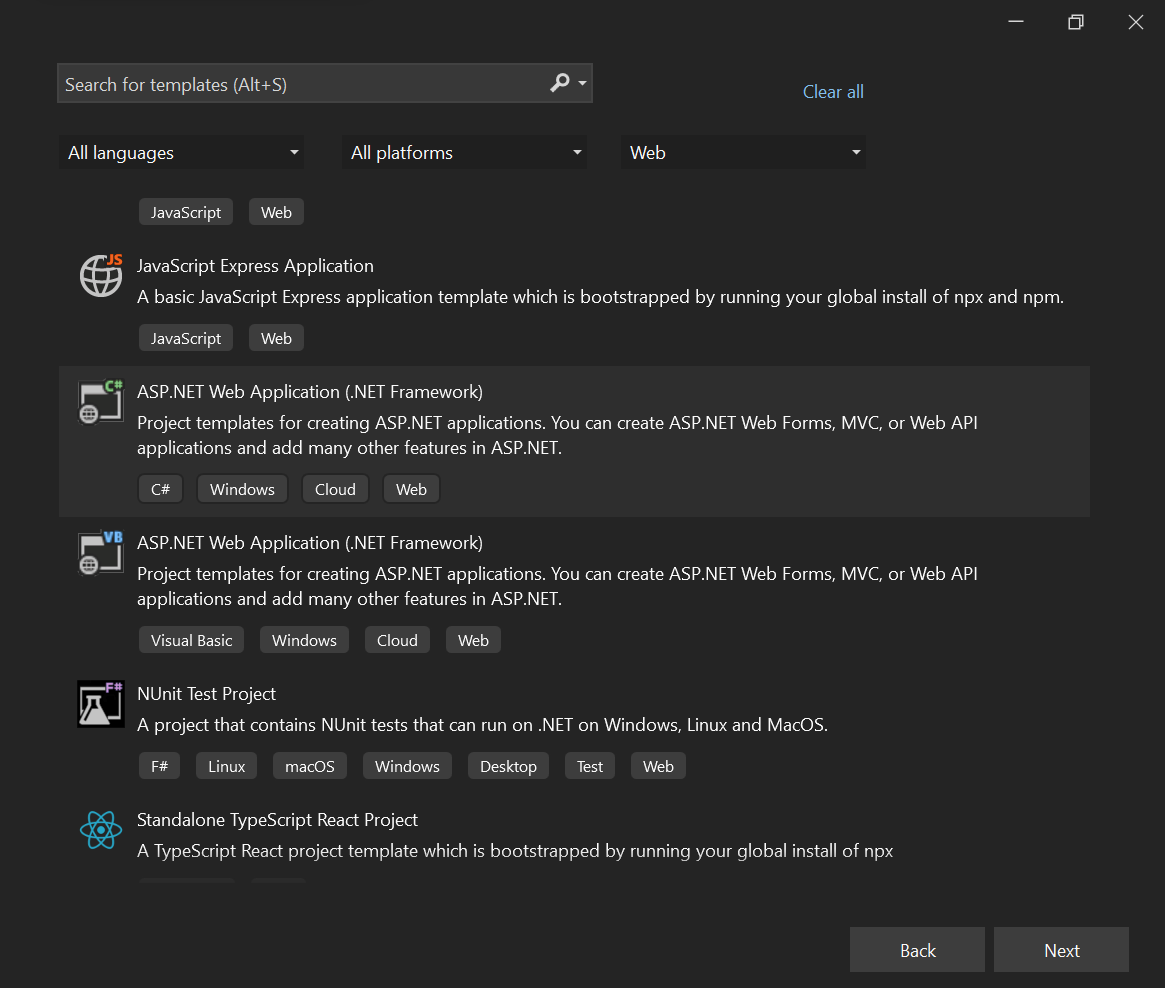
1. Codes are easy to maintain and they can be extended easily.
2. The MVC model component can be tested separately.
3. The components of MVC can be developed simultaneously.
4. It reduces complexity by dividing an application into three units. Model, view, and controller.
5. It supports Test Driven Development (TDD).
6. It works well for Web apps that are supported by large teams of web designers and developers.
7. This architecture helps to test components independently as all classes and objects are independent of each other
8. Search Engine Optimization (SEO) Friendly.

**Disadvantages of MVC**

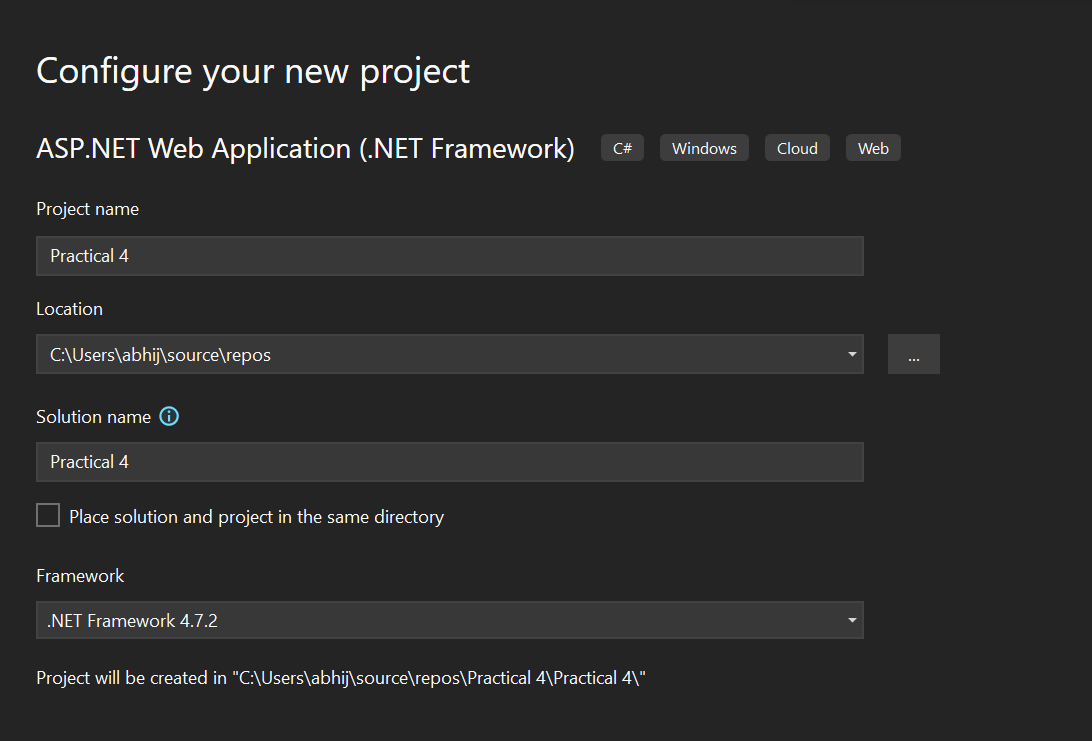
1. It is difficult to read, change, test, and reuse this model
2. It is not suitable for building small applications.
3. The inefficiency of data access in view.
4. The framework navigation can be complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.
5. Increased complexity and Inefficiency of data

**Step 1: Open Visual Studio and Create a New Project**

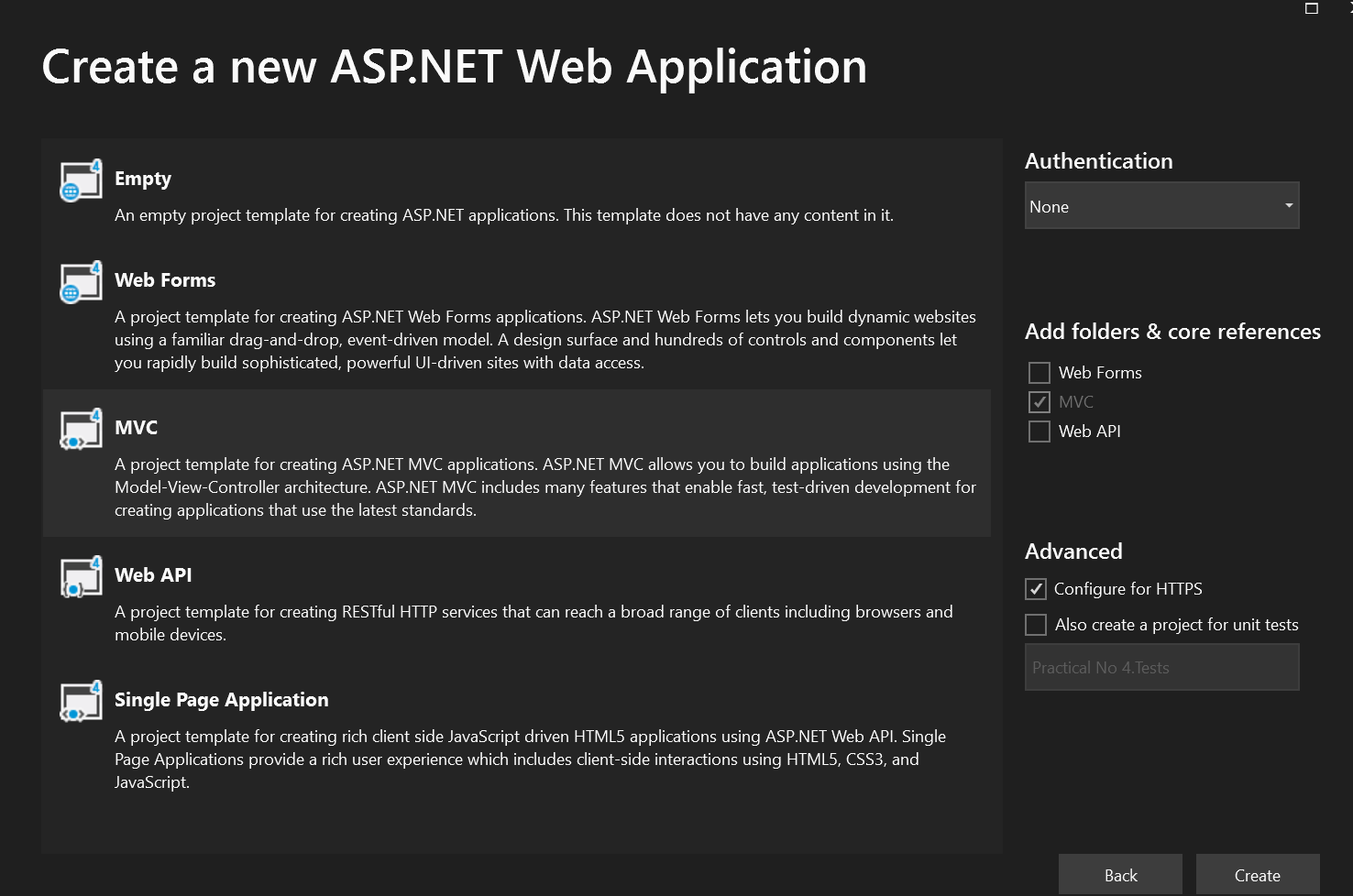
1. In the "Create a new project" dialog, select "ASP.NET Core Web Application" from the list.



1. Choose a name and location for your project.

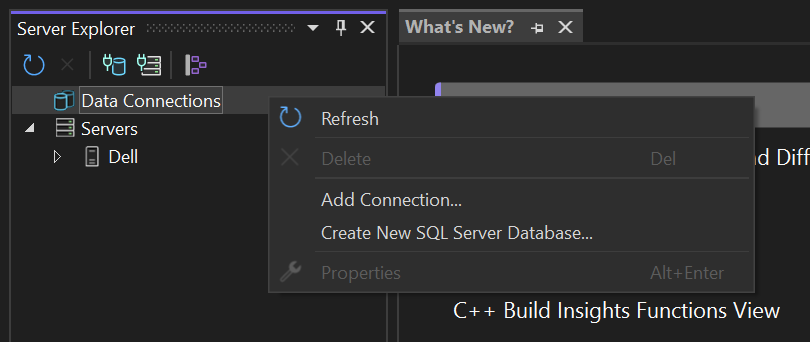


1. Choose the "Web Application (Model-View-Controller)" template. Click "Create."

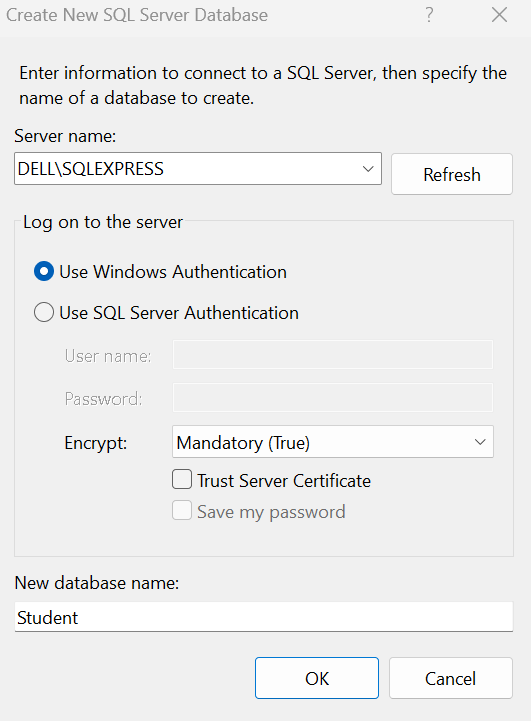


**Step 2: Open Server Explorer**

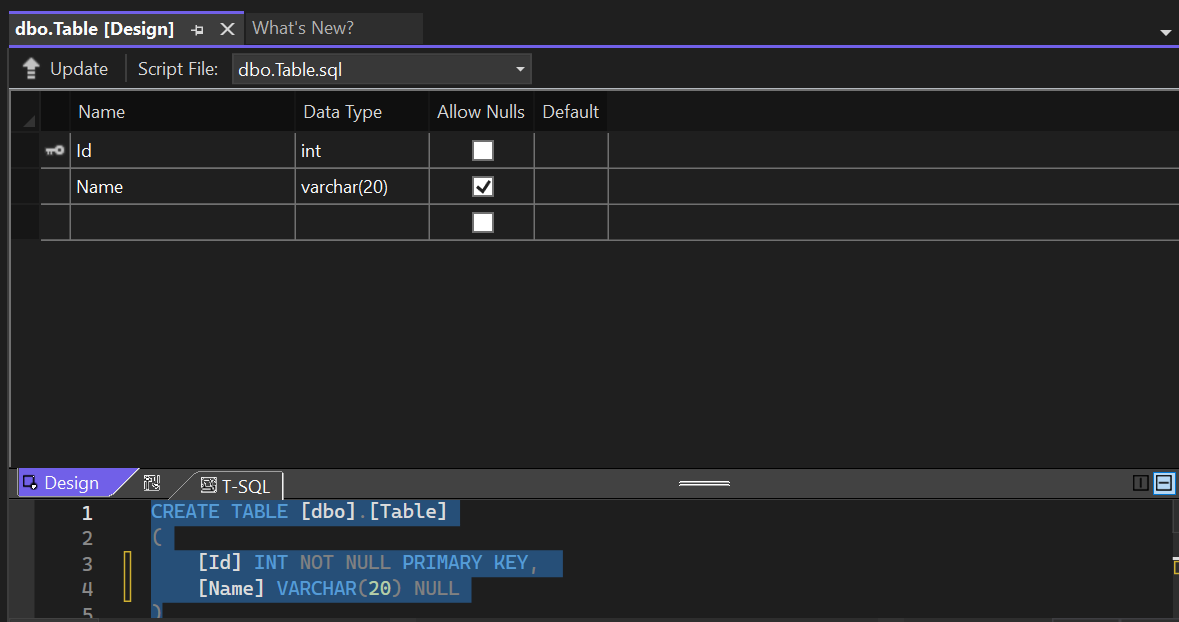
1. In Visual Studio, go to the "View" menu >> Select "Server Explorer" from the dropdown.
2. Connect to a Database >> In the Server Explorer, right-click on "Data Connections" >> Select "Add Connection."

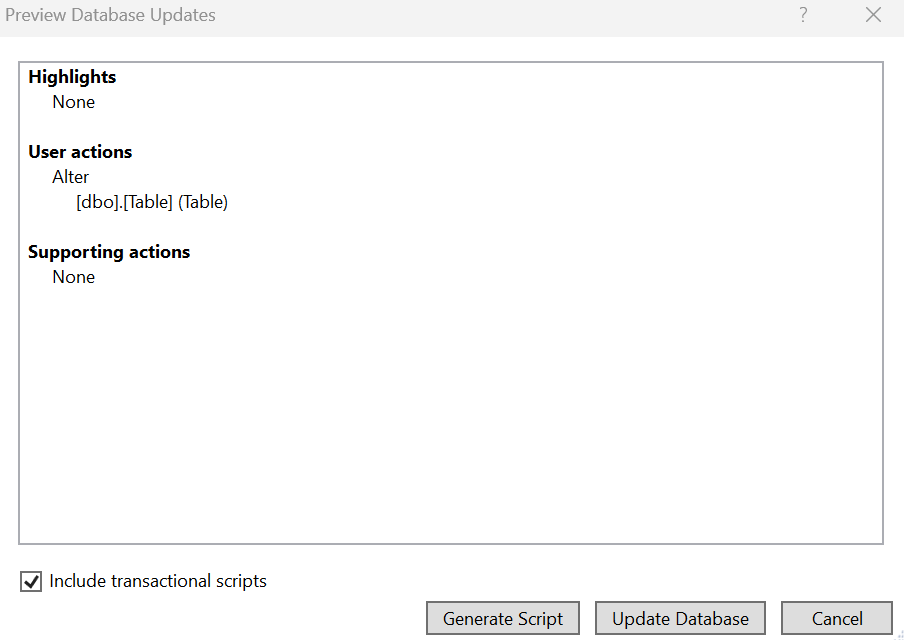


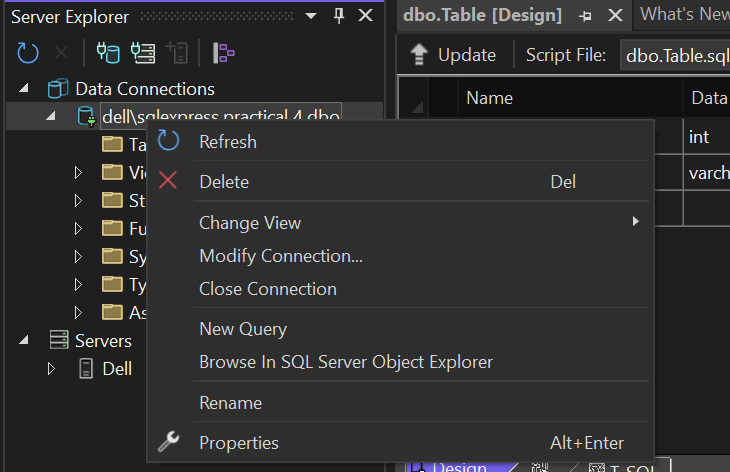
1. Enter the Server name and choose the authentication method (Windows Authentication or SQL Server Authentication) >> Click "Connect."



