

LẬP TRÌNH C# 5

BÀI 6: WEB API - SQL SERVER

- ⦿ web API Template – Sql Server
- ⦿ web API user define – Sql Server
- ⦿ web Client gọi RESTful APIs




- ❑ web API - ASP.NET Core kết hợp được với nhiều loại csdl khác nhau (MySQL, Mongoddb, Sql server...)
- ❑ Có thể sử dụng template API hoặc tự tạo API tương tác trên csdl
- ❑ EF Core hỗ trợ cơ chế Scaffold tự tạo controller API với các API Action Methods cơ bản (HttpPost, HttpDelete, HttpPut, HttpGet...)





- ❑ Sử dụng EF core code first hoặc database first
- ❑ Tạo ứng dụng quản lý crud 'Reservation' bằng ASP.NET Core, chọn API template


Create a new ASP.NET Core web application

.NET Core ASP.NET Core 3.1

 **Empty**
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

 **API**
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

 **Web Application**
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

 **Web Application (Model-View-Controller)**
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

❑ Sử dụng EF core cấu hình kết nối csdl

❖ Thiết lập chuỗi kết nối

```
"ConnectionStrings": {  
  "ApiServerSql": "Server=.;Database=ApiCoreSql;Trusted_Connection=True;"  
},
```

❖ Đăng ký database context service (InventoryContext)

```
public void ConfigureServices(IServiceCollection services)  
{  
    var connection = Configuration.GetConnectionString("ApiServerSql");  
    services.AddDbContextPool<ConsumeClientContext>(options => options.UseSqlServer(connection));  
    services.AddControllers();  
}
```

❑ Sử dụng EF core có lớp Context và Entity

```
public ConsumeClientContext(DbContextOptions<ConsumeClientContext> options)
    : base(options)
```

```
{
}
```

6 references

```
public virtual DbSet<Reservation> Reservation { get; set; }
```

