Haute École Léonard de Vinci
SCIENCES ET TECHNIQUES

I314A
.NET Outils et Concepts d'Application d'Entreprise

ASP.NET Core Web API

Sommaire

- SOAP vs REST
- Introduction SOA
- SOA : principes
- SOA vs REST
- ASP.NET Web API – version minimale
- ASP.NET Web API – version avec Controllers

SOAP vs REST

- SOAP
  - Multi-protocole (HTTP, TCP, ….)
  - Support transaction
  - Sécurité accrue (contrats)
  - Rigidité (contrats)
  - SOAP disparait de + en en +
- REST
  - Uniquement HTTP/HTTPS
  - Simple (facile pour exposer une API)
  - Trt données -> XML, JSON, ….

SOAP vs REST

- Le framework . NET
  - SOAP
    - Via WCF
  - REST
    - Via WCF REST
    - Via ASP.NET Web Api

ASP.NET Core WEB API

- API minimale
  - Pour micro-service
  - Dépendances minimales
  - Tout dans un (minimum de) fichier(s)
- API avec controllers
  - Classe controller héritent de ControllerBase
  - API peut être répartie sur plusieurs controllers
  - Configuration via annotations
    - [HttpPost], [HttpGet], ….
    - [ApiController]
    - [Route("[controller]")]
    - `[ProducesResponseType(200, Type = typeof(CustomerDTO))]`

API Minimale - Exemple

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment()){

// use of Swagger !!!

        app.UseSwagger();

        app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// GET /hello.

app.MapGet("/hello", () => { return("Hello World"); })

app.Run();
```

Swagger

- Implémente les spécifications OpenAPI (standard)
- Documentation de l'API interactive (SwaggerUI)
- Existe pour beaucoup de langages (Node.js, .NET, ….)

API avec Controllers

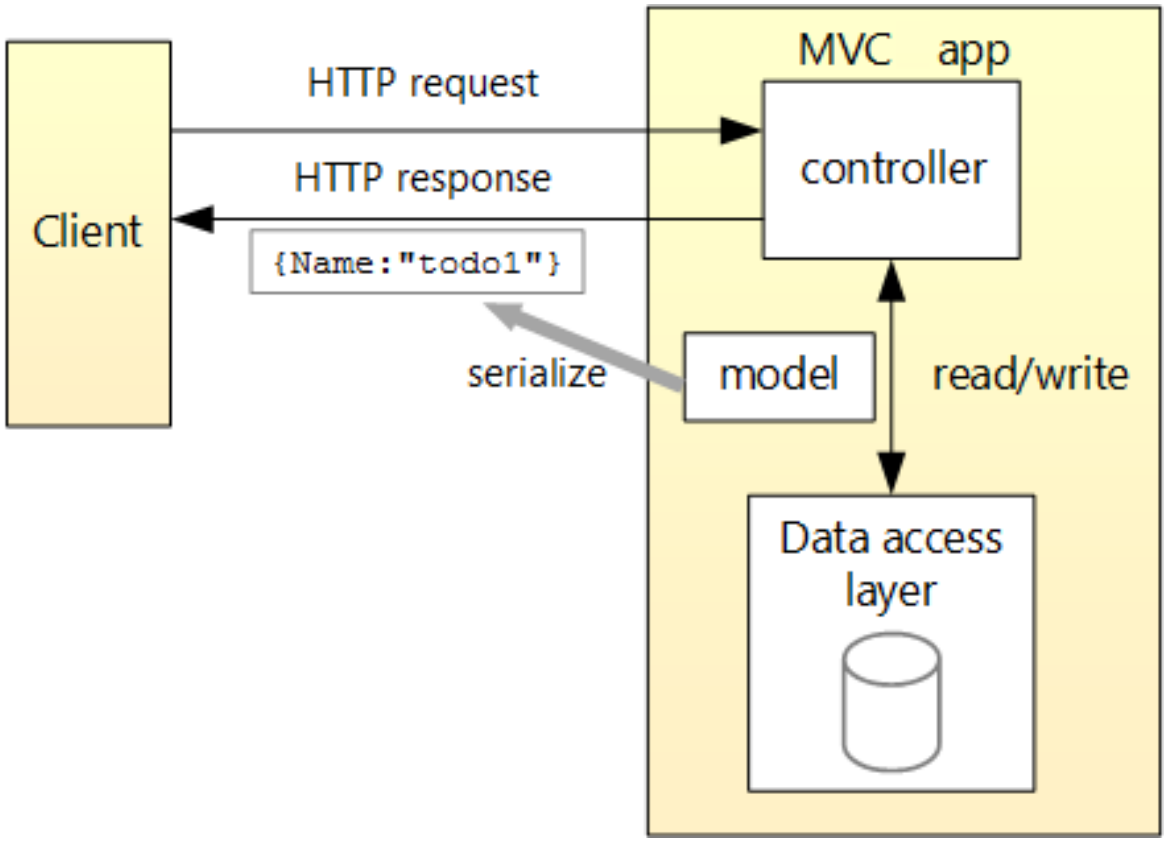

Image issue de : https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio

API avec Controllers

- Un dossier Controllers
- Un dossier avec les modèles
  - Modèles seront des DTO et pas les Entities
    - Sécurité : Inutile d'exposer tous les champs d'une entité
    - Performance : Inutile de transférer tous les champs sur le réseau
    - Erreur : supprimer les références circulaires
    - …

Inférence modèle

- Le modèle sera désérialisé automatiquement quand il sera envoyé vers l'API dans un modèle
  - Ex: `public ActionResult PostBook([FromBody] BookDTO book)`
- Il est possible d'indiquer au framework où/comment rechercher l'information pour créer le modèle
  - [FromBody] : récupérer les valeurs depuis le request body
  - [FromQuery] : récupérer les valeurs depuis le query string
  - [FromHeader]
  - [FromForm]
  - …
- Les méthodes POST utilise généralement [FromBody] avec leurs arguments. [FromBody] s'attend à recevoir un modèle en JSON !

Type de retour

- ActionResult<T>
  - Ok
  - NotFound
  - CreatedRoute
  - Accepted
  - …
- Ex :

```
[HttpGet]
public ActionResult<CustomerDTO> GetCustomer(int id)
{
    Customer c = repo.GetById(id);
    if (c == null) return NotFound();
    else {
      CustomerDTO cdto = toDTO(c);
      return Ok(cdto);
    }
}
```

API avec Controllers

```
// Program.cs
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

API avec Controllers

```
// HelloController.cs
[ApiController]
[Route("/api/[controller]")]
public class HelloController : ControllerBase {

      // GET /api/Hello
      [HttpGet]
      public string Hello(string Name)
      {
            return "Hello "+ Name;
      }
}
```

API vec Controllers

```
// HelloController.cs
[ApiController]
[Route("/api/[controller]")]

[HttpGet("hello")]  -> /api/Hello/hello
public string Get()

[HttpGet("{id}")]  -> /api/Hello/5
public string Get(int id)
```

Logging

- Program.cs

```
builder.Services.AddHttpLogging(options =>
{
    options.LoggingFields = HttpLoggingFields.All;
    options.RequestBodyLogLimit = 4096;
    options.ResponseBodyLogLimit = 4096;
});
…
var app = builder.Build();
app.UseHttpLogging();
```

Logging

- appsettings.Development.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Information"
    }
  }
}
```

Plus d'informations

- https://docs.microsoft.com/fr-fr/aspnet/core/web-api/?view=aspnetcore-6.0