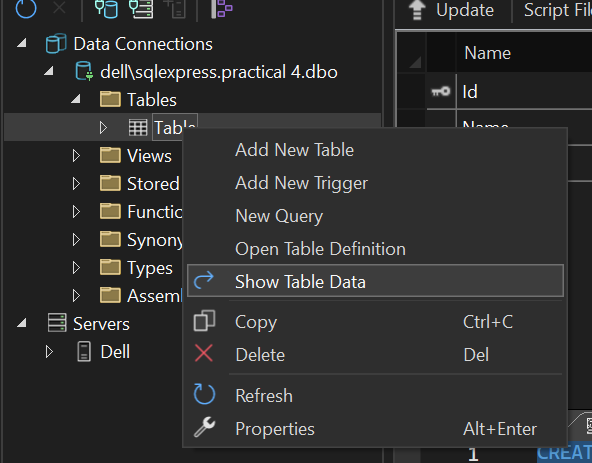
**Step 3: Create a New Table**

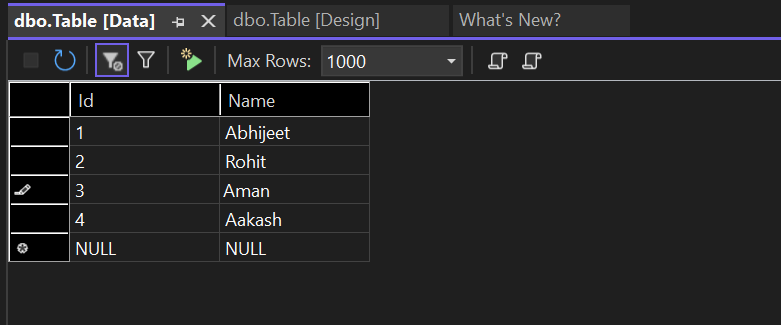




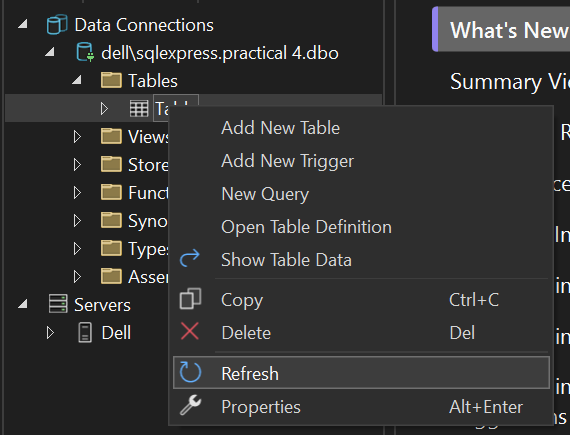


**Click on Show Table Data to view the data in the table.**

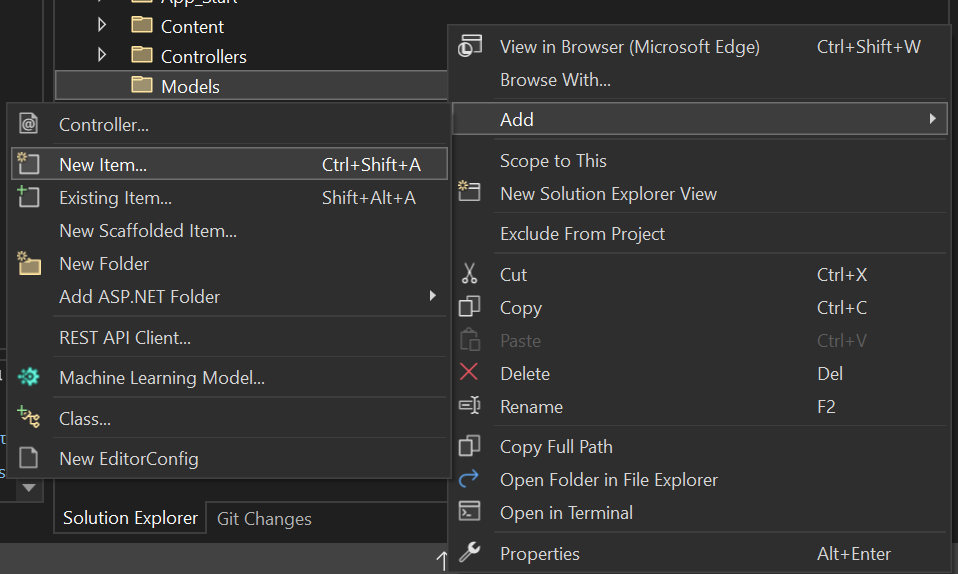


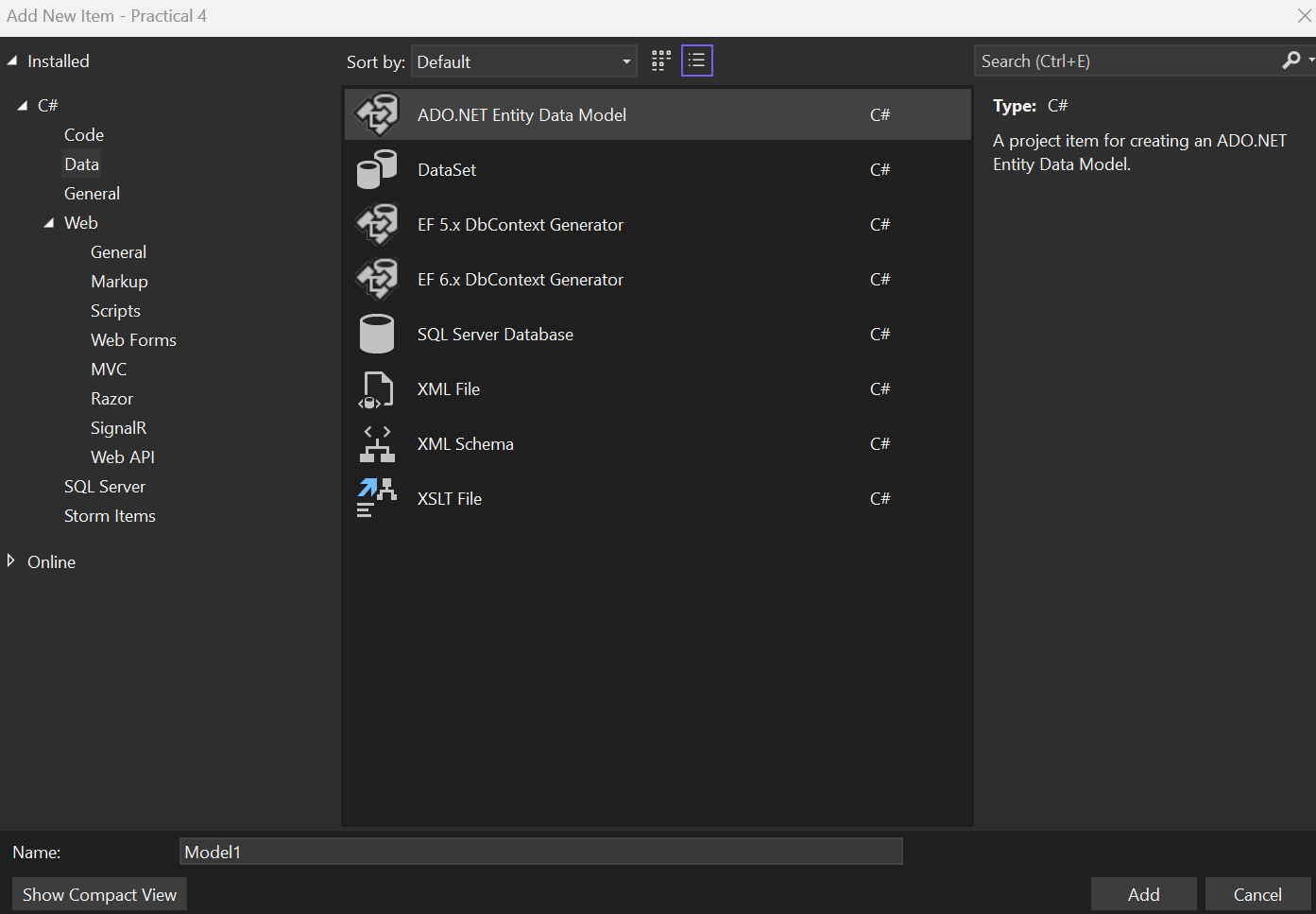


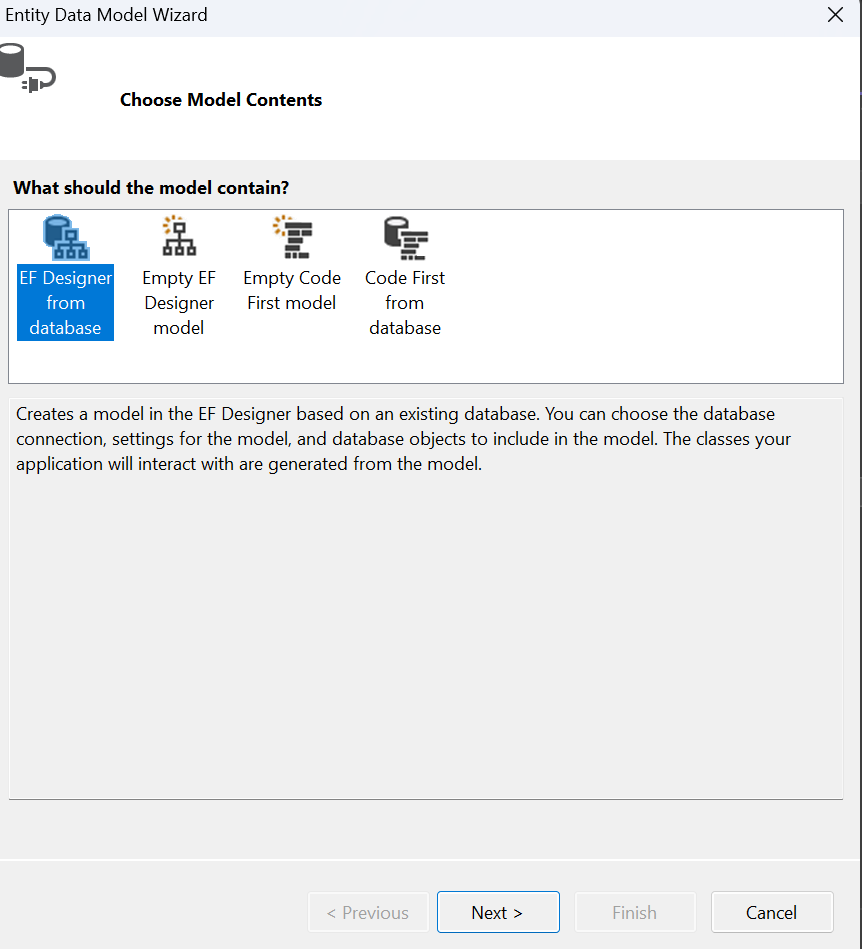
**Click on Refresh Option**

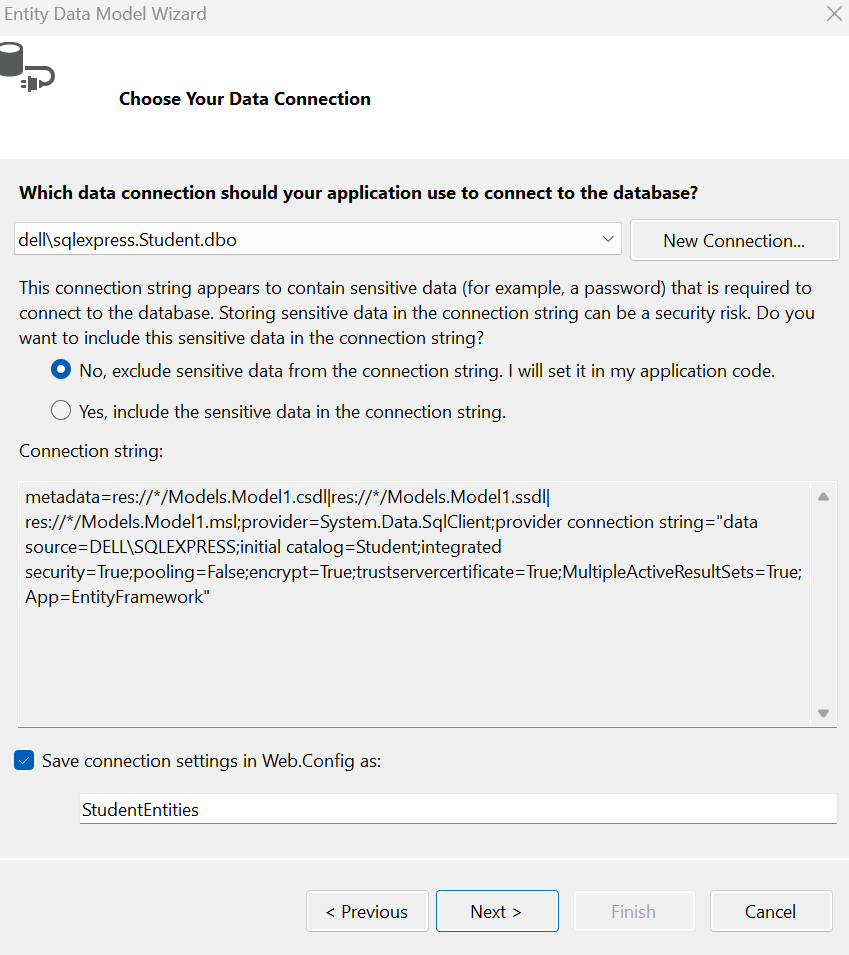


**Click on Models >> Add >> New Item**

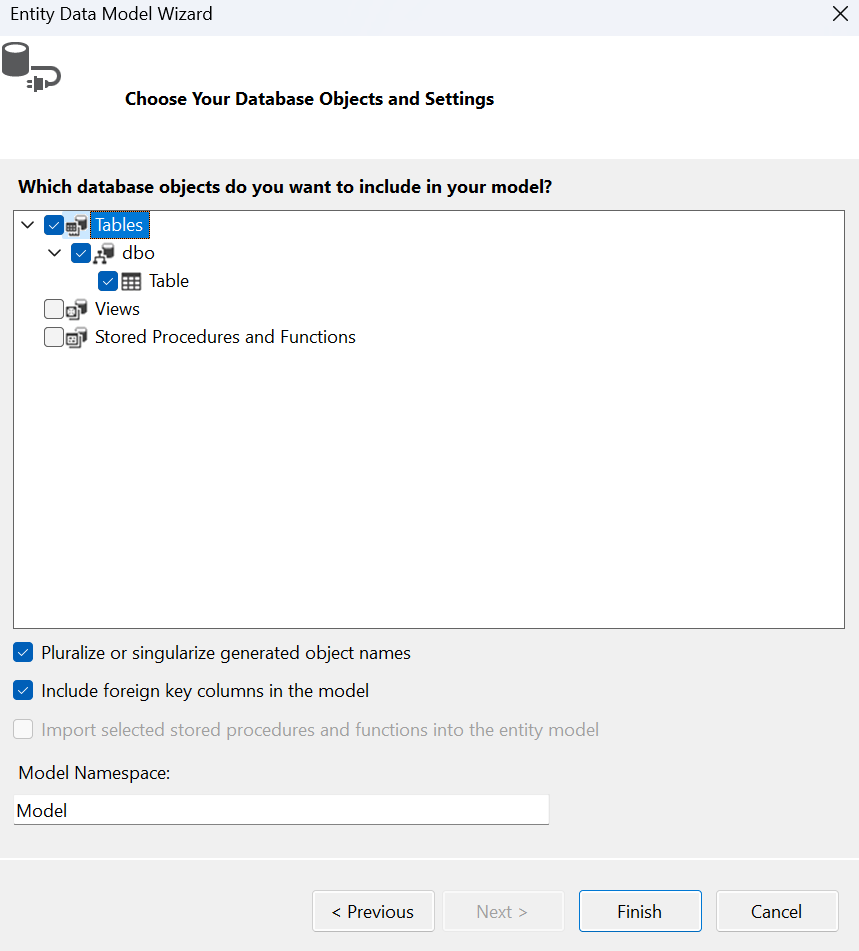


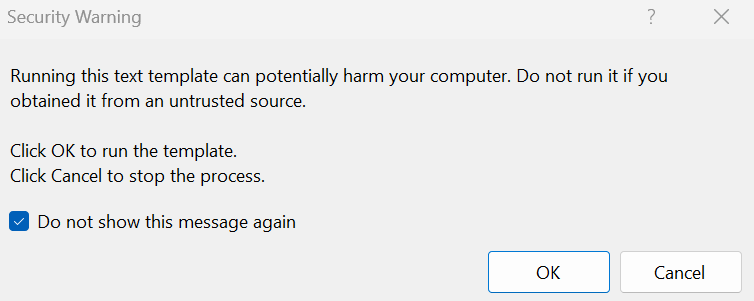


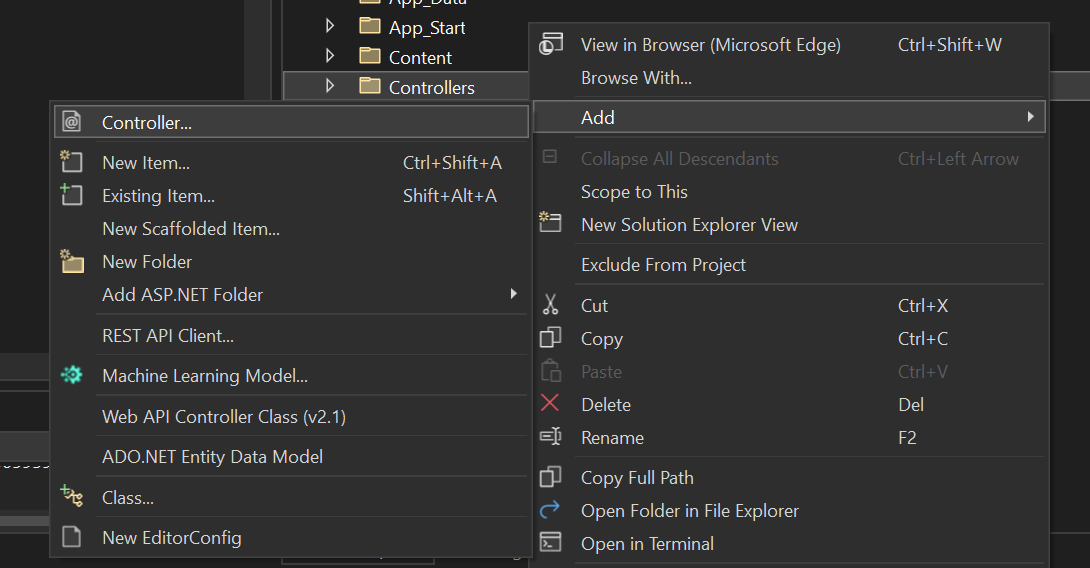


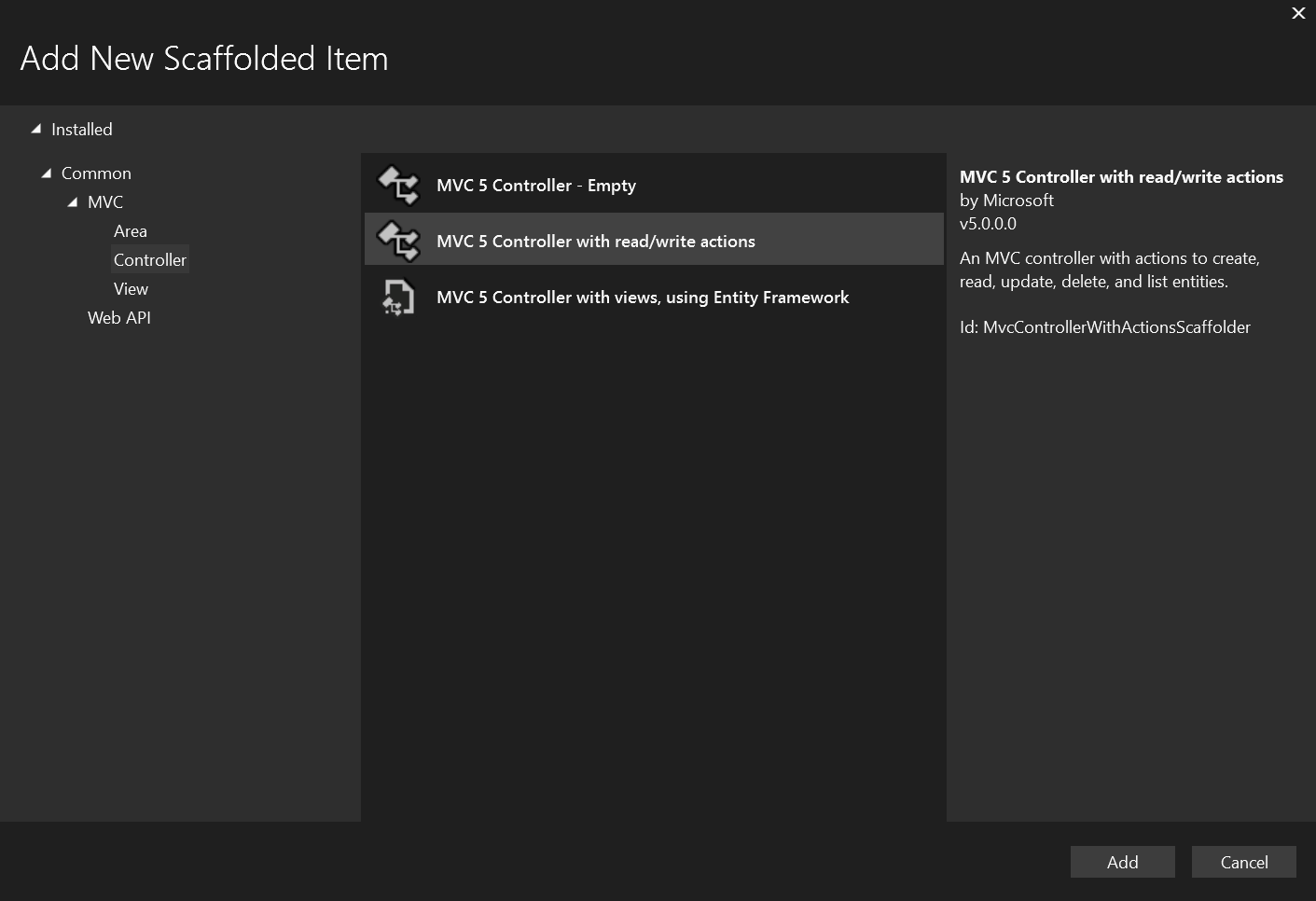


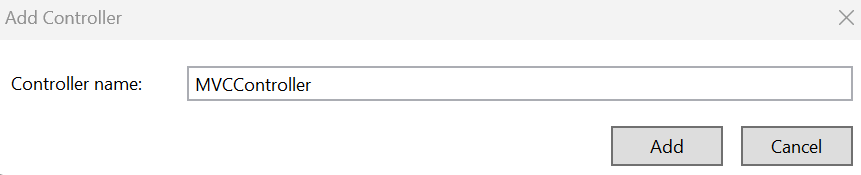


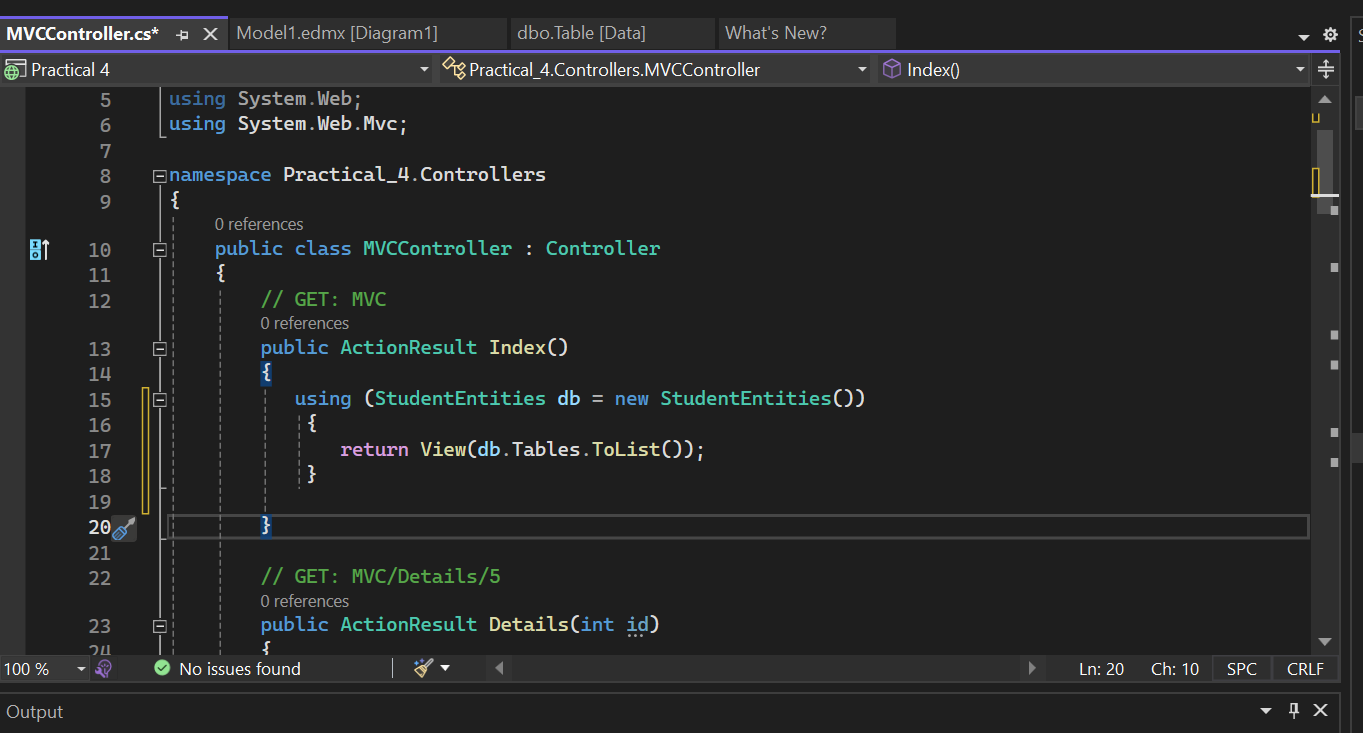


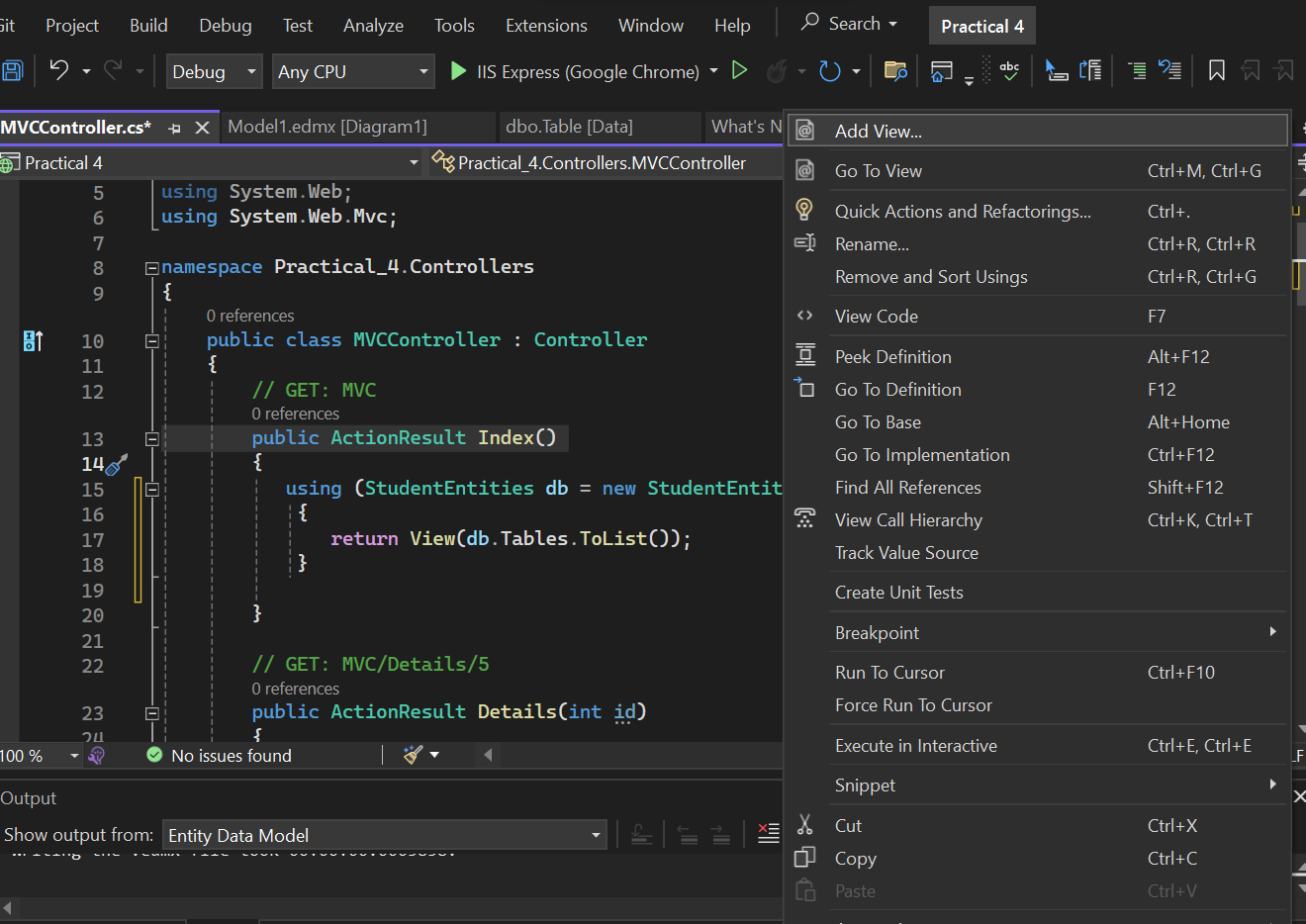