0 references

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
```

```
{
```

```
    modelBuilder.Entity<Reservation>(entity =>
```

```
    {
```

```
        entity.Property(e => e.EndLocation)
```

```
            .HasMaxLength(250)
```

```
            .IsUnicode(false);
```

```
        entity.Property(e => e.Name)
```

```
            .HasMaxLength(250)
```

```
            .IsUnicode(false);
```

```
        entity.Property(e => e.StartLocation)
```

```
            .HasMaxLength(250)
```

```
            .IsUnicode(false);
```

```
    });
```

```
OnModelCreatingPartial(modelBuilder);
```

```
public partial class Reservation
```

```
{
```

3 references

```
    public int Id { get; set; }
```

1 reference

```
    public string Name { get; set; }
```

1 reference

```
    public string StartLocation { get; set; }
```

1 reference

```
    public string EndLocation { get; set; }
```

```
}
```

- ❑ Sử dụng kỹ thuật scaffolding để build REST APIs
 1. Right-click the Controllers folder, choose Add, and then click Controller
 2. Select API Controller with actions using the Entity Framework template
 3. Choose model class and context class, and then name the controller

Add API Controller with actions, using Entity Framework

Model class: Reservation (ApiserverSql.Models)

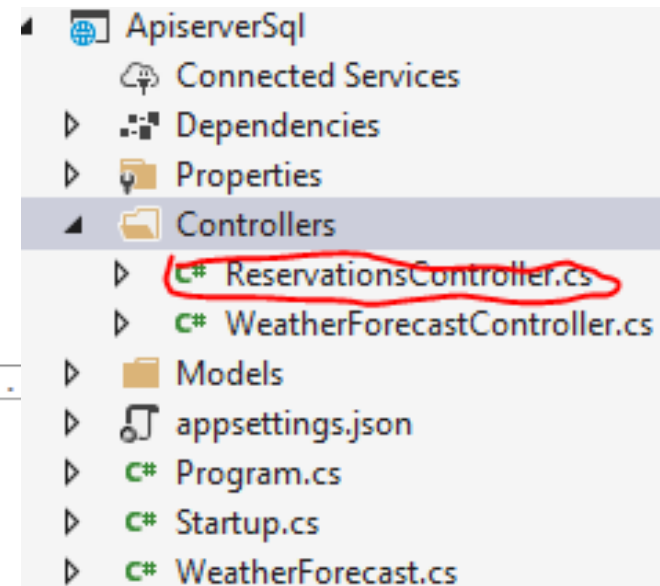
Data context class: ConsumeClientContext (ApiserverSql.Models) +

Controller name: ReservationsController

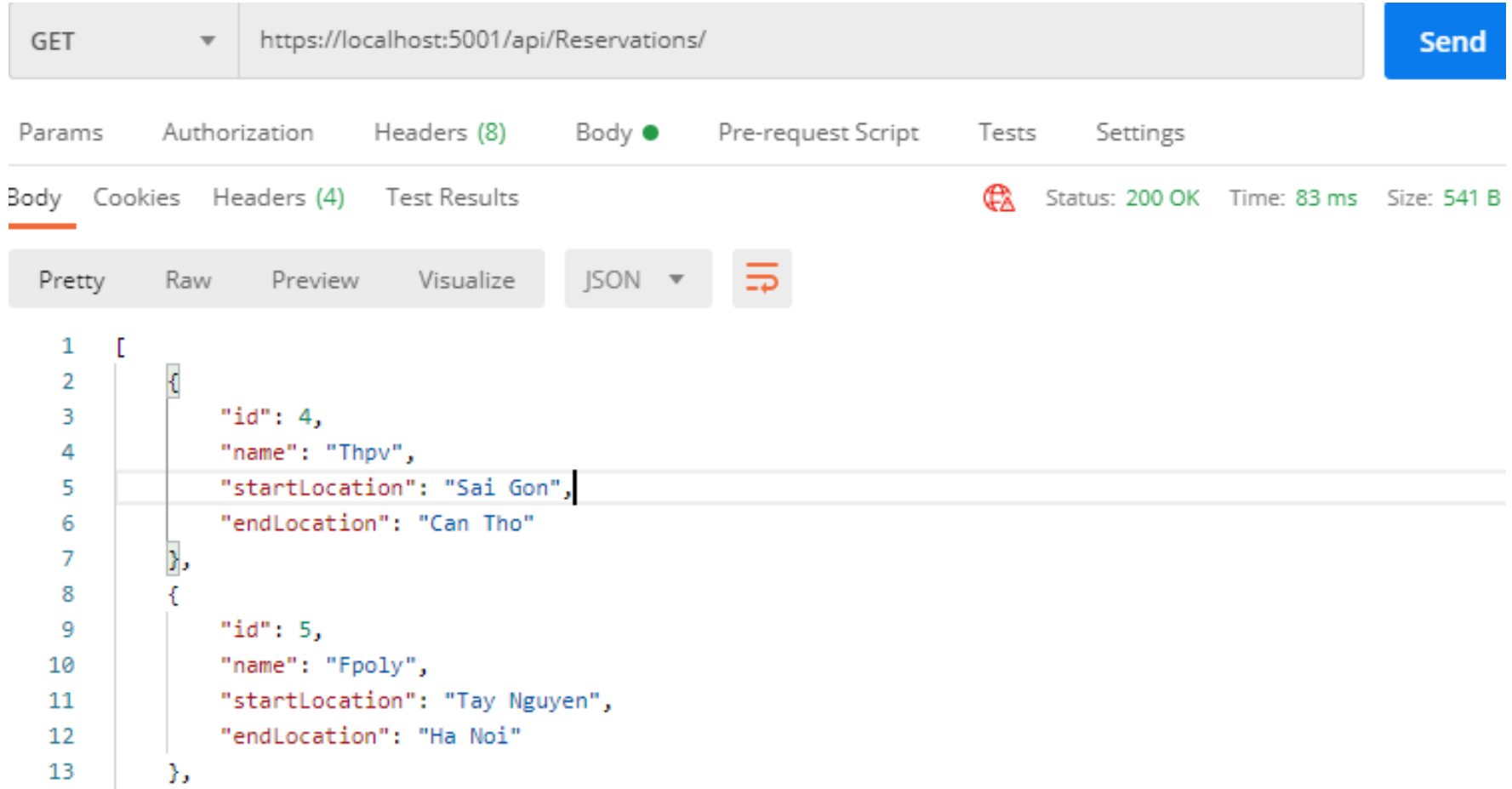
Add Cancel

- scaffolding để build REST APIs thành công, ứng dụng sẽ tự tạo controller với các API Action Methods cơ bản (HttpPost, HttpDelete, HttpPut, HttpGet)

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class ReservationsController : ControllerBase
{
    private readonly ConsumeClientContext _context;
    0 references
    public ReservationsController(ConsumeClientContext context) {...}
    // GET: api/Reservations
    [HttpGet]
    0 references
    public async Task<ActionResult<IEnumerable<Reservation>>> GetReservation() ..
    // GET: api/Reservations/5
    [HttpGet("{id}")]
    0 references
    public async Task<ActionResult<Reservation>> GetReservation(int id) {...}
    // PUT: api/Reservations/5 ...
    [HttpPut("{id}")]
    0 references
    public async Task<IActionResult> PutReservation(int id, Reservation reservation) {...}
    // POST: api/Reservations ...
    [HttpPost]
    0 references
    public async Task<ActionResult<Reservation>> PostReservation(Reservation reservation) {...}
    // DELETE: api/Reservations/5
    [HttpDelete("{id}")]
    0 references
    public async Task<ActionResult<Reservation>> DeleteReservation(int id) {...}
    1 reference
    private bool ReservationExists(int id) {...}
}
```



❑ Kiểm tra đọc dữ liệu với Postman



❑ Kiểm tra thêm dữ liệu với Postman

The screenshot shows the Postman interface for a POST request. The URL is `https://localhost:5001/api/Reservations/`. The request body is a JSON object with the following fields: `"name": "Thepv", "startLocation": "Tay Nguyen", "endLocation": "Sai Gon"`. The response status is `201 Created` with a time of `151 ms` and a size of `288 B`. The response body is a JSON object with the following fields: `"id": 1006, "name": "Thepv", "startLocation": "Tay Nguyen", "endLocation": "Sai Gon"`.

