Правильная валидация в веб-приложениях

Пётр Кудинов

Правильная* валидация в веб-приложениях

Пётр Кудинов

---

* Для какого-то определения слова "правильная".

Что такое валидация?

Проверка входных данных на соответствие критериям.

Зачем она нужна?

Чтобы не повредить существующие данные, не выполнить недопустимую операцию и т.д.

Проблема: дублирование кода

- Transaction script:
  - client
  - server
- Domain model:
  - client,
  - app service / controller / DTO attributes
  - entity

Пример: ValidationAttribute, entity

```
class /* App. */PersonDto {
  [Required]
  [StringLength(100)]
  public string FirstName { get; set; }
}
class /* Domain. */Person {
  public Person(string firstName) {
    if(string.IsNullOrEmpty(firstName))
      throw ArgumentNullException(nameof(firstName));
    if(firstName.Length > 100)
      throw ArgumentException("Name too long", nameof(firstName));
    FirstName = firstName;
  }
  public string FirstName { get; private set; }
}
```

Пример: ValidationAttribute, entity

```
class /* App. */PersonDto {
  [Required]
  [StringLength(100)]
  public string FirstName { get; set; }
}
class /* Domain. */Person {
  public Person(string firstName) {
    if(string.IsNullOrEmpty(firstName))
      throw ArgumentNullException(nameof(firstName));
    if(firstName.Length > 100)
      throw ArgumentException("Name too long", nameof(firstName));
    FirstName = firstName;
  }
  public string FirstName { get; private set; }
}
```

Бизнес-правила продублированы в сущности и второстепенном коде.

Что, если не валидировать в домене?

```
class /* App. */PersonDto {
  [Required]
  [StringLength(100)]
  public string FirstName { get; set; }
}
class /* Domain. */Person {
  public Person(string firstName) {
    FirstName = firstName;
  }
  public string FirstName { get; private set; }
}
```

Если сущность создаётся не через DTO, валидации не происходит.

Что, если валидировать **только** в домене?

```
class /* App. */PersonDto {
  public string FirstName { get; set; }
}
class /* Domain. */Person {
  public Person(string firstName) {
    if(string.IsNullOrEmpty(firstName))
      throw ArgumentNullException(nameof(firstName));
    if(firstName.Length > 100)
      throw ArgumentException("Name too long", nameof(firstName));
    FirstName = firstName;
  }
  public string FirstName { get; private set; }
}
```

- При ошибках валидации отвечаем 500.
- `ArgumentException` слишком общий.

Будем бросать специальное исключение

```
public Person(string firstName) {
  if(string.IsNullOrEmpty(firstName))
    throw new ValidationException(nameof(firstName), "Required");
  if(firstName.Length > 100)
    throw new ValidationException(nameof(firstName), "Name too long");
  FirstName = firstName;
}
```

… и обрабатывать его в Middleware

```
public class ErrorHandlerMiddleware {
  // ...
  public async Task InvokeAsync(HttpContext context) {
    try {
      await _next(context);
    } catch (ValidationException ve) {
      // ...
      var result = new ObjectResult(ve.Message) {
        StatusCode = StatusCodes.Status400BadRequest
      };
      await result.ExecuteResultAsync(actionContext);
    }
  }
}
```

Проблема: сообщаем только о первой найденной ошибке.

```
public Person(string firstName, string lastName) {
  if(string.IsNullOrEmpty(firstName))
    throw new ValidationException(nameof(firstName),
      "First name is required");
  if(string.IsNullOrEmpty(lastName))
    throw new ValidationException(nameof(lastName),
      "Last name is required");
  // ...
}
```

Решение: воспользуемся библиотекой Admonish.

```
public Person(string firstName, string lastName) {
  Validator
    .Create()
    .NonNullOrWhiteSpace(nameof(firstName), firstName)
    .NonNullOrWhiteSpace(nameof(lastName), lastName)
    .ThrowIfInvalid();
  // ...
}
```

Ошибки накапливаются до `ThrowIfInvalid()`.

Admonish

- Валидирует примитивные типы, не DTO.
- Исключение можно заменить.

```
public void ConfigureServices(IServiceCollection services)
{
  // ...
  Admonish.Validator.UnsafeConfigureException(
    ValidationResult r =>
      new CustomValidationException(r.ToDictionary()));
}
```

Почему не FluentValidation?

Почему не FluentValidation?

— Я написал вот такое, почему оно неправильно валидирует?

Почему не FluentValidation?