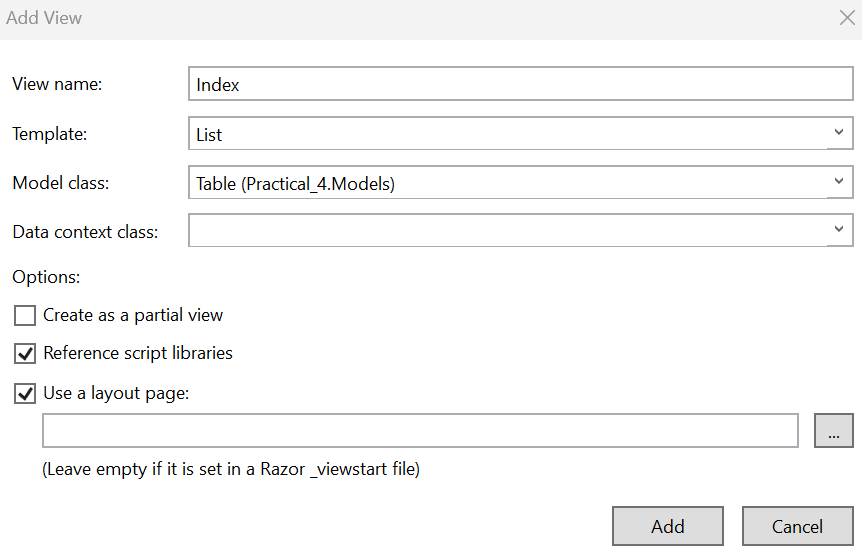


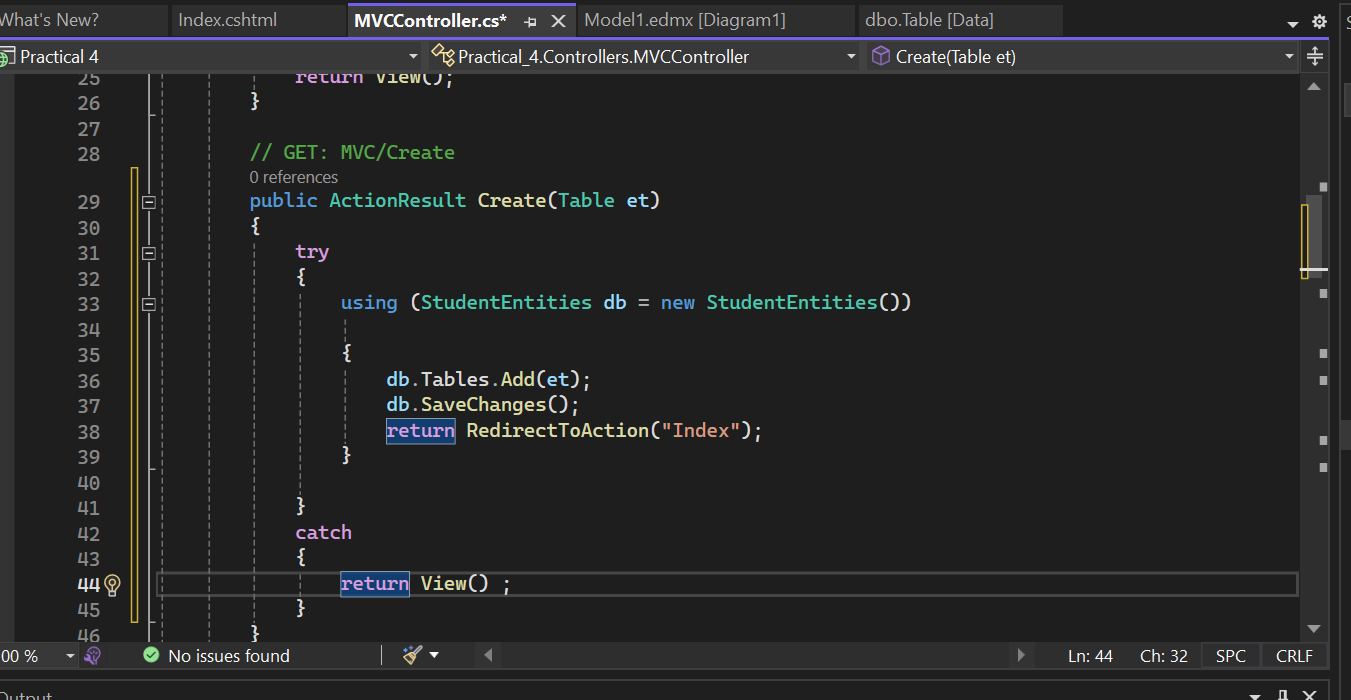


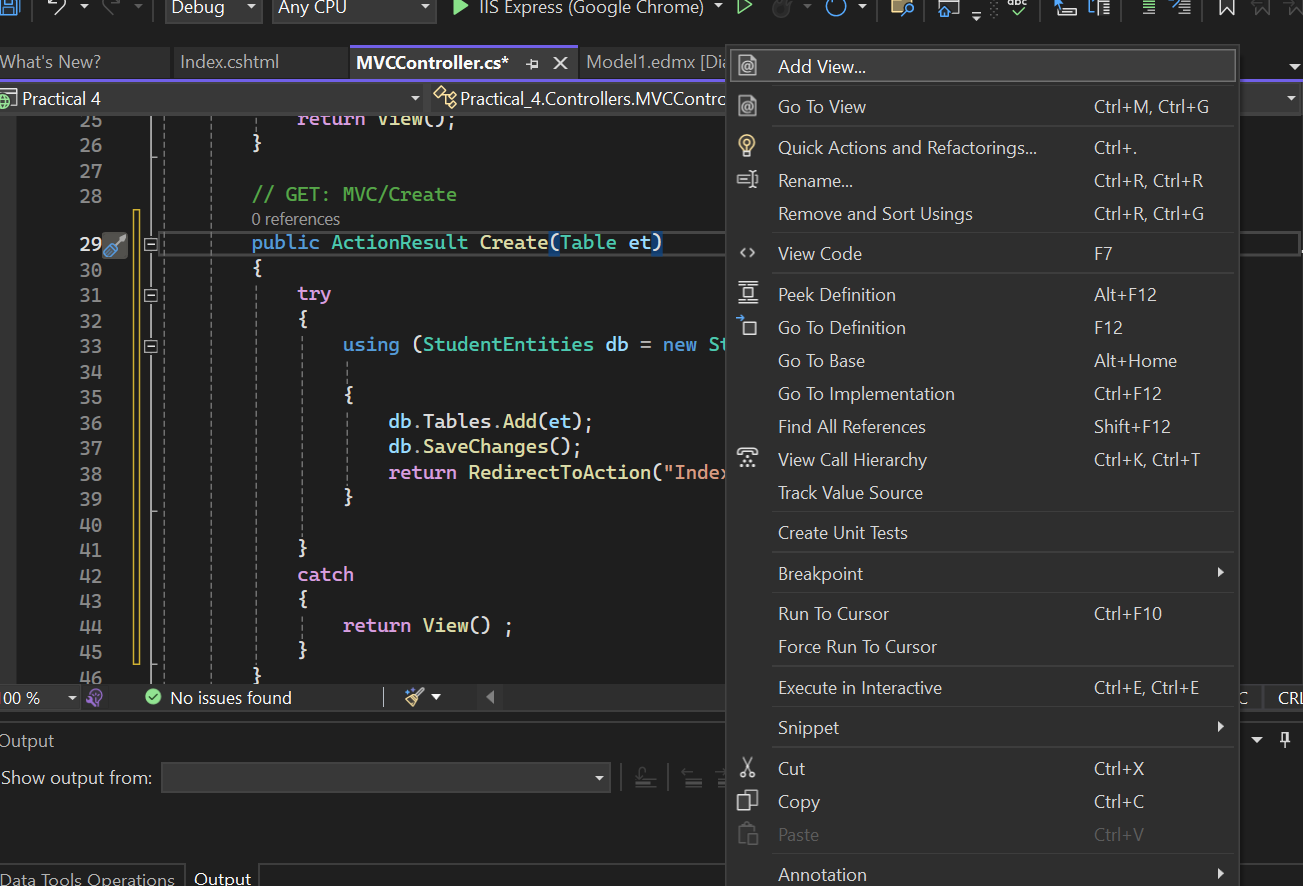


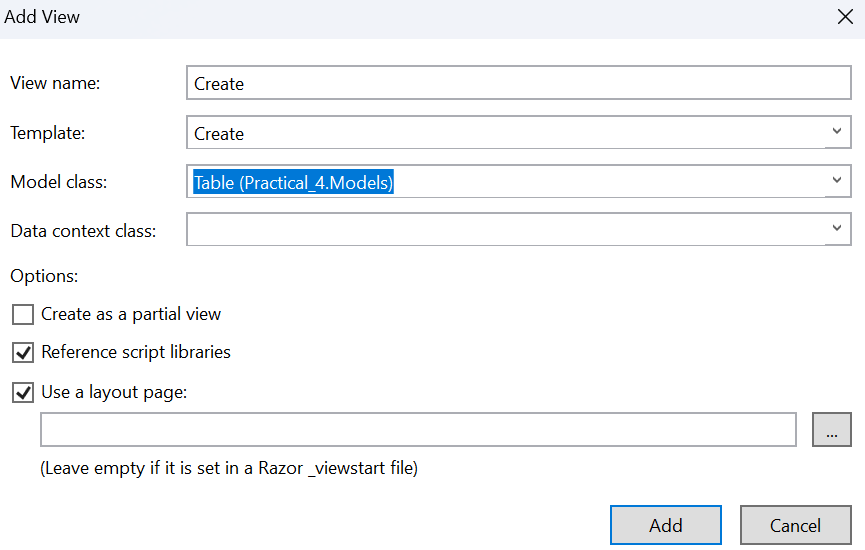


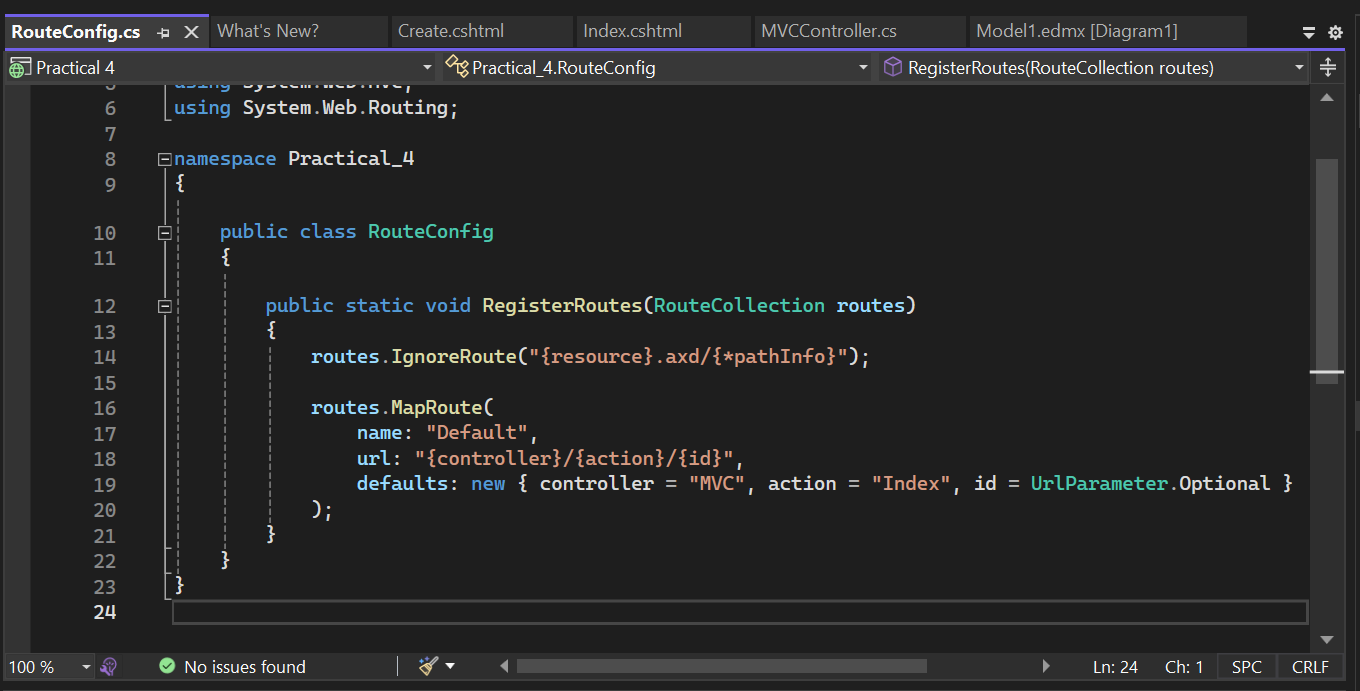




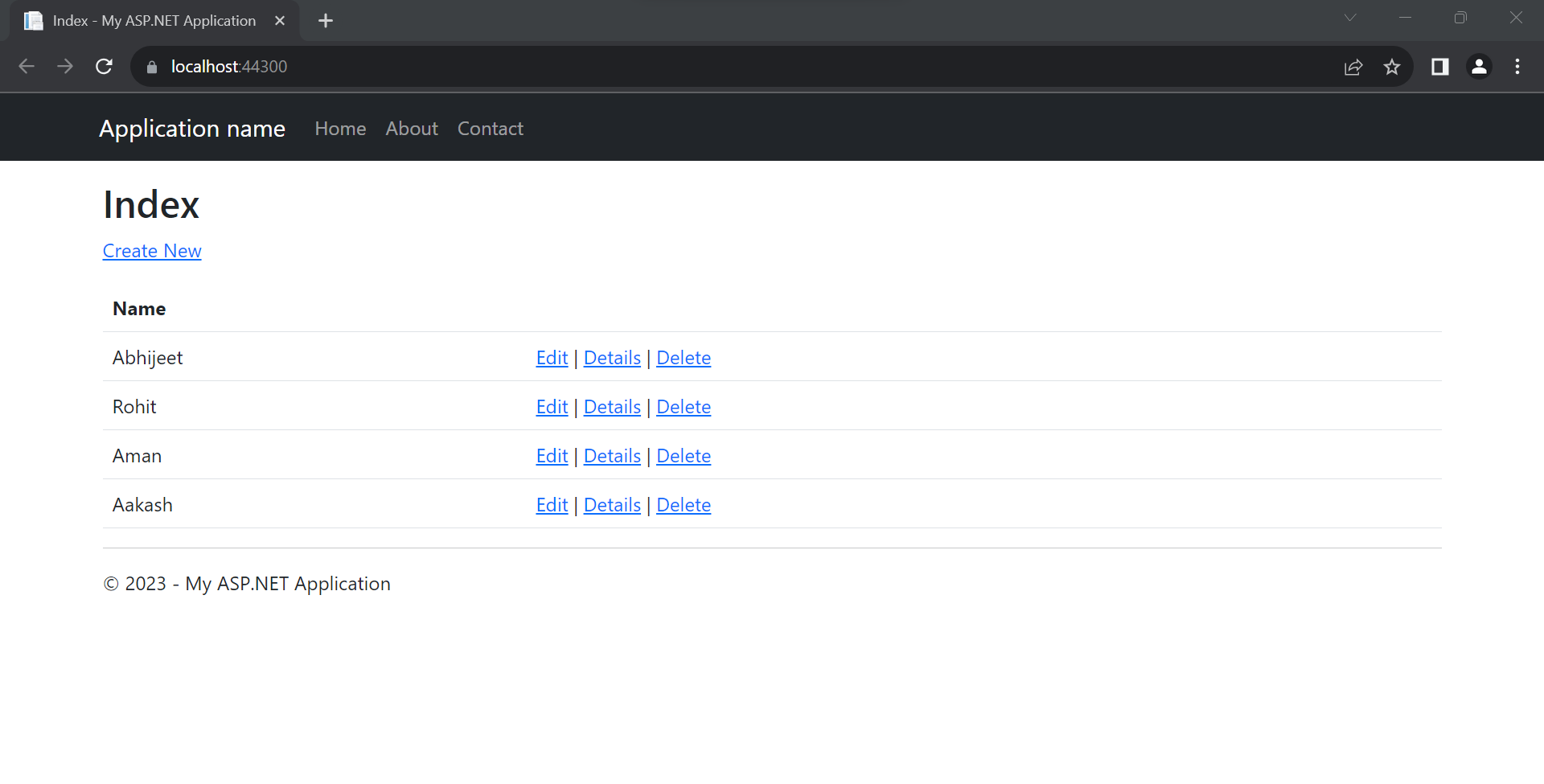








**Output**



Practical No 5

**Aim: Create Service Endpoint.**

1. A Server Endpoint is a physical address available on a network that allows clients to access one or more services provided by a server. Server endpoint is specified by its URL string.
2. Besides standard URLs e.g. with HTTP or HTTPS schemes, OPC Unified Architecture uses its own schemes, such as “opc.tcp”.

**Identification**

It is identified by a URI or URL, such as http://example.com/soap-service-endpoint.