POST `https://localhost:5001/api/Reservations/` Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

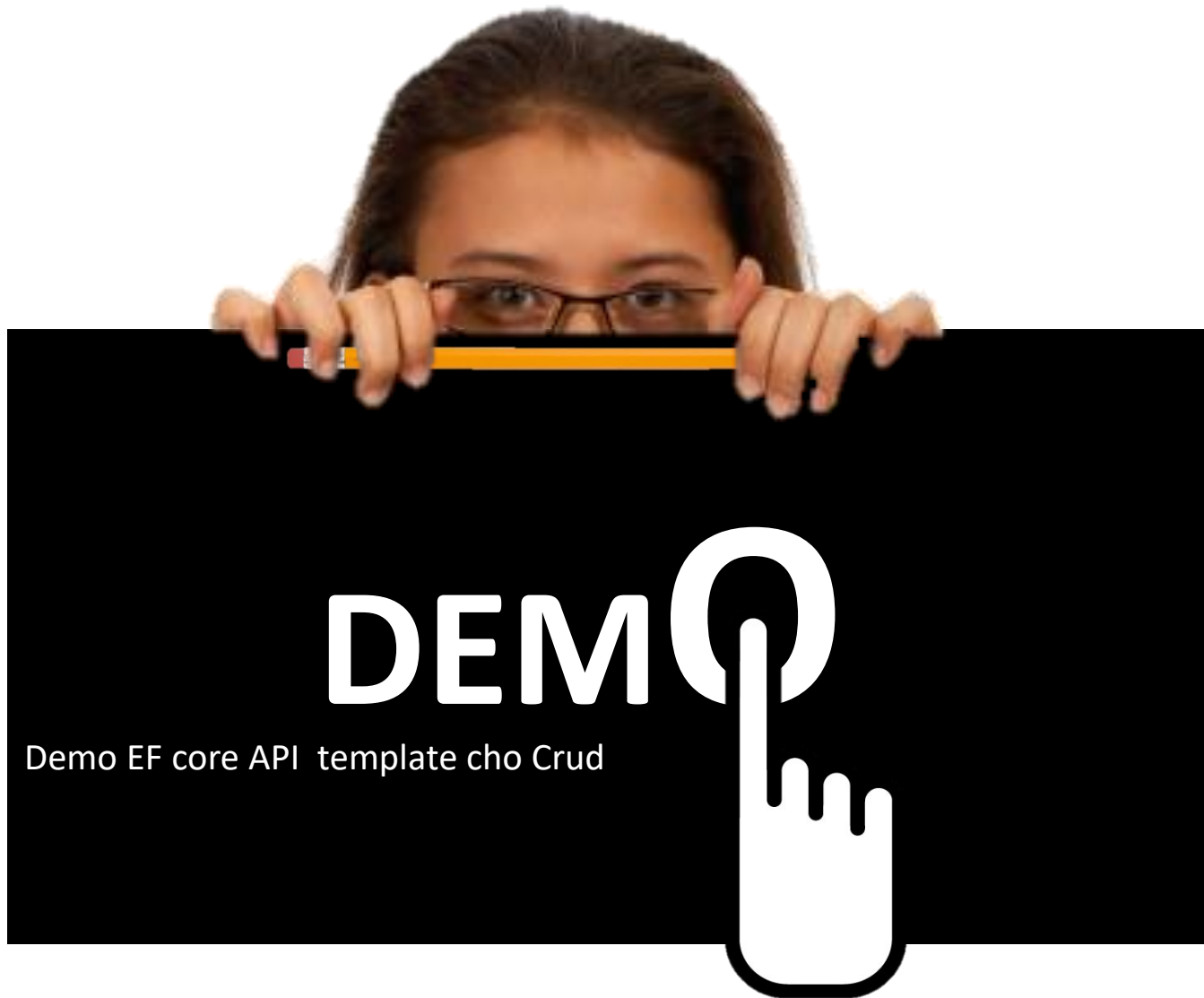
☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "name": "Thepv",
3   "startLocation": "Tay Nguyen",
4   "endLocation": "Sai Gon"
5 }
```

Body Cookies Headers (5) Test Results 🌐 Status: 201 Created Time: 151 ms Size: 288 B

Pretty Raw Preview Visualize **JSON** ▼ ↺

```
1 {
2   "id": 1006,
3   "name": "Thepv",
4   "startLocation": "Tay Nguyen",
5   "endLocation": "Sai Gon"
6 }
```




❑ Tạo ứng dụng Asp.net core Empty


Create a new ASP.NET Core web application

.NET Core


ASP.NET Core 3.1

 **Empty**


An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

 **API**


A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

 **Web Application**

A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

 **Web Application (Model-View-Controller)**

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

 **Angular**

Authentication

No Authentication

[Change](#)

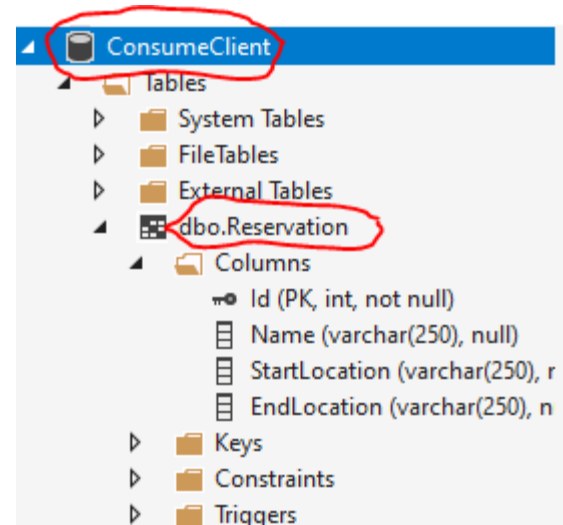
Advanced

☒ Configure for HTTPS

☐ Enable Docker Support
(Requires [Docker Desktop](#))

Linux

❑ Sử dụng Ef core cho csdl, model, context