— Я написал вот такое, почему оно неправильно валидирует?

```
RuleFor(x =>x.RequestAmount.TryParseDecimal())
.LessThan(x => x.ContractAmount.TryParseDecimal()
  - _purchaseOrderInvoiceService.GetPaymentTotalByContractId(
    x.ContractId.Value))
.When(x => x.ContractId != null
  && _contractService.GetById(x.ContractId.Value).Amount > 0)
.When(x => x.SaveCommand == FinancePurchaseOrderCommand.SubmitPurchaseO
.WithMessage("You cannot request an amount greater than ...");
```

Почему не FluentValidation?

— Я написал вот такое, почему оно неправильно валидирует?

```
RuleFor(x =>x.RequestAmount.TryParseDecimal())
.LessThan(x => x.ContractAmount.TryParseDecimal()
  - _purchaseOrderInvoiceService.GetPaymentTotalByContractId(
    x.ContractId.Value))
.When(x => x.ContractId != null
  && _contractService.GetById(x.ContractId.Value).Amount > 0)
.When(x => x.SaveCommand == FinancePurchaseOrderCommand.SubmitPurchaseO
.WithMessage("You cannot request an amount greater than ...");
```

Почему не FluentValidation?

— Я написал вот такое, почему оно неправильно валидирует?

```
RuleFor(x =>x.RequestAmount.TryParseDecimal())
.LessThan(x => x.ContractAmount.TryParseDecimal()
  - _purchaseOrderInvoiceService.GetPaymentTotalByContractId(
    x.ContractId.Value))
.When(x => x.ContractId != null
  && _contractService.GetById(x.ContractId.Value).Amount > 0)
.When(x => x.SaveCommand == FinancePurchaseOrderCommand.SubmitPurchaseO
.WithMessage("You cannot request an amount greater than ...");
```

Слишком гибко. Слишком много свободы.

Формат ответа при ошибке

```
curl http://localhost:5000/api/entities \
  -H 'Content-Type: application/json' -d '{name: \"\", age: -2}'
```

```
{
  "errors":{
    "age":["The value must be greater than or equal to 0."],
    "name":["The value must not be null or empty ..."]
  },
  "type":"urn:acme-corp:validation-error",
  "title":"One or more validation errors occurred.",
  "status":400
}
```

ProblemDetails

https://tools.ietf.org/html/rfc7807

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
"type": "https://example.net/validation-error",
"title": "Your request parameters didn't validate.",
"errors": [ {
      "name": "age",
      "reason": "must be a positive integer"
    },
    {
      "name": "color",
      "reason": "must be 'green', 'red' or 'blue'"}
  ]
}
```

ProblemDetails

- Стандарт (все совместимые клиенты поймут).
- type - это URI, он не меняется. Можно хардкодить на клиенте и основывать поведение на нём.

Настройка ASP.Net Core

ProblemDetails при ошибках, найденных встроенным механизмом ASP.Net (GET-запросы, ошибки парсинга DTO).

```
// Startup.cs
[assembly: ApiController]
namespace WebApplication {
    // ...
}
```

```
public void ConfigureServices(IServiceCollection services) {
  services.AddMvc()
    .SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
  // ...
}
```

Пример ответа при `CompatibilityVersion.Version_2_1`

```
curl "http://localhost:5000/api/entities/count?minAge=a"
```

```
[Route("count")]
[HttpGet]
public int GetCount(int minAge) {
  return _service.GetCount(minAge);
}
```

```
{"minAge":["The value 'a' is not valid."]}
```

✕ Не ОК.

Пример ответа при CompatibilityVersion.Version_2_2

```
curl "http://localhost:5000/api/entities/count?minAge=a"
```

```
[Route("count")]
[HttpGet]
public int GetCount(int minAge) {
  return _service.GetCount(minAge);
}
```

```
{"errors":{"minAge":["The value 'a' is not valid."]},
"title":"One or more validation errors occurred.",
"status":400,
"traceId":"0HLL5F74T6VSV:00000001"}
```

☑ ОК

Нерешённые проблемы

- Теряется структура DTO (в ошибке видим имя параметра entity-метода, а не свойства DTO).
- Можно забыть вызов `ThrowIfInvalid()`.
- Локализация?

Резюме. Как валидировать?

- Накапливать ошибки (можно использовать Admonish для этого).
- Бросать специальное исключение.
- Обрабатывать исключение в Middleware.
- Отвечать в формате Problem Details (RFC 7807).

Пример приложения

http://bit.ly/admonish-sample