**Communication Entry Point**

Clients use the server endpoint to communicate with the SOAP service.

**Responsibilities**

1. Processes incoming SOAP requests.
2. Hosts various web service operations.
3. Provides a WSDL document describing service functionalities.

**WSDL (Web Services Description Language)**

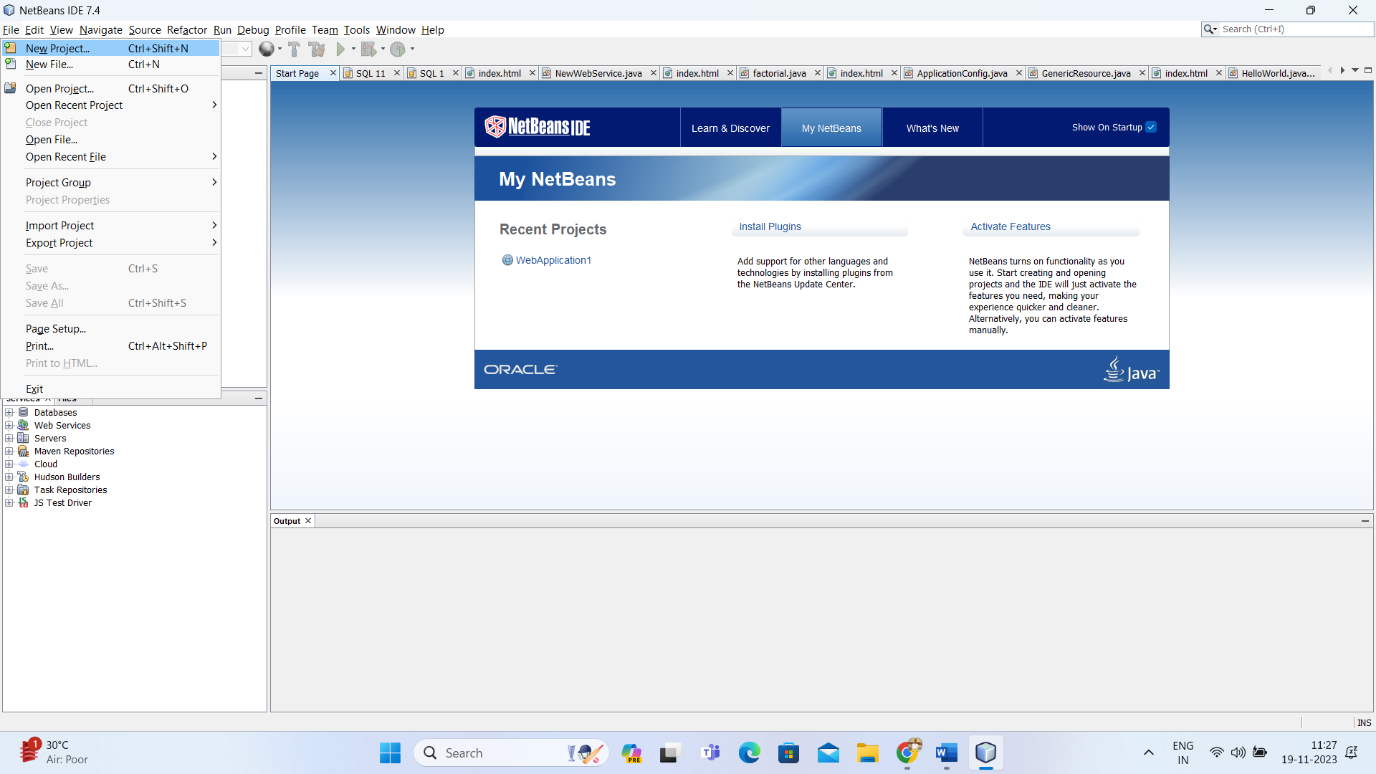
1. Typically includes a WSDL document that outlines the SOAP service's operations and communication details.
2. Implementation: In Java, for example, the server endpoint is often defined using technologies like JAX-WS or Apache CXF.

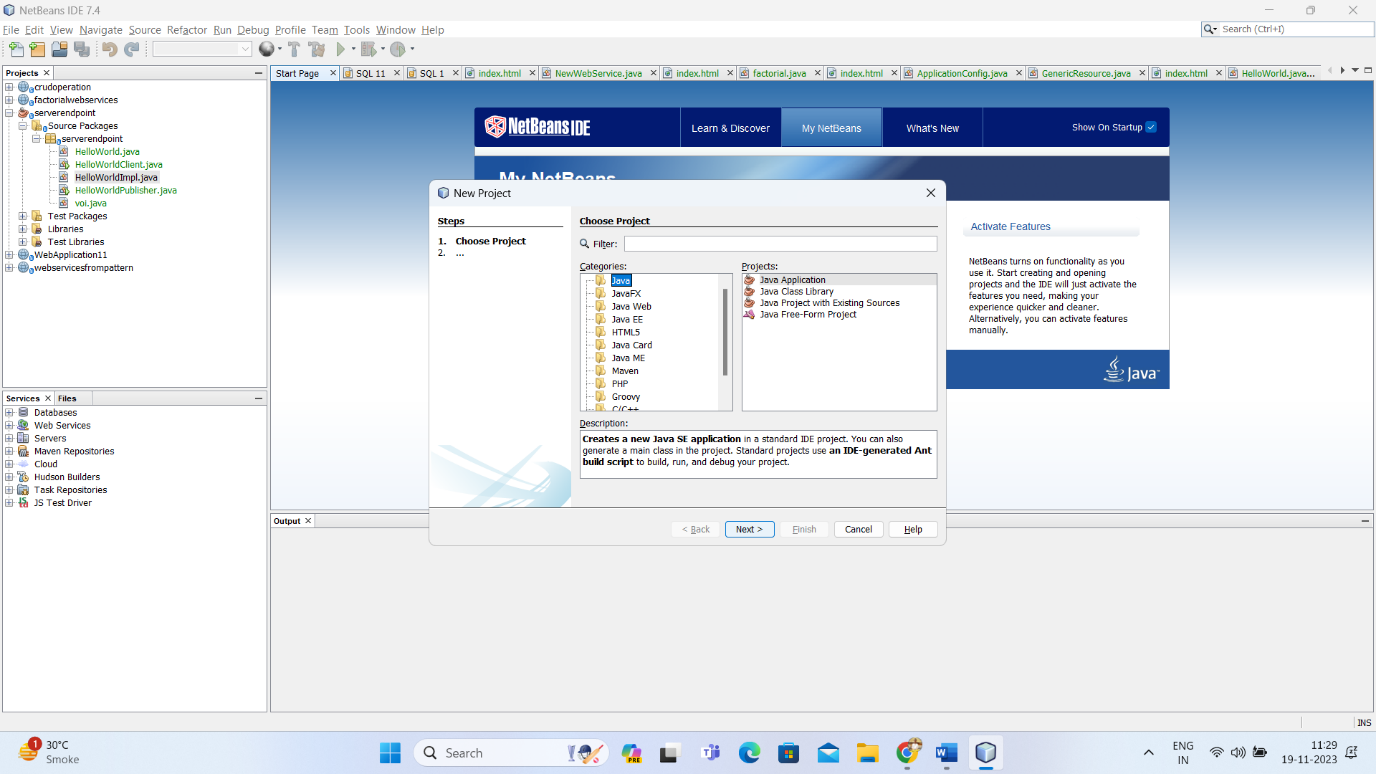
**Functionality Description**

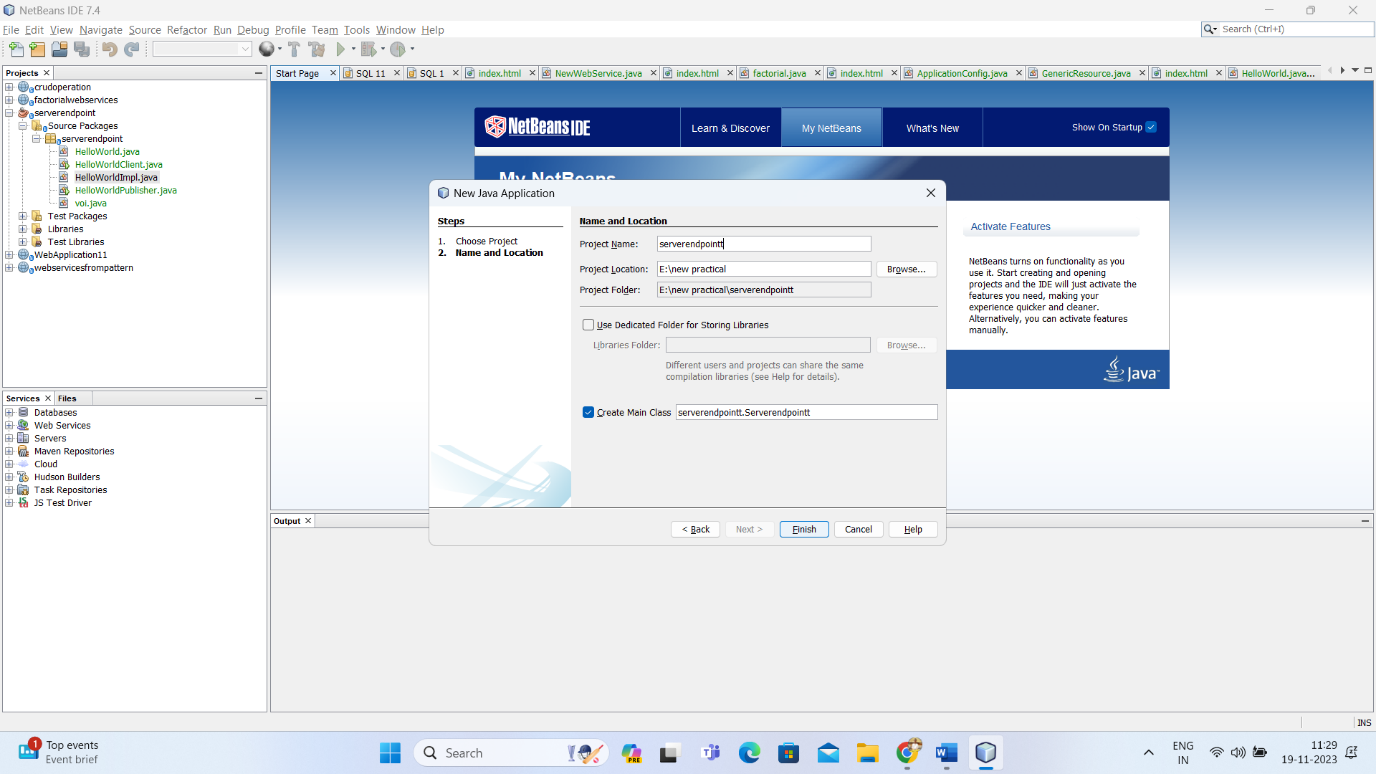
The server endpoint class contains methods annotated with @WebMethod or similar annotations, representing SOAP operations.