```
public partial class Reservation
{
    0 references
    public int Id { get; set; }
    3 references
    public string Name { get; set; }
    3 references
    public string StartLocation { get; set; }
    3 references
    public string EndLocation { get; set; }
}
```

```
public ConsumeClientContext(DbContextOptions<ConsumeClientContext> options)
    : base(options)
{
    4 references
    public virtual DbSet<Reservation> Reservation { get; set; }
    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Reservation>(entity =>
        {
            entity.Property(e => e.EndLocation)
                .HasMaxLength(250)
                .IsUnicode(false);

            entity.Property(e => e.Name)
                .HasMaxLength(250)
                .IsUnicode(false);

            entity.Property(e => e.StartLocation)
                .HasMaxLength(250)
                .IsUnicode(false);
        });

        OnModelCreatingPartial(modelBuilder);
    }
}
```

❑ Tạo Interface và class quy định các thành phần crud

```
public class Repository : IRepository
{
    private readonly ConsumeClientContext _context;
    0 references
    public Repository(ConsumeClientContext context)
    {
        _context = context;
    }
    2 references
    public Reservation Reservations(int id)
    {
        var reservation = _context.Reservation.Find(id);

        if (reservation == null)
        {
            return null;
        }
        return reservation;
    }
    2 references
    public IEnumerable<Reservation> Reservations()
    { return _context.Reservation.ToList(); }
```

```
interface IRepository
{
    2 references
    public IEnumerable<Reservation> Reservations();
    2 references
    public Reservation Reservations(int id);
    2 references
    Reservation AddReservation(Reservation reservation);
    2 references
    Reservation UpdateReservation(Reservation reservation);
    2 references
    void DeleteReservation(int id);
}

public Reservation AddReservation(Reservation reservation)
{
    _context.Add(reservation);
    _context.SaveChanges();
    return reservation;
}
2 references
public void DeleteReservation(int id)
{
    var reservation = _context.Reservation.Find(id);
    _context.Reservation.Remove(reservation);
    _context.SaveChanges();
}
2 references
public Reservation UpdateReservation([FromForm]Reservation reservation)
{
    _context.Update(reservation);
    _context.SaveChanges();
    return reservation;
}
```

❑ Tạo controller và các api action method thông qua interface

```
[HttpPost]
0 references
public Reservation Post( Reservation res) =>
repository.AddReservation(new Reservation
{
    Name = res.Name,
    StartLocation = res.StartLocation,
    EndLocation = res.EndLocation
});
```

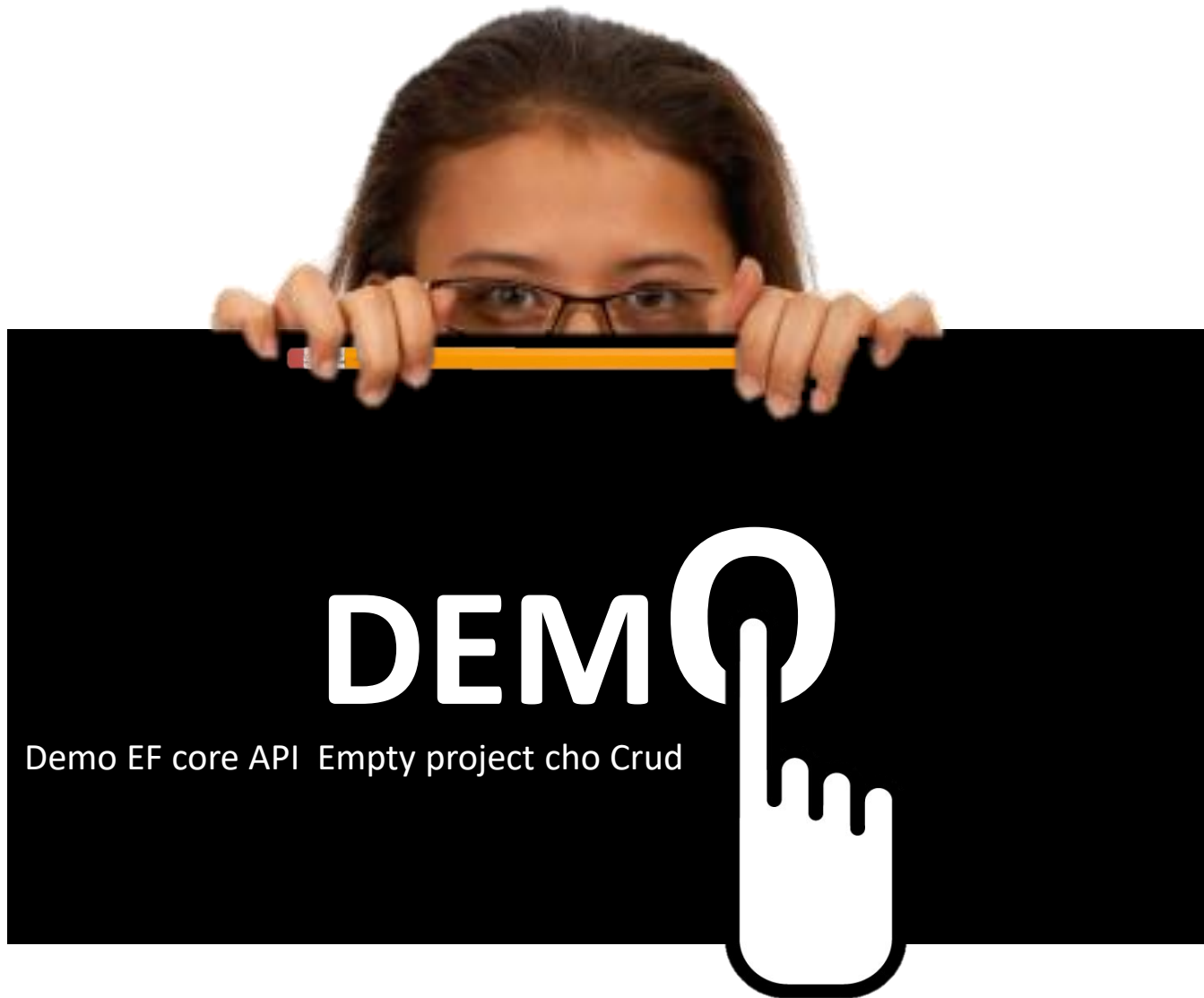
```
[HttpPut]
0 references
public Reservation Put([FromBody]Reservation res) => repository.UpdateReservation(res);
```

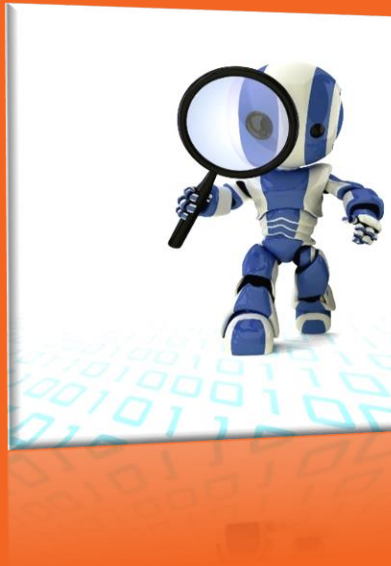
```
[HttpDelete("{id}")]
0 references
public void Delete(int id) => repository.DeleteReservation(id);
```

```
[ApiController]
[Route("api/[Reservation]")]
1 reference
public class ReservationController : ControllerBase
{
    private IRepository repository;
    0 references
    public ReservationController(IRepository repo) => repository = repo;

    [HttpGet]
    0 references
    public IEnumerable<Reservation> Get() => repository.Reservations();

    [HttpGet("{id}")]
    0 references
    public ActionResult<Reservation> Get(int id)
    {
        if (id == 0)
            return BadRequest("Value must be passed in the request body.");
        return Ok(repository.Reservations(id));
    }
}
```





LẬP TRÌNH C# 5

BÀI 6: WEB API - SQL SERVER (P2)

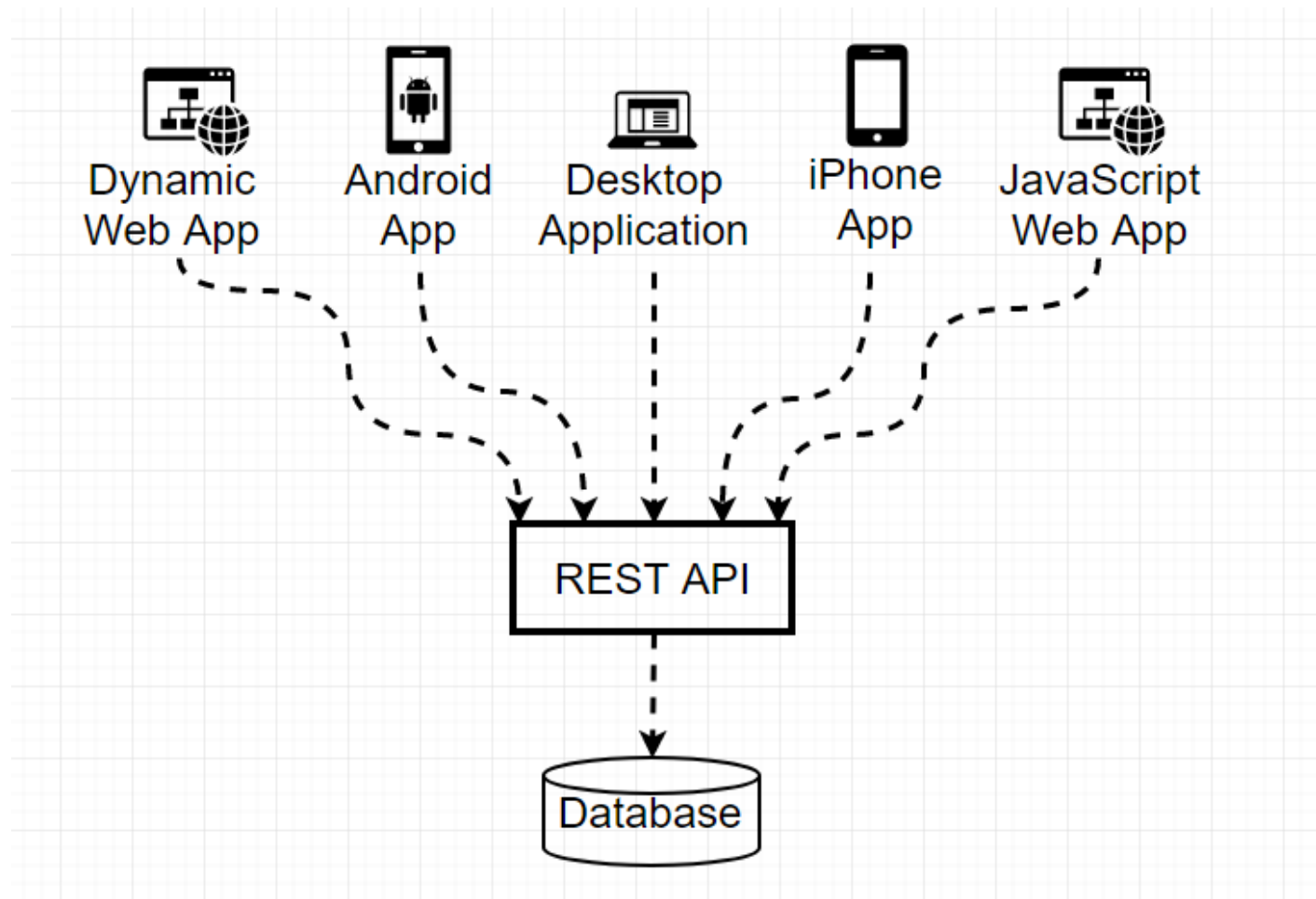
❑ Có nhiều cách sử dụng API khác nhau

- ❖ HttpWebRequest/Response Class
- ❖ WebClient Class
- ❖ HttpClient Class
- ❖ RestSharp NuGet Package
- ❖ ServiceStack Http Utils
- ❖ Flurl
- ❖ DalSoft.RestClient



□ Hầu hết các client side platform hiện nay đều tương thích và tương tác được với RESTful APIs

- ❖ JQuery
- ❖ Angular
- ❖ Android
- ❖ Ios
- ❖ ...



❑ Tạo ứng dụng Asp.net core Mvc

Create a new ASP.NET Core web application

The screenshot shows the 'Create a new ASP.NET Core web application' dialog in Visual Studio. At the top, there are two dropdown menus: '.NET Core' and 'ASP.NET Core 5.0', both highlighted with a red box. Below these, a list of project templates is displayed. The 'ASP.NET Core Empty' template is selected, indicated by a red arrow. The other templates listed are 'ASP.NET Core Web API', 'ASP.NET Core Web App', 'ASP.NET Core Web App (Model-View-Controller)', 'ASP.NET Core with Angular', and 'ASP.NET Core with React.js'. To the