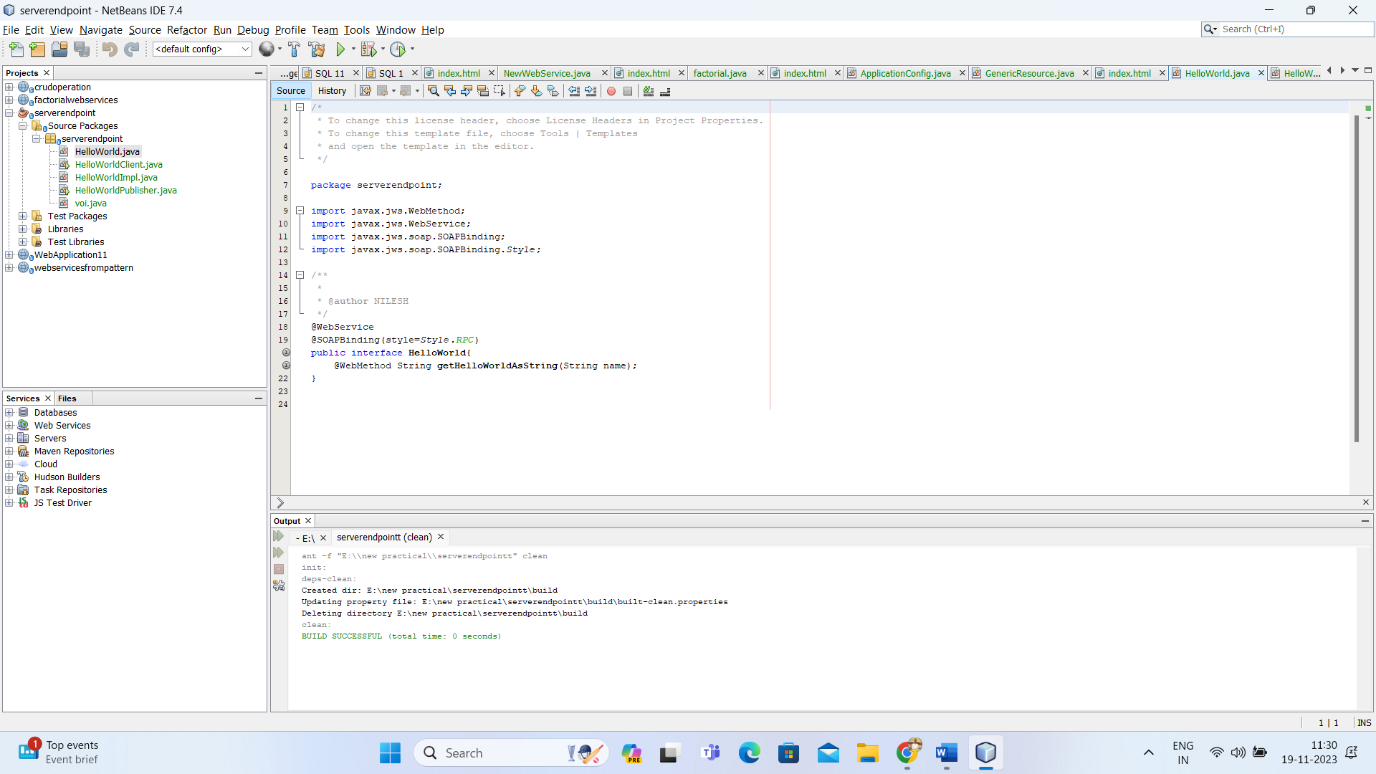
**Gateway**

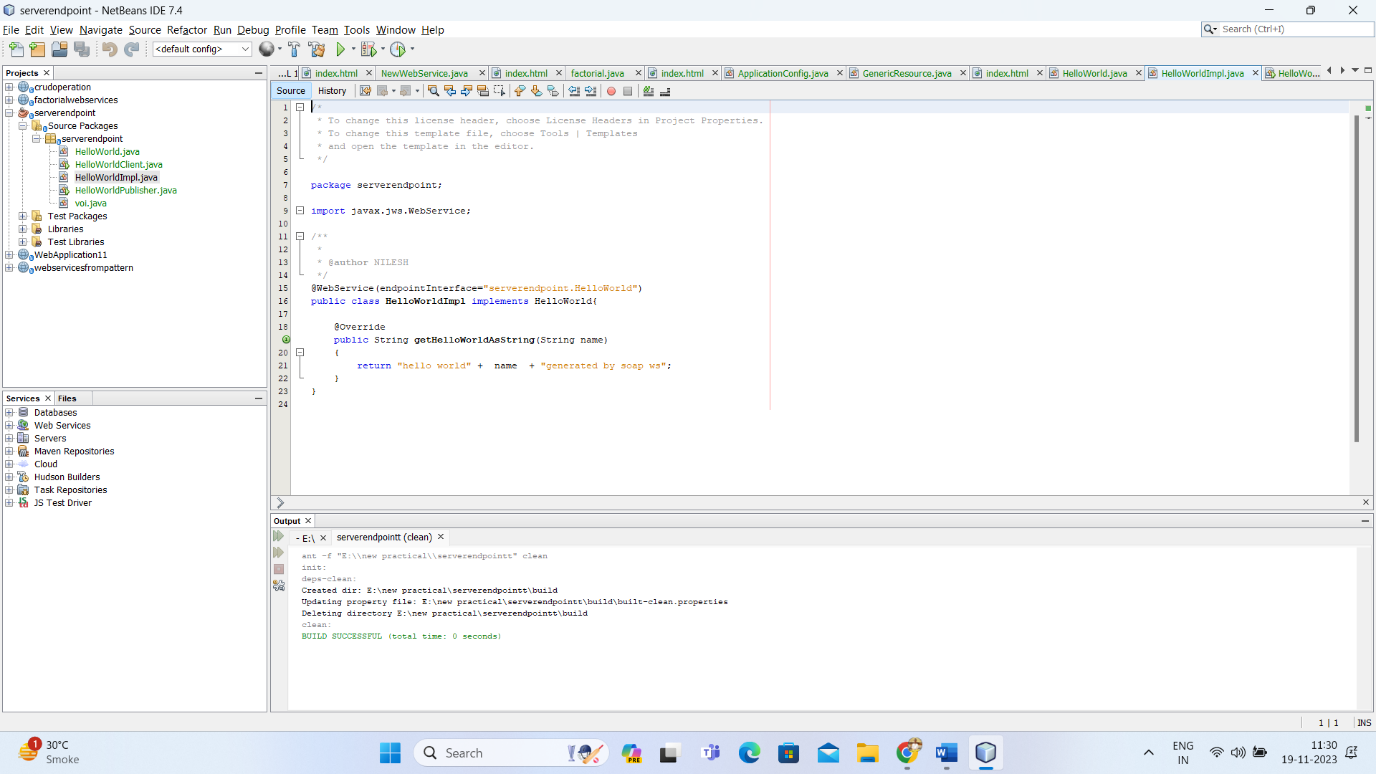
Acts as a gateway for clients to interact with the SOAP service.

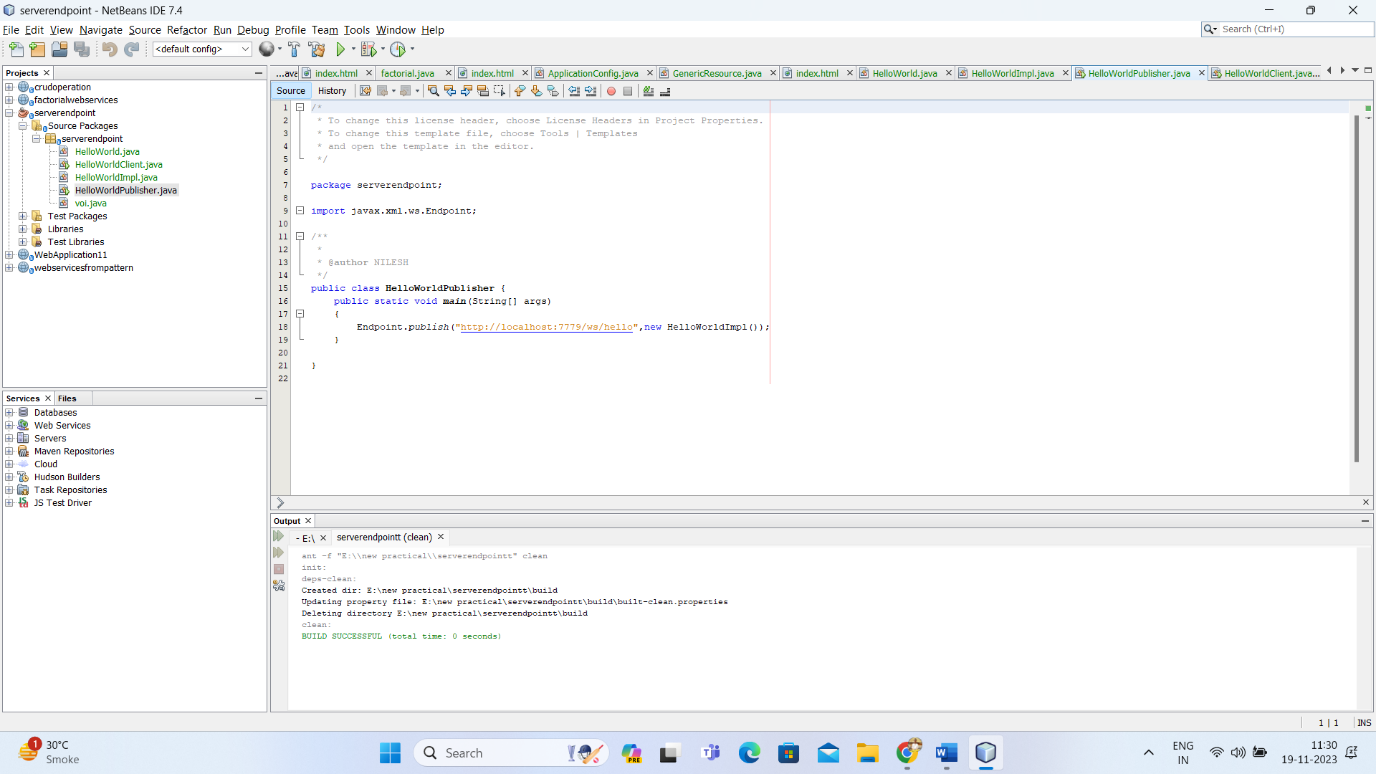


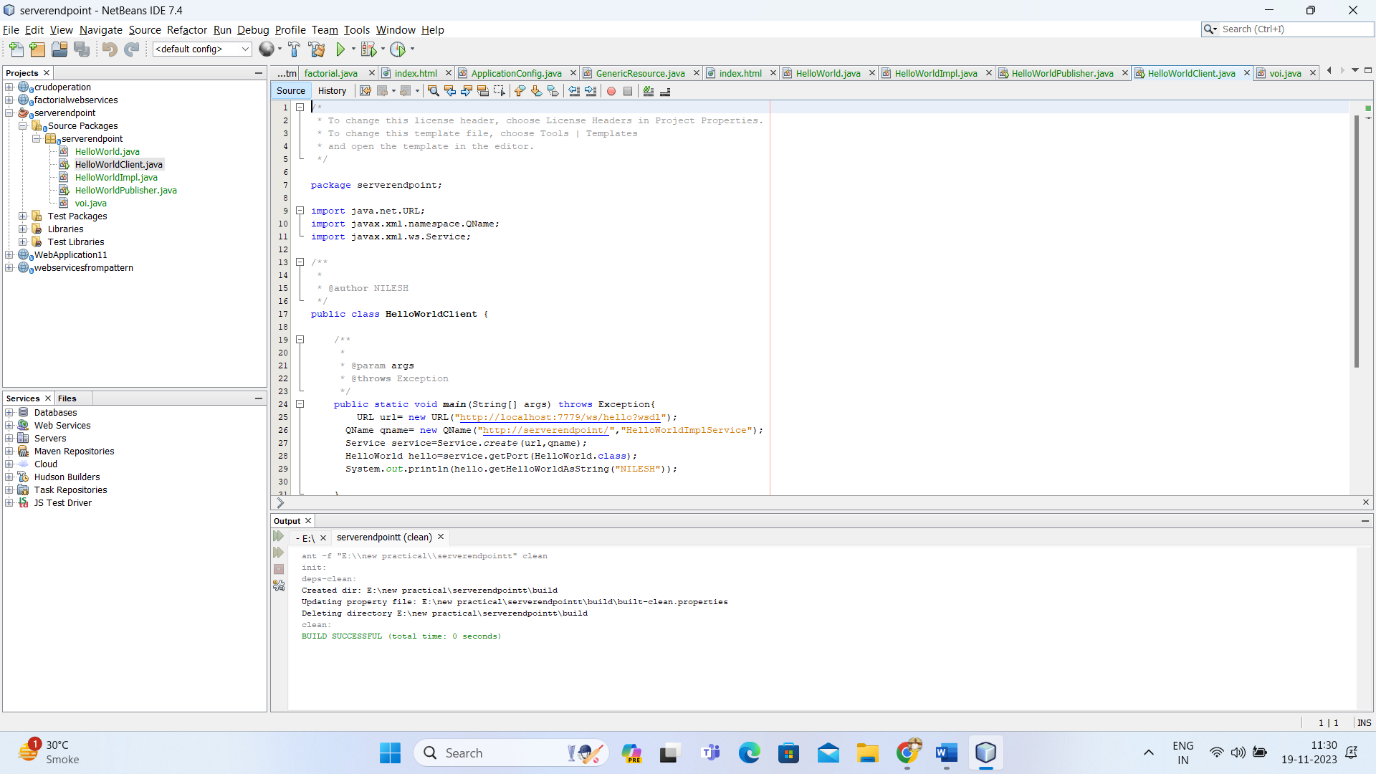


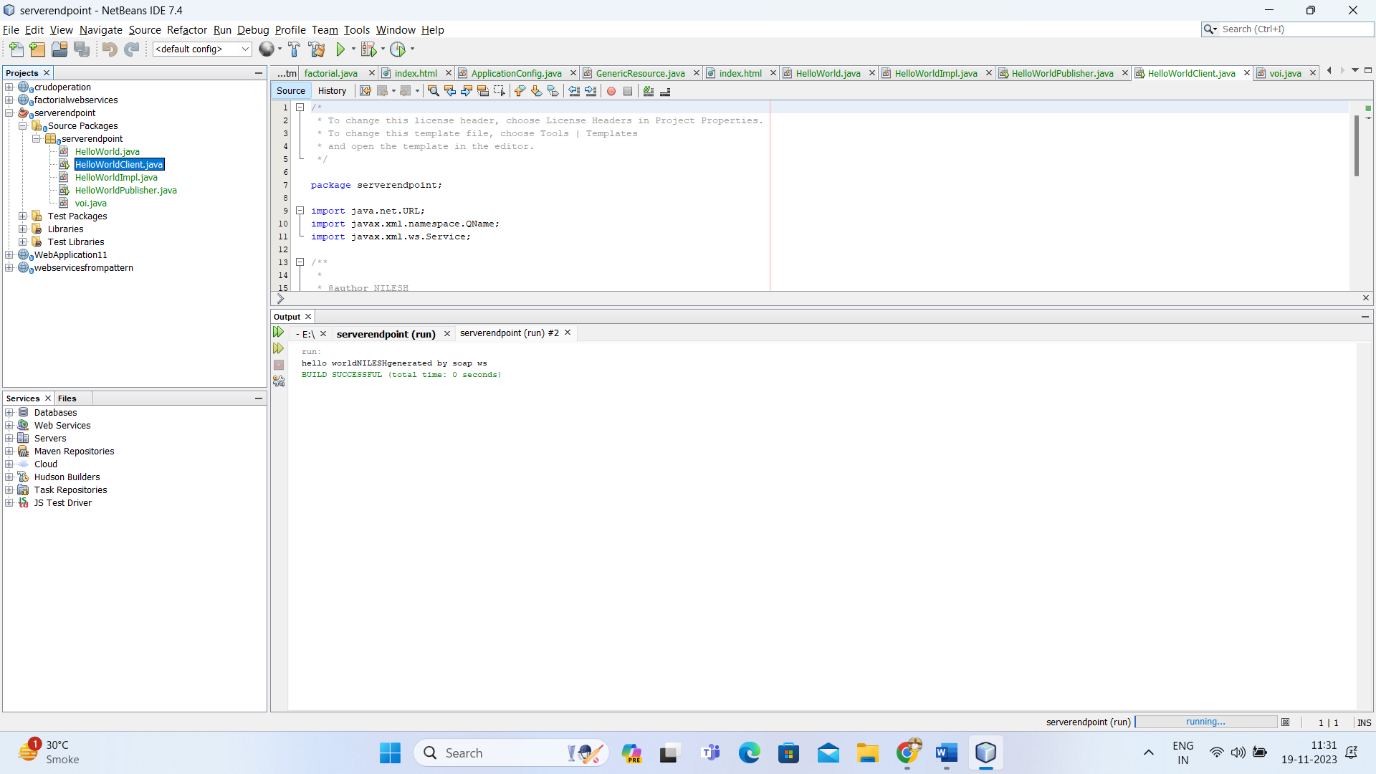












Practical No 6

**Aim: RESTful Services using Web API.**

**Web API**

1. A web API is an application programming interface (API) for either a web server or a web browser.
2. As a web development concept, it can be related to a web application's client side (including any web frameworks being used).
3. A server-side web API consists of one or more publicly exposed endpoints to a defined request–response message system, typically expressed in JSON or XML by means of an HTTP-based web server.
4. A server API (SAPI) is not considered a server-side web API, unless it is publicly accessible by a remote web application.