right of the template list, there are configuration options under 'Authentication' (No Authentication, Change) and 'Advanced' (Configure for HTTPS, Enable Docker Support). At the bottom right, there are 'Back' and 'Create' buttons, with the 'Create' button highlighted by a red arrow.

.NET Core ASP.NET Core 5.0

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

ASP.NET Core with Angular
A project template for creating an ASP.NET Core application with Angular

ASP.NET Core with React.js

[Get additional project templates](#)

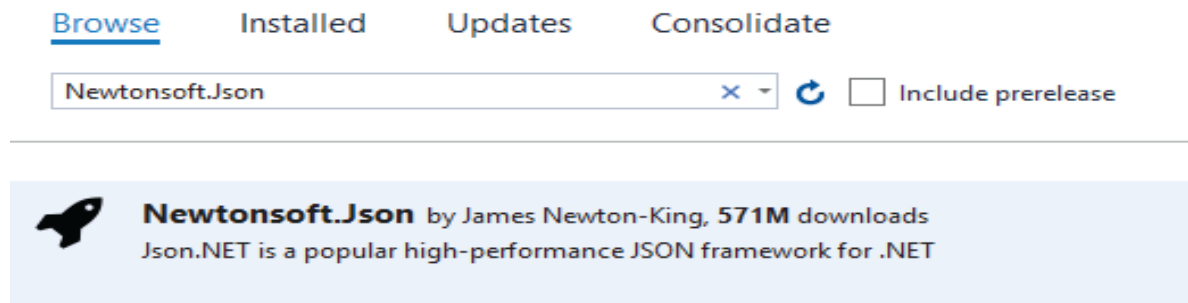
Authentication
No Authentication
[Change](#)

Advanced
☒ Configure for HTTPS
☐ Enable Docker Support
(Requires [Docker Desktop](#))
Windows

Author: Microsoft
Source: Templates 5.0.0

Back Create

- ❑ Cài đặt Newtonsoft.Json package: hỗ trợ Deserialize JSON sang object



- ❑ Sử dụng HttpClient() class tạo các Request tới RESTful APIs

Method	Description
GetAsync	Send a GET request to the specified Uri as an asynchronous operation.
PostAsync	Send a POST request to the specified Uri as an asynchronous operation.
PutAsync	Send a PUT request to the specified Uri as an asynchronous operation.
SendAsync	Send an HTTP request as an asynchronous operation.
PatchAsync	Sends a PATCH request with a cancellation token as an asynchronous operation.
DeleteAsync	Send a DELETE request to the specified Uri as an asynchronous operation.

- ❑ Tạo action trong controller gọi tới Url endpoint của RESTful APIs, xử lý DeserializeObject json sang object
- ❑ Action method phải được thiết kế bất đồng bộ khi dùng HttpClient

[HttpGet]

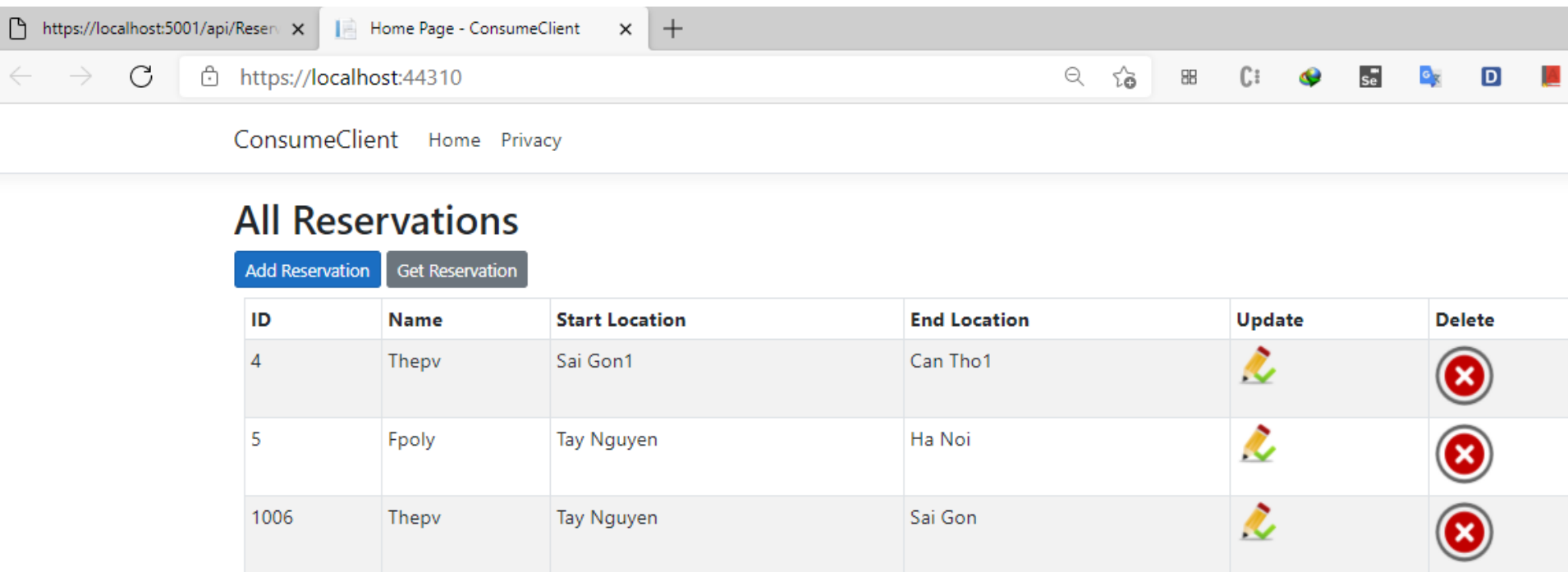
0 references