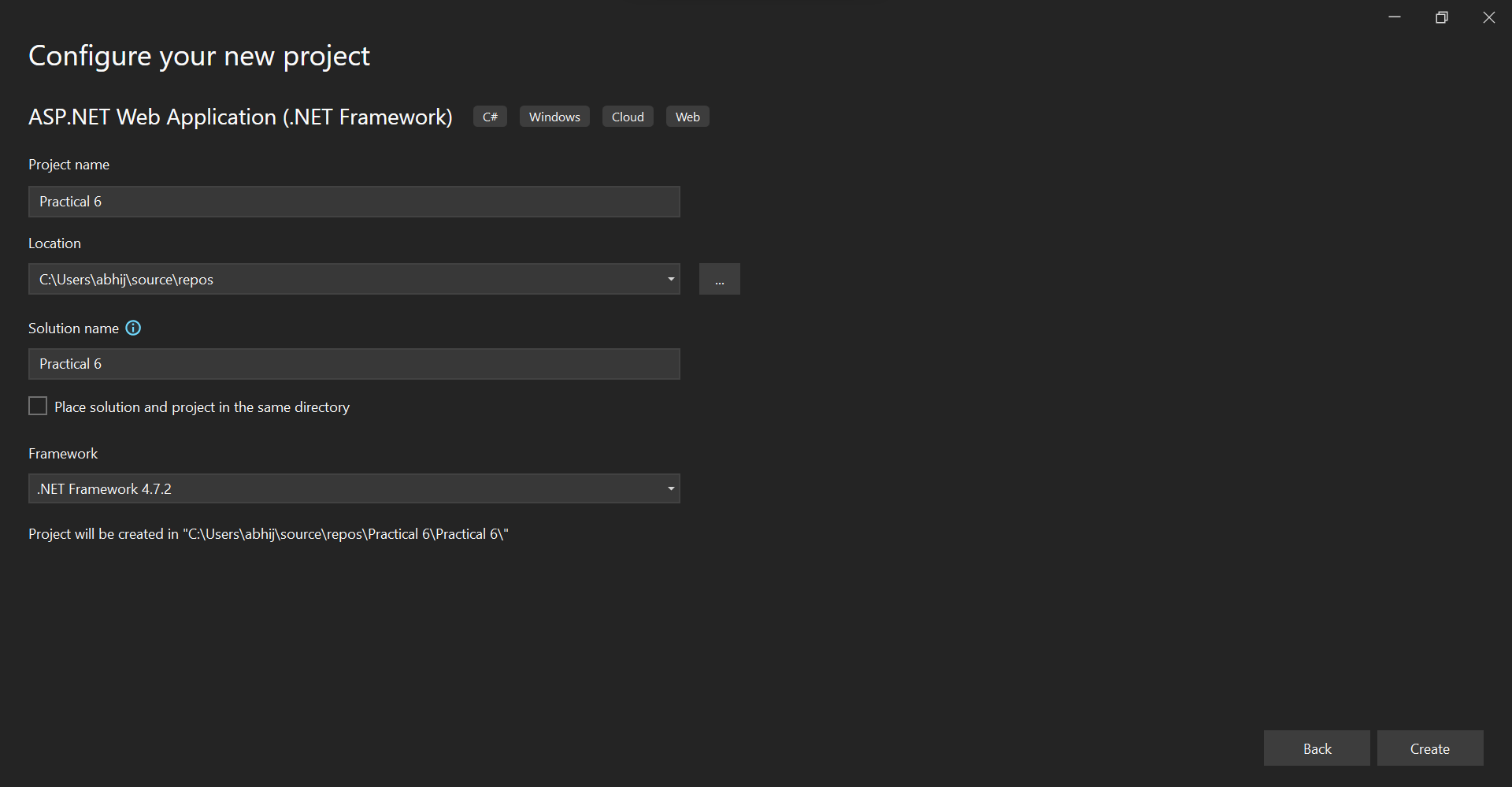
**Client side**

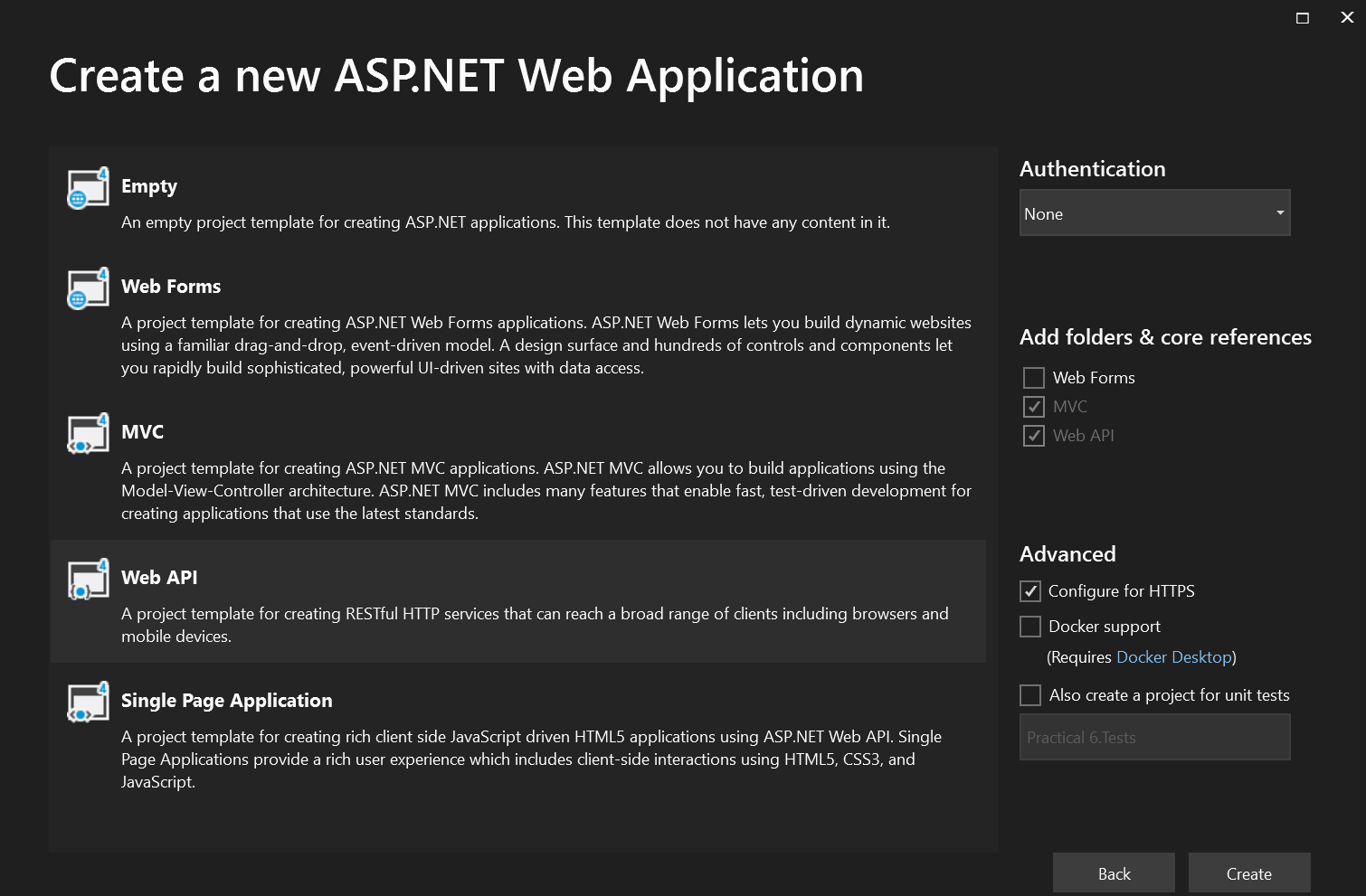
1. A client-side web API is a programmatic interface to extend functionality within a web browser or other HTTP client.
2. Originally these were most commonly in the form of native plug-in browser extensions however most newer ones target standardized JavaScript bindings.
3. The Mozilla Foundation created their WebAPI specification which is designed to help replace native mobile applications with HTML5 applications.
4. Google created their Native Client architecture which is designed to help replace insecure native plug-ins with secure native sandboxed extensions and applications.
5. They have also made this portable by employing a modified LLVM AOT compiler.

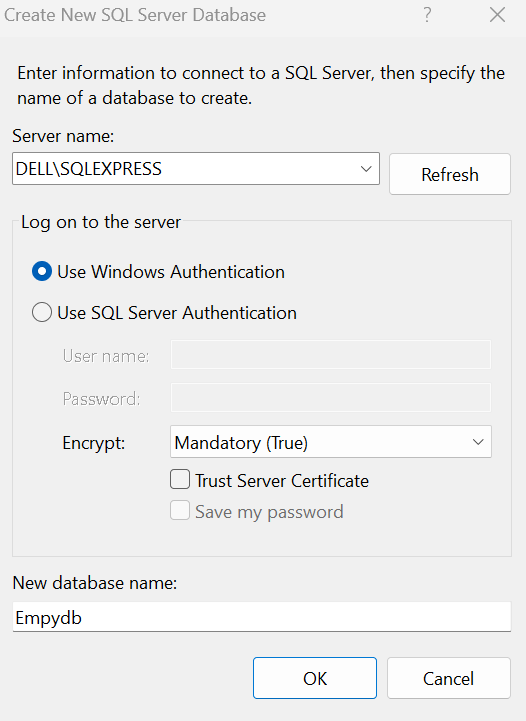
**Server side**

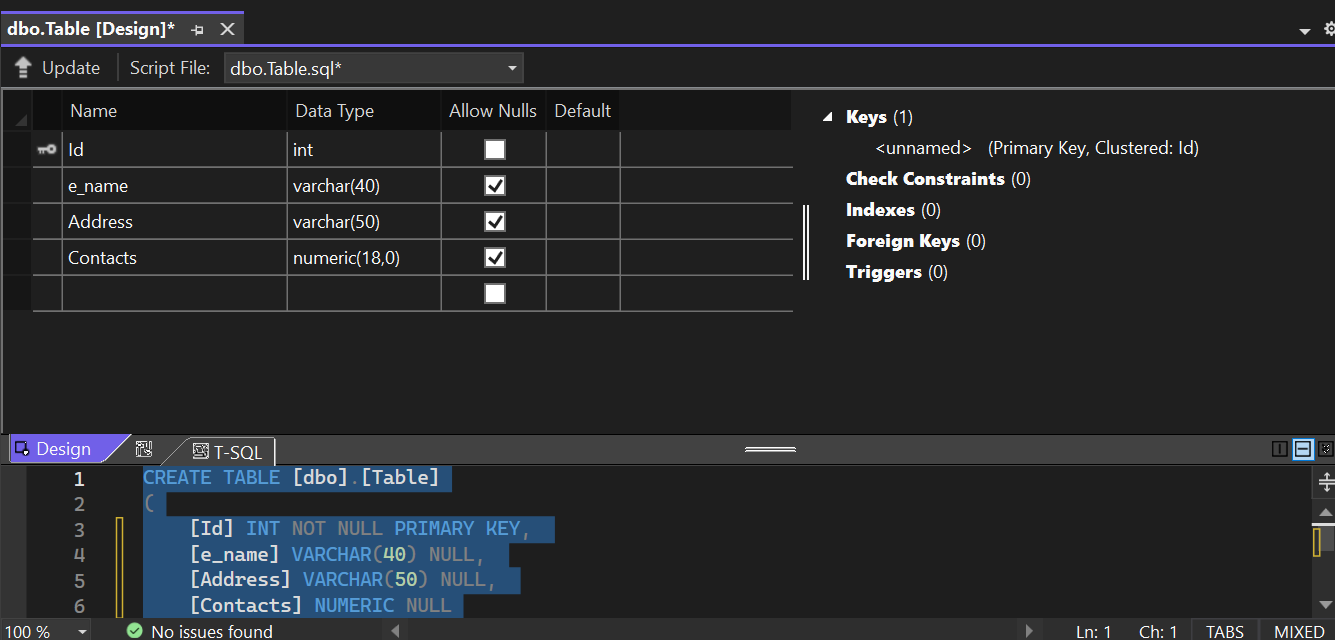
1. A server-side web API consists of one or more publicly exposed endpoints to a defined request–response message system, typically expressed in JSON or XML.
2. The web API is exposed most commonly by means of an HTTP-based web server.
3. Mashups are web applications which combine the use of multiple server-side web APIs.
4. Webhooks are server-side web APIs that take input as a Uniform Resource Identifier (URI) that is designed to be used like a remote named pipe or a type of callback such that the server acts as a client to dereference the provided URI and trigger an event on another server which handles this event thus providing a type of peer-to-peer IPC.

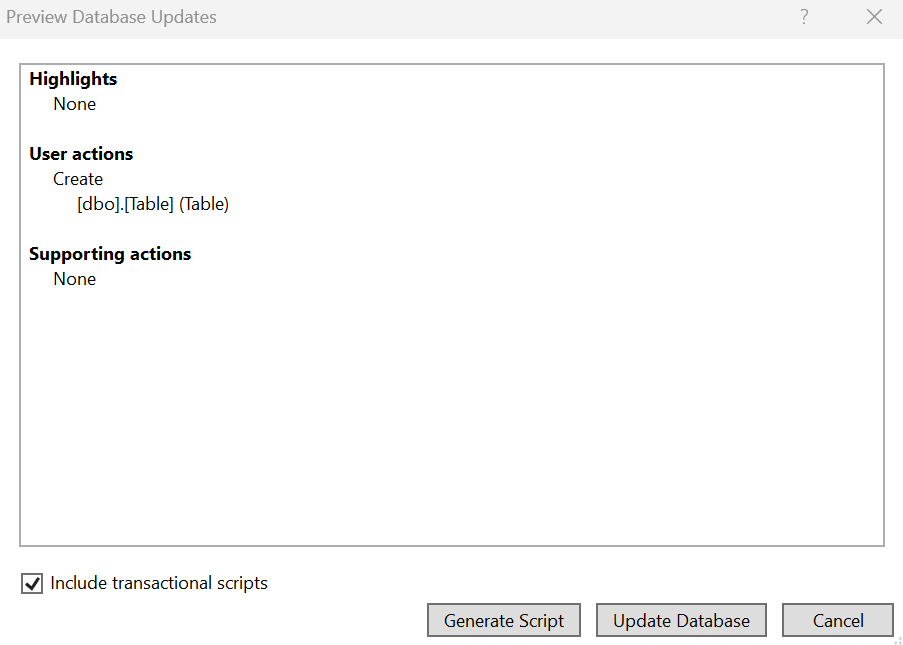
**Create new project >> select c sharp >> select Web API**