```
public async Task<IActionResult> GetAll()
{
    //Get all
    List<Reservation> reservationList = new List<Reservation>();
    using (var httpClient = new HttpClient())
    {
        using (var response = await httpClient.GetAsync("https://localhost:5001/api/Reservations"))
        {
            string apiResponse = await response.Content.ReadAsStringAsync();
            reservationList = JsonConvert.DeserializeObject<List<Reservation>>(apiResponse);
        }
    }
    return View(reservationList);
}
```







❑ Tạo view hiển thị kết quả với 2 button Delete và Update

```
<table class="table table-sm table-striped table-bordered m-2">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Start Location</th>
      <th>End Location</th>
      <th>Update</th>
      <th>Delete</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var r in Model)
    {
      <tr>
        <td>@r.Id</td>
        <td>@r.Name</td>
        <td>@r.StartLocation</td>
        <td>@r.EndLocation</td>
        <td>
          <a asp-action="UpdateReservation" asp-route-id="@r.Id">
            
          </a>
        </td>
        <td>
          <form asp-action="DeleteReservation" method="post">
            <input type="hidden" value="@r.Id" name="ReservationId" />
            <input type="image" src="/icon/close.png" />
          </form>
        </td>
      </tr>
    }
```

❑ Chạy RESTful APIs và client kiểm tra kết quả



The screenshot shows a web browser with two tabs: 'https://localhost:5001/api/Reserv...' and 'Home Page - ConsumeClient'. The address bar shows 'https://localhost:44310'. The page title is 'ConsumeClient' with links for 'Home' and 'Privacy'. The main content area is titled 'All Reservations' and contains two buttons: 'Add Reservation' (blue) and 'Get Reservation' (grey). Below the buttons is a table with the following data:

ID	Name	Start Location	End Location	Update	Delete
4	Thepv	Sai Gon1	Can Tho1		
5	Fpoly	Tay Nguyen	Ha Noi		
1006	Thepv	Tay Nguyen	Sai Gon		

❑ Yêu cầu dữ liệu từ Api theo Id