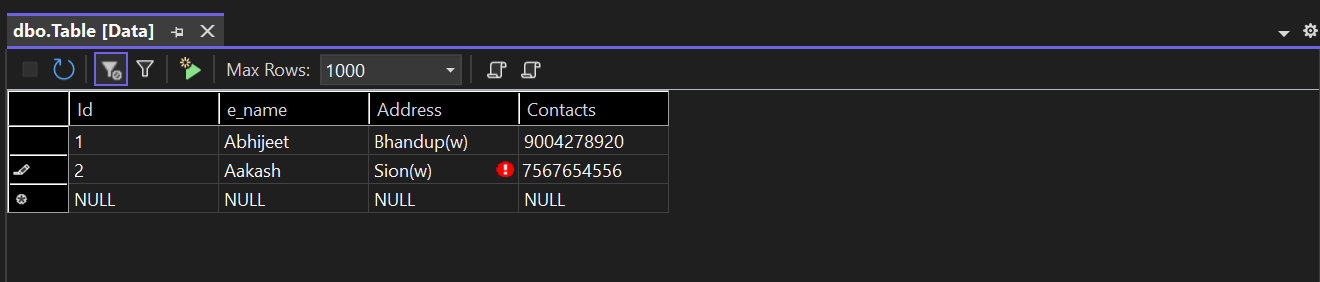


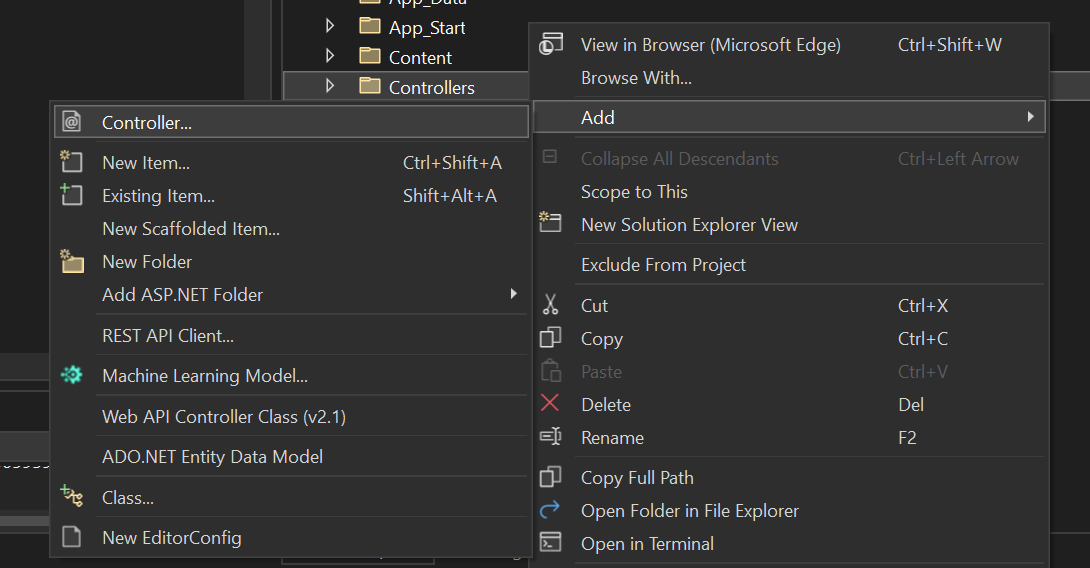




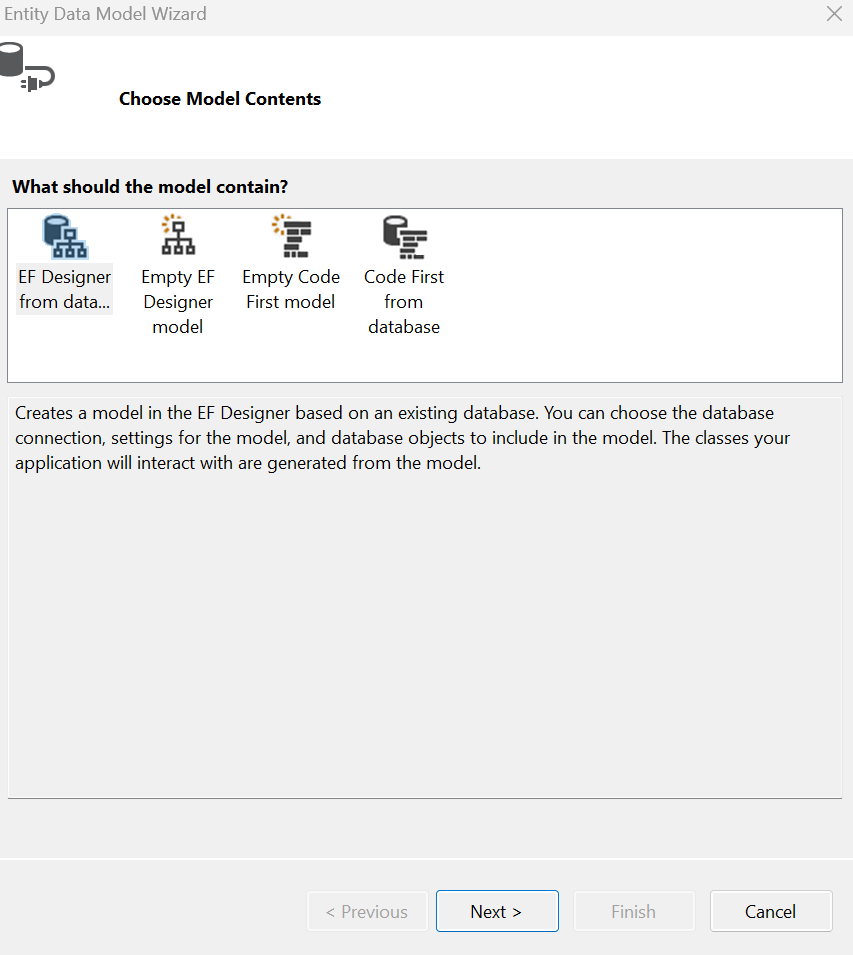


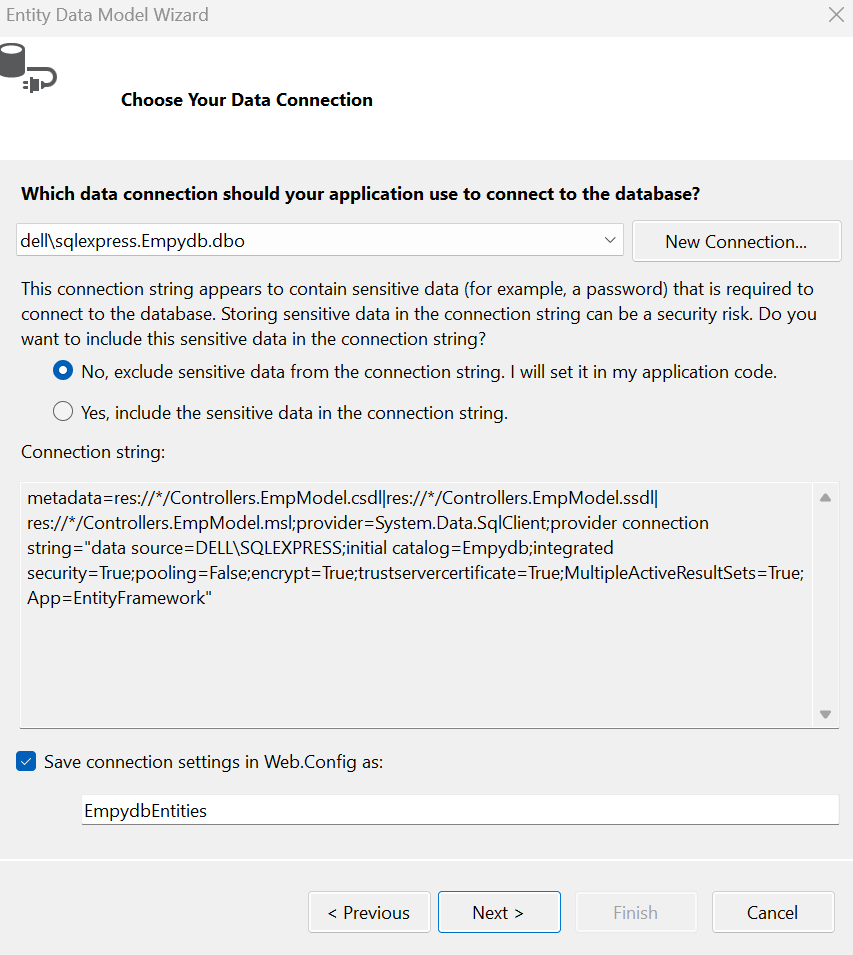


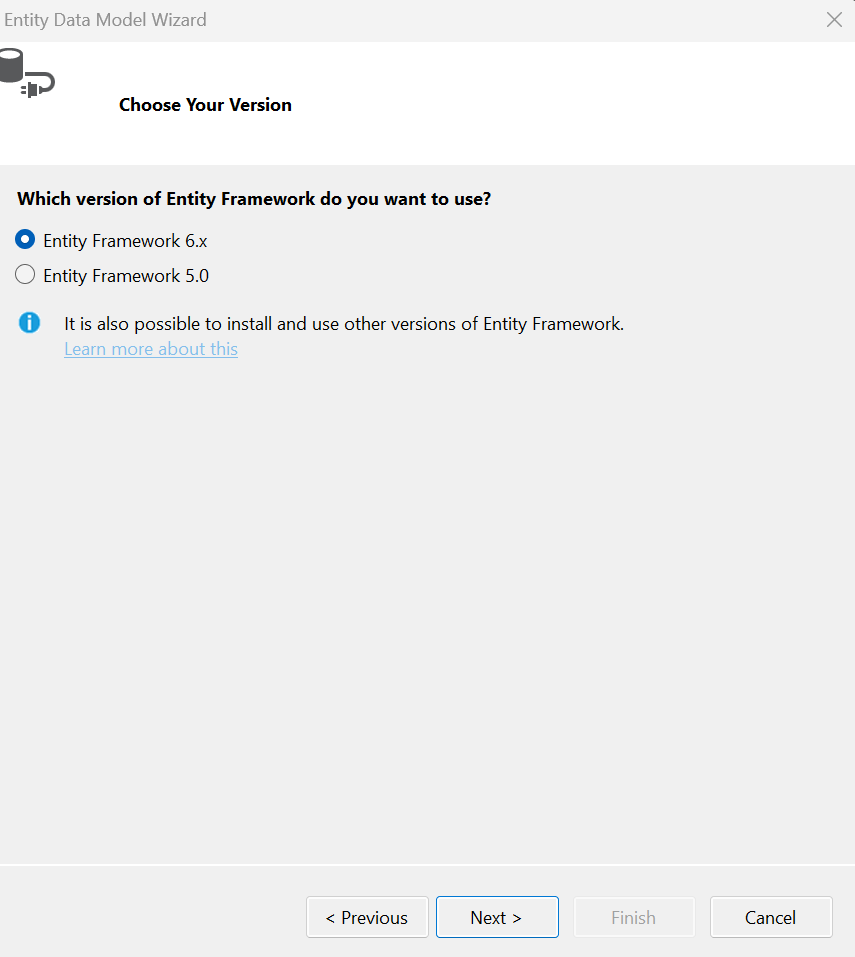


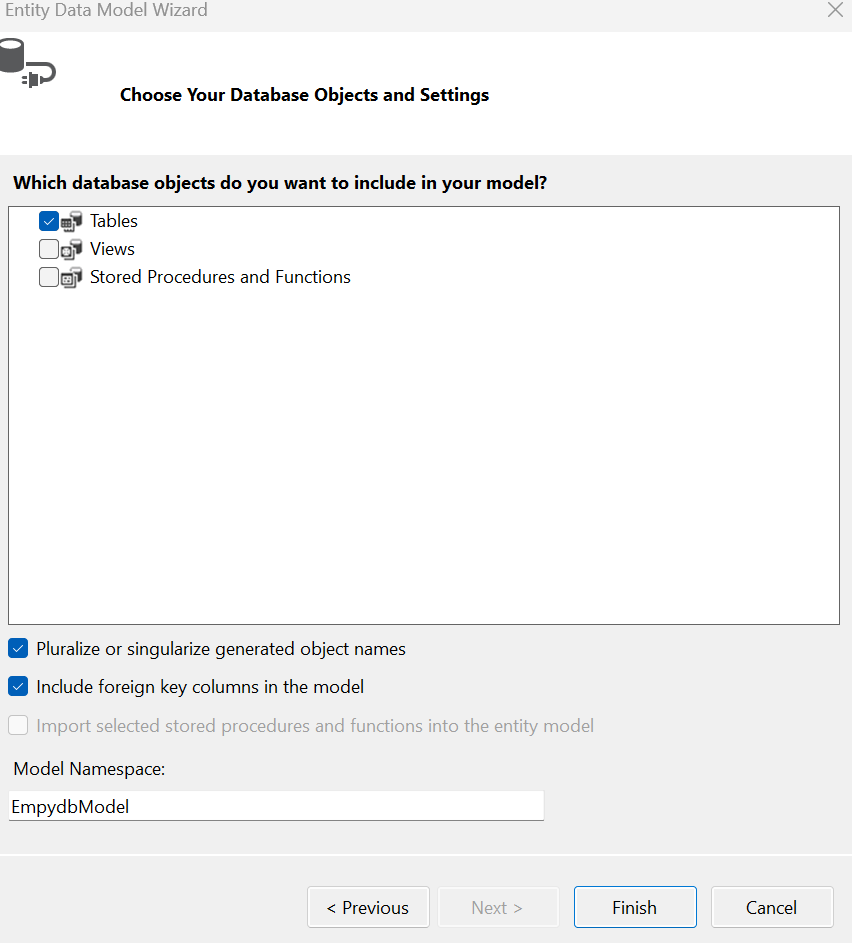


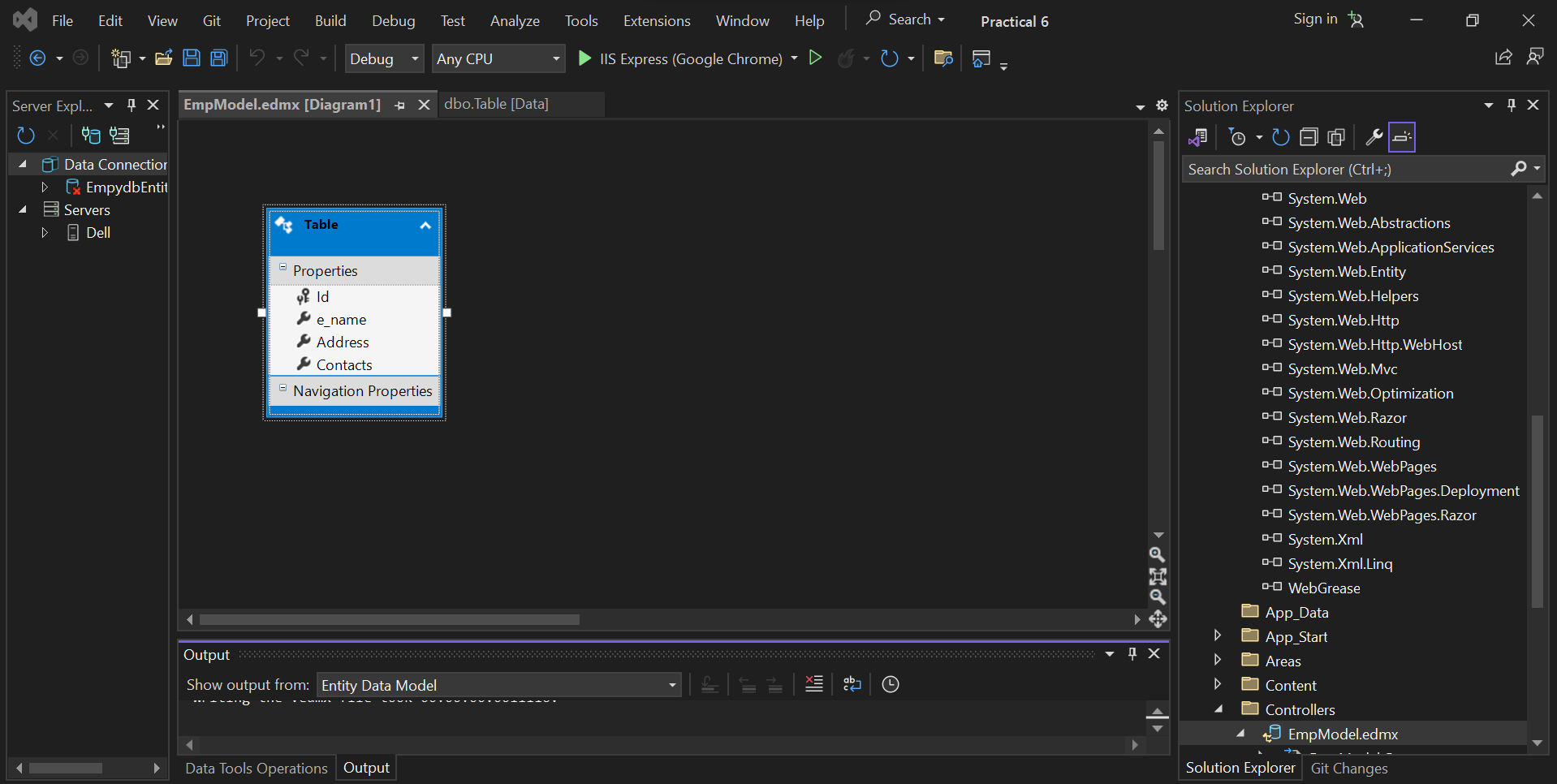


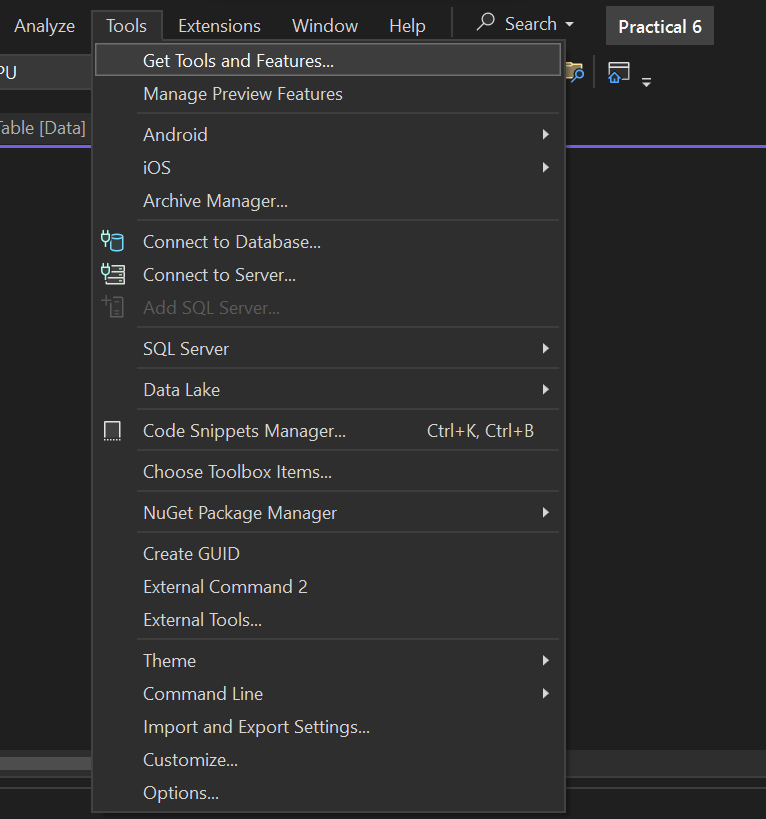


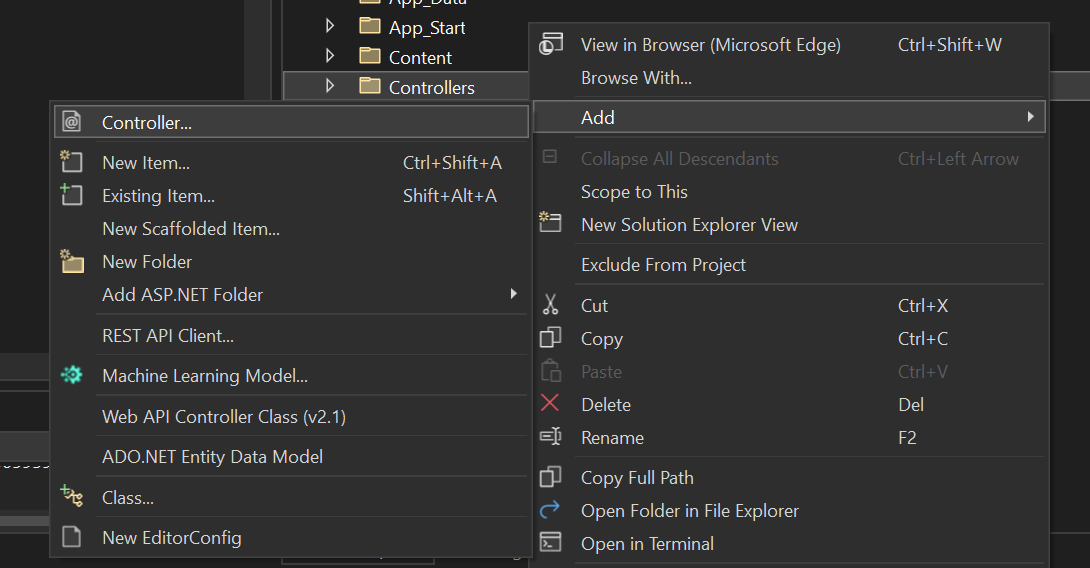


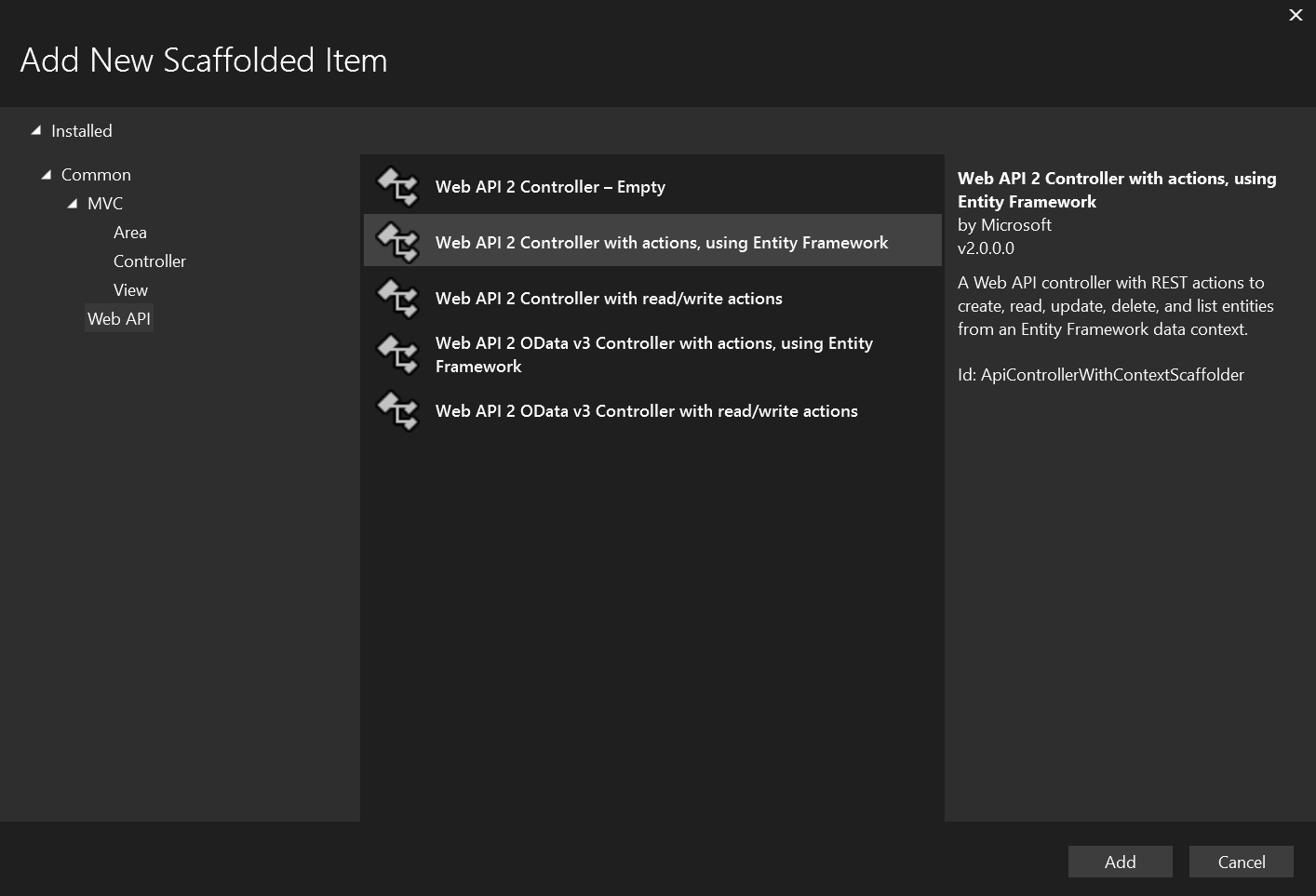


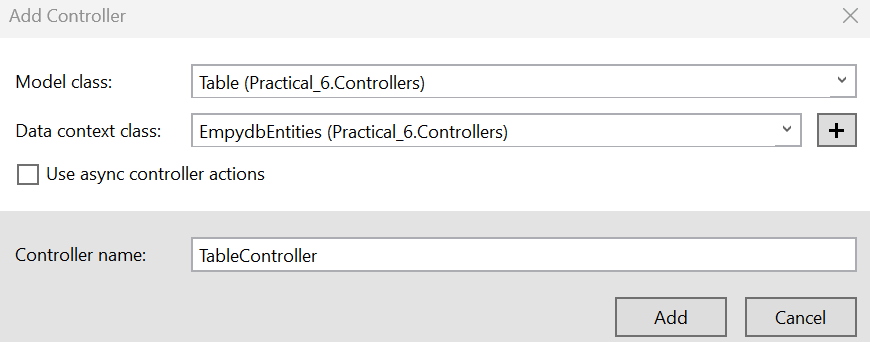




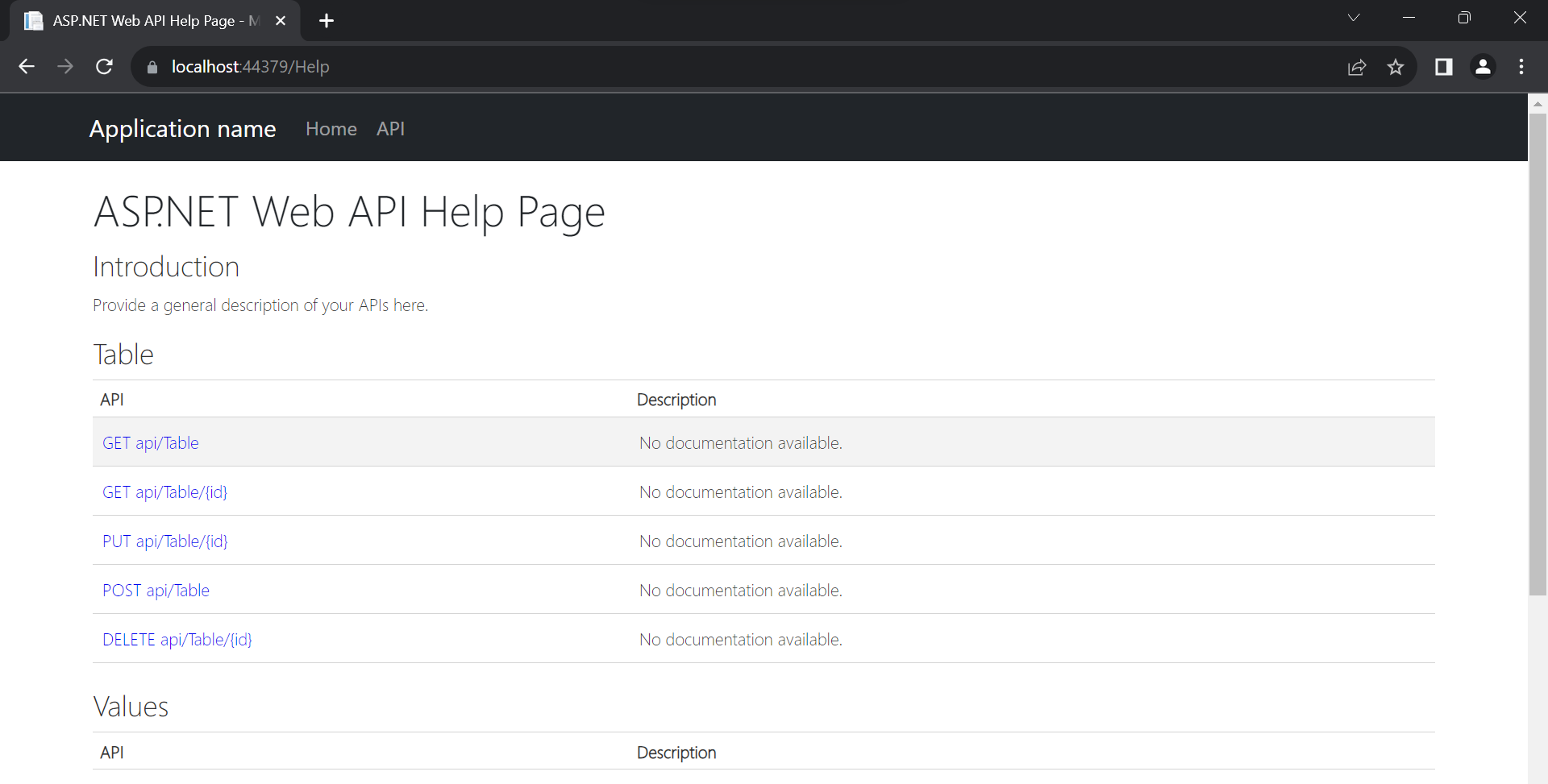


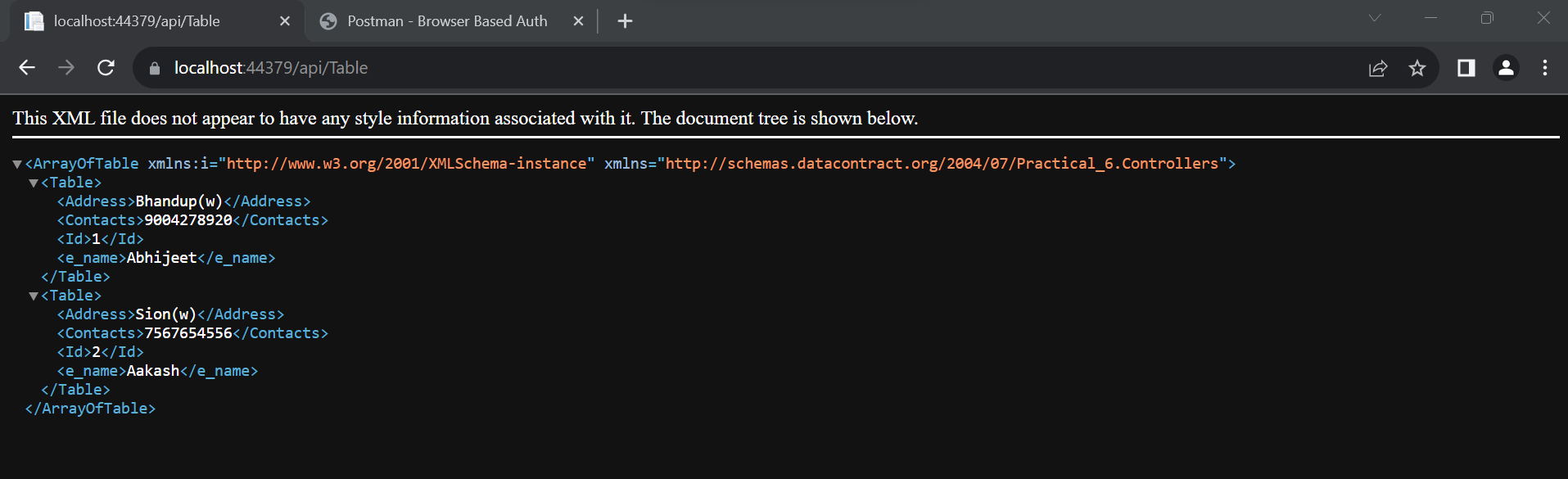


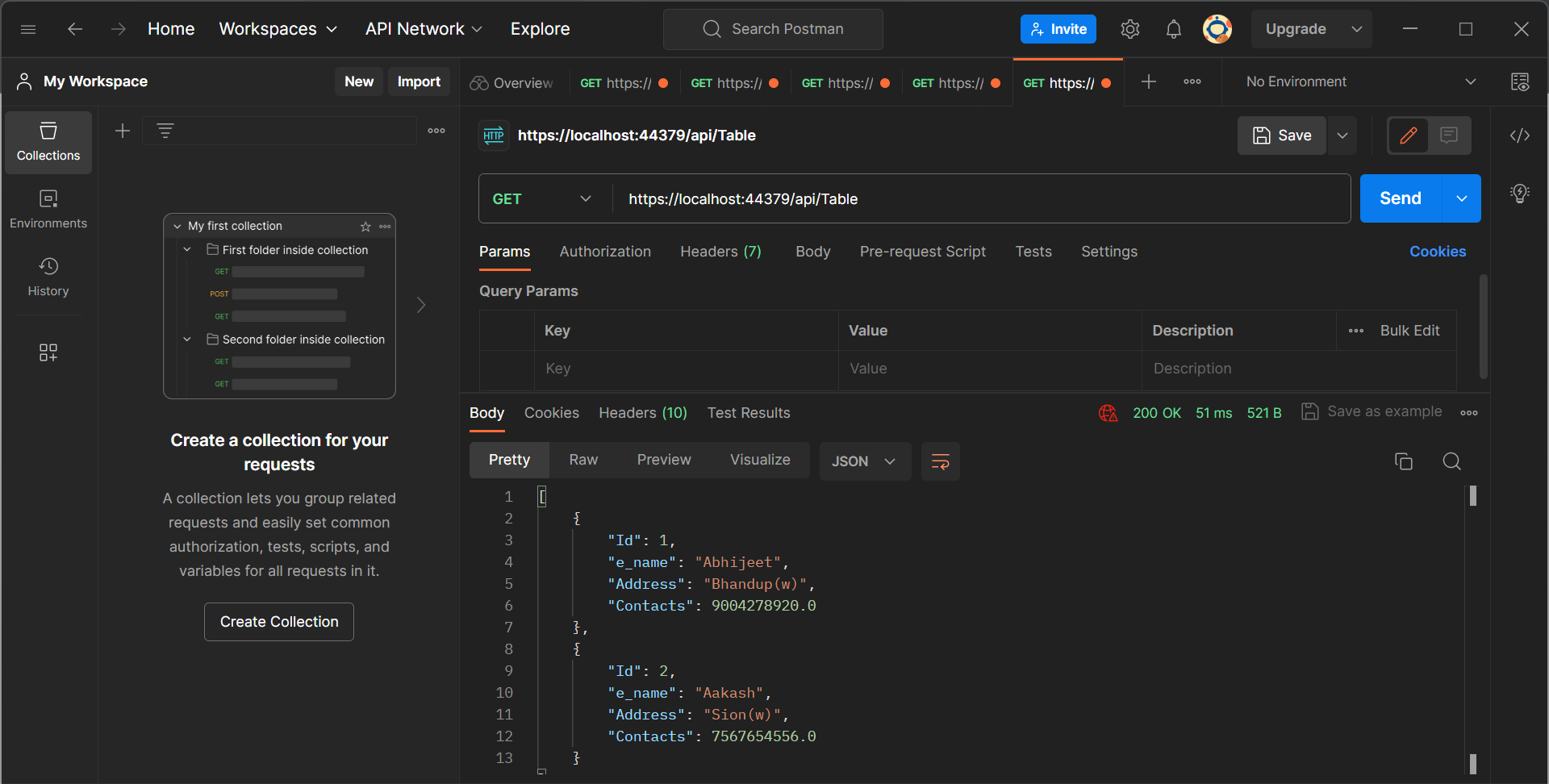




**Output**







Practical No 7

**Aim: Develop a Windows Azure Hello World application**

**Microsoft Azure**

1. Azure is a cloud computing platform and an online portal that allows you to access and manage cloud services and resources provided by Microsoft.
2. These services and resources include storing your data and transforming it, depending on your requirements.
3. To get access to these resources and services, all you need to have is an active internet connection and the ability to connect to the Azure portal.

**Things that you should know about Azure**

* It was launched on February 1, 2010, significantly later than its main competitor, AWS.
* It’s free to start and follows a pay-per-use model, which means you pay only for the services you opt for.
* Interestingly, 80 percent of the Fortune 500 companies use Azure services for their cloud computing needs.
* Azure supports multiple programming languages, including Java, Node Js, and C#.
* Another benefit of Azure is the number of data centers it has around the world. There are 42 Azure data centers spread around the globe, which is the highest number of data centers for any cloud platform. Also, Azure is planning to get 12 more data centers, which will increase the number of data centers to 54, shortly.

**Why Use Azure?**

Now that you know more about Azure and the services it provides, you might be interested in exploring the various uses of Azure.

**Application development**

You can create any web application in Azure.

**Testing**

After developing an application successfully on the platform, you can test it.

**Application hosting**

Once the testing is done, Azure can help you host the application.

**Create virtual machines**

You can create virtual machines in any configuration you want with the help of Azure.

**Integrate and sync features**

Azure lets you integrate and sync virtual devices and directories.

**Collect and store metrics**

Azure lets you collect and store metrics, which can help you find what works.

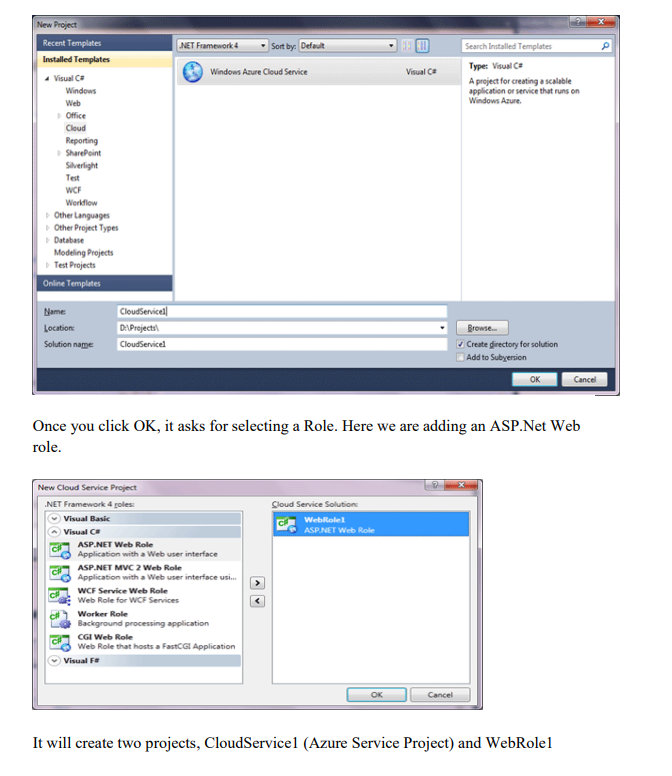
**Virtual hard drives**

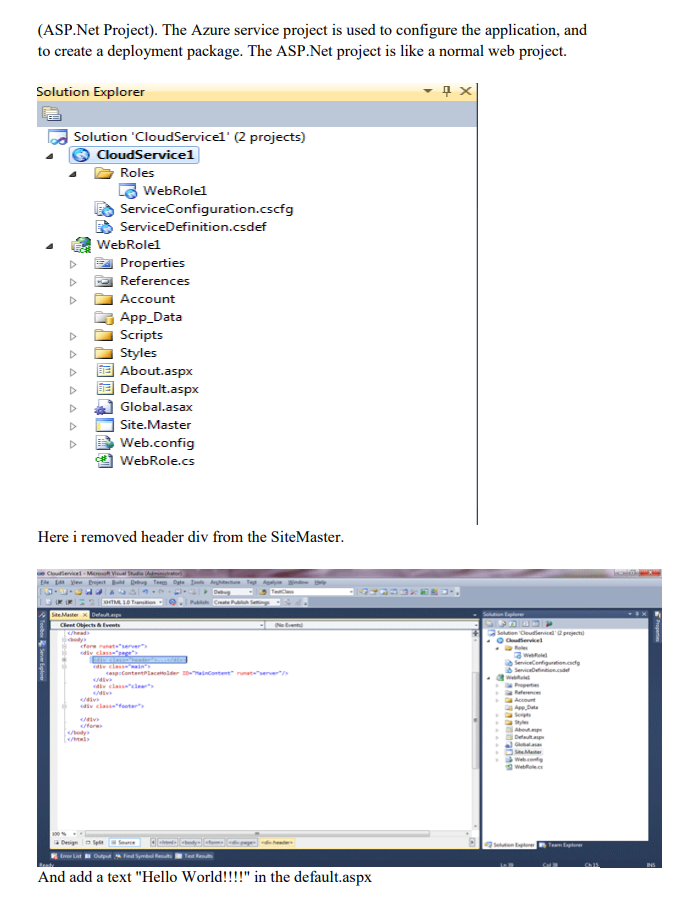
These are extensions of the virtual machines; they provide a huge amount of data storage.

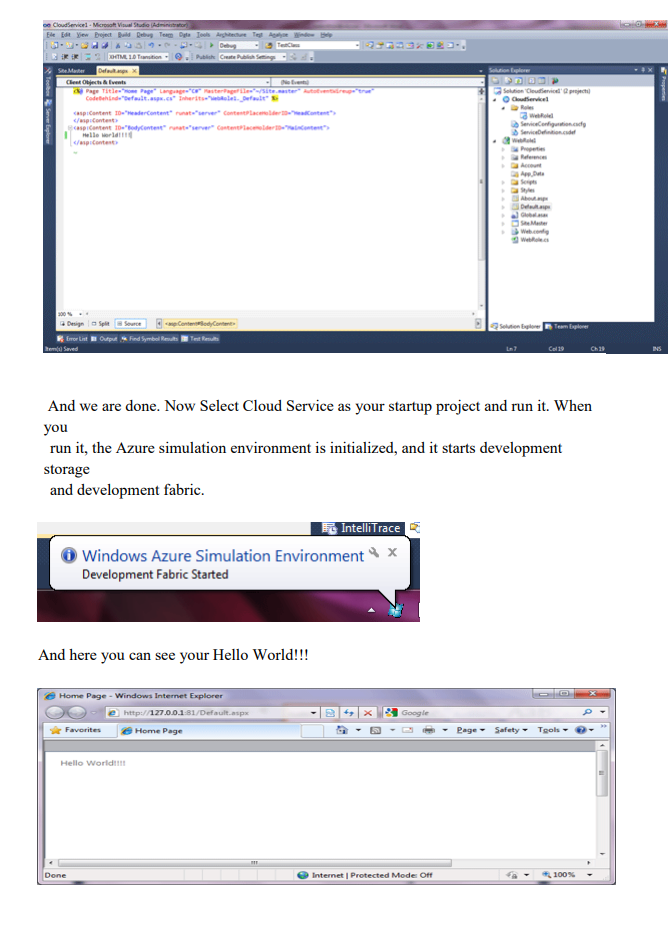
In this experiment, we will see how to create a "Hello World!!!"

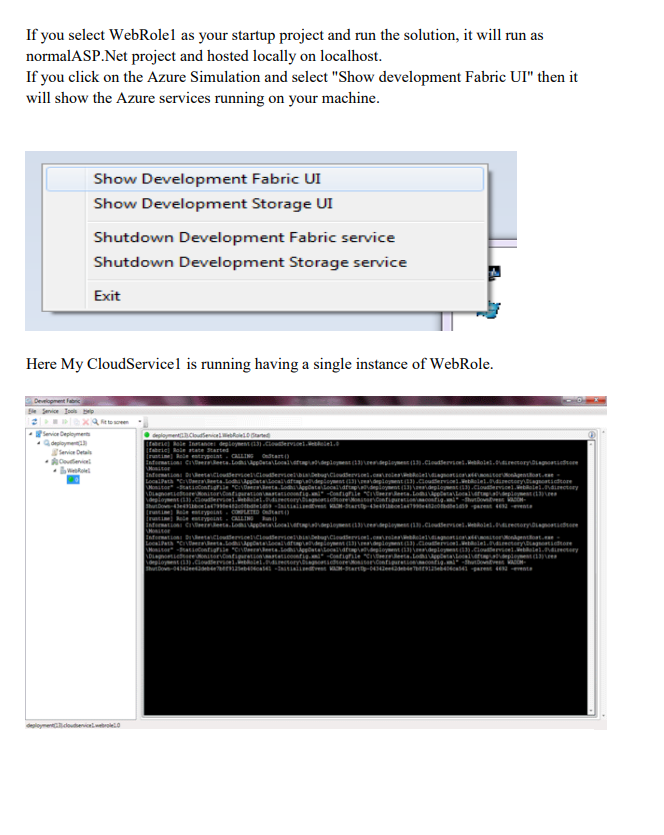
Start Visual Studio and select a new project.

In the cloud template select Windows Azure Cloud Service. Once you click OK, it asks for selecting a Role. Here we are adding an ASP.Net.









Practical No 8

Aim: To Install Google App Engine. Create hello world app and other simple web applications using python/java.

1. Google App Engine (often referred to by the acronym GAE or simply App Engine) is a cloud computing platform as a service for developing and hosting web applications in Google-managed data centers.
2. Applications are sandboxed and run across multiple servers.
3. App Engine offers automatic scaling for web applications—as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand.
4. Google App Engine primarily supports Go, PHP, Java, Python, Node.js,.NET, and Ruby applications, although it can also support other languages via "custom runtimes".
5. The service is free up to a certain level of consumed resources and only in standard environments; it is not available in flexible environments. Fees are charged for additional storage, bandwidth, or instance hours required by the application.
6. It was first released as a preview version in April 2008 and came out of preview in September 2011.

**Procedure**

Use Eclipse to create a Google App Engine (GAE) Java project (hello world example), run it locally, and deploy it to Google App Engine account.

**Tools used**

1) JDK 1.6

2) Eclipse 3.7 + Google Plugin for Eclipse

3) Google App Engine Java SDK 1.6.3.1 P.S Assume JDK1.6 and Eclipse 3.7 are installed.

**1. Install Google Plugin for Eclipse**

Read this guide – how to install Google Plugin for Eclipse. If you install the Google App Engine Java SDK together with “Google Plugin for Eclipse“, then go to step 2, Otherwise, get the Google App Engine Java SDK and extract it.

2. Create New Web Application Project In Eclipse toolbar, click on the Google icon, and select “New Web Application Project…”