```
public ViewResult GetReservation() => View();  
[HttpPost]  
0 references  
public async Task<IActionResult> GetReservation(int id)  
{  
    Reservation reservation = new Reservation();  
    using (var httpClient = new HttpClient())  
    {  
        using (var response = await httpClient.GetAsync("https://localhost:5001/api/Reservation/" + id))  
        {  
            if (response.StatusCode == System.Net.HttpStatusCode.OK)  
            {  
                string apiResponse = await response.Content.ReadAsStringAsync();  
                reservation = JsonConvert.DeserializeObject<Reservation>(apiResponse);  
            }  
            else  
                ViewBag.StatusCode = response.StatusCode;  
        }  
    }  
    return View(reservation);  
}
```

Yêu cầu dữ liệu từ Api theo Id

```
@if (Model != null)
{
    <h2>Reservation</h2>
    <table class="table table-sm table-striped table-bordered m-2">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Start Location</th>
                <th>End Location</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>@Model.Id</td>
                <td>@Model.Name</td>
                <td>@Model.Start Location</td>
                <td>@Model.End Location</td>
            </tr>
        </tbody>
    </table>
}
```

Get Reservation by Id

Back

Id:

4

Get Reservation

Reservation

ID	Name	Start Location	End Location
4	Thepv	Sai Gon1	Can Tho1

❑ Thêm dữ liệu từ Api

```
public async Task<IActionResult> AddReservation(Reservation reservation)
{
    Reservation receivedReservation = new Reservation();
    using (var httpClient = new HttpClient())
    {
        StringContent content = new StringContent(JsonConvert.SerializeObject(reservation),
            Encoding.UTF8, "application/json");

        using (var response = await httpClient.PostAsync("https://localhost:5001/api/Reservations", content))
        {
            string apiResponse = await response.Content.ReadAsStringAsync();
            receivedReservation = JsonConvert.DeserializeObject<Reservation>(apiResponse);
        }
    }

    return View(receivedReservation);
}
```

□ Thêm dữ liệu từ Api

```
@if (Model != null)
{
    <h2>Reservation</h2>
    <table class="table table-sm table-striped table-bordered m-2">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Start Location</th>
                <th>End Location</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>@Model.Id</td>
                <td>@Model.Name</td>
                <td>@Model.StartLocation</td>
                <td>@Model.EndLocation</td>
            </tr>
        </tbody>
    </table>
    <form asp-action="AddReservation" method="post">
        <div class="form-group">
            <label for="Name">Name:</label>
            <input class="form-control" name="Name" />
        </div>
        <div class="form-group">
            <label for="StartLocation">Start Location:</label>
            <input class="form-control" name="StartLocation" />
        </div>
        <div class="form-group">
            <label for="EndLocation">End Location:</label>
            <input class="form-control" name="EndLocation" />
        </div>
        <div class="text-center panel-body">
            <button type="submit" class="btn btn-sm btn-primary">Add</button>
        </div>
    </form>
}
```

❑ Thêm dữ liệu từ Api

Add a Reservation

[Back](#)

Name:

Start Location:

End Location:

[Add](#)

Reservation

ID	Name	Start Location	End Location
1007	Insert data	HN	SG

❑ Cập nhật dữ liệu

[HttpPost]

0 references

```
public async Task<IActionResult> UpdateReservation(Reservation reservation)
{
    Reservation receivedReservation = new Reservation();
    using (var httpClient = new HttpClient())
    {
        var content = new MultipartFormDataContent();
        content.Add(new StringContent(reservation.Id.ToString()), "Id");
        content.Add(new StringContent(reservation.Name), "Name");
        content.Add(new StringContent(reservation.StartLocation), "StartLocation");
        content.Add(new StringContent(reservation.EndLocation), "EndLocation");
        using (var response = await httpClient.PutAsync("https://localhost:5001/api/Reservation", content))
        {
            string apiResponse = await response.Content.ReadAsStringAsync();
            ViewBag.Result = "Success";
            receivedReservation = JsonConvert.DeserializeObject<Reservation>(apiResponse);
        }
    }
    return View(receivedReservation);
}
```

□ Cập nhật dữ liệu

```
if (ViewBag.Result == "Success")
```

```
<h2>Reservation</h2>
<table class="table table-sm table-striped table-bordered m-2">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Start Location</th>
      <th>End Location</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>@Model.Id</td>
      <td>@Model.Name</td>
      <td>@Model.StartLocation</td>
      <td>@Model.EndLocation</td>
    </tr>
  </tbody>
</table>
```

```
<form method="post">
  <div class="form-group">
    <label asp-for="Id"></label>
    <input class="form-control" asp-for="Id" readonly />
  </div>
  <div class="form-group">
    <label asp-for="Name"></label>
    <input class="form-control" asp-for="Name" />
  </div>
  <div class="form-group">
    <label asp-for="StartLocation"></label>
    <input class="form-control" asp-for="StartLocation" />
  </div>
  <div class="form-group">
    <label asp-for="EndLocation"></label>
    <input class="form-control" asp-for="EndLocation" />
  </div>
  <div class="text-center panel-body">
    <button type="submit" class="btn btn-sm btn-primary">Update</button>
  </div>
</form>
```

❑ Cập nhật dữ liệu

Update a Reservation [Back](#)

Id

4

Name

Thepv

StartLocation

Sai Gon1

EndLocation

Can Tho1

[Update](#)

Reservation

ID	Name	Start Location	End Location
4	Thepv	Sai Gon	Can Tho



Tổng kết bài học

- ⦿web API Template – Sql Server
- ⦿web API user define – Sql Server
- ⦿web Client gọi RESTful APIs





KẾT THÚC