# NestJS – Day 9 Assignment

### 1. Create a DTO class for a CreateProductDto with the following fields:

```
export class CreateProductDto {
  name: string;
  price: number;
  tags?: string[];
}
```

```
export class CreateProductDto {
  name: string;

  price: number;

  tags?: string[];
}
```

## 2. Add validation rules: price >= 1, tags at least 2 characters each:

```
import {
  IsString,
  IsOptional,
  IsNotEmpty,
  Min,
  MinLength,
  IsNumber,
  IsArray,
} from 'class-validator';

export class CreateProductDto {

  @IsNotEmpty()
  @IsString()
  name: string;

  @IsNumber()
  @Min(1)
  price: number;


  @IsOptional()
  @IsArray()
  @MinLength(2, { each: true })
  tags?: string[];


}

```

## 3. In your controller, apply a ValidationPipe globally to validate incoming DTOs.

```
import { Body, Controller, Get, Post, UsePipes, ValidationPipe } from '@nestjs/common';
import { CreateProductDto } from './dto/create-product.dto/create-product.dto';

@Controller('product')
@UsePipes(new ValidationPipe())
export class ProductController {

    @Post()
    getProduct(@Body() dto:CreateProductDto){
        return {message :dto};
    }


}
```

POST http://localhost:3000/product Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{"bio":"Jafar"}
```

Status: 400 Bad Request Size: 200 Bytes Time: 12 ms

Response Headers 6 Cookies Results Docs

```
{
  "message": [
    "name must be a string",
    "name should not be empty",
    "price must not be less than 1",
    "price must be a number conforming to the specified constraints"
  ],
  "error": "Bad Request",
  "statusCode": 400
}
```

POST http://localhost:3000/product Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{"name":"Jafar",
  "price":1233,
  "tags":["tag1","tag2"]

}
```

Status: 201 Created Size: 64 Bytes Time: 6 ms

Response Headers 6 Cookies Results Doc

```
{
  "message": {
    "name": "Jafar",
    "price": 1233,
    "tags": [
      "tag1",
      "tag2"
    ]
  }
}
```

## 4. POST /products route using CreateProductDto:

```
@Post()
create(@Body() dto: CreateProductDto) {
 return dto;
}
```

```
import { Body, Controller, Get, Post, UsePipes, ValidationPipe } from '@nestjs/common';
import { CreateProductDto } from './dto/create-product.dto/create-product.dto';

@Controller('product')
@UsePipes(new ValidationPipe())
export class ProductController {

    @Post()
    getProduct(@Body() dto:CreateProductDto){
        return {message :dto};
    }


}
```

## 5. Test endpoint using Postman or curl:

POST http://localhost:3000/product Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{"name":"Jafar",
  "price":"Free",
  "tags":["tag1","tag2"]

}
```

Status: 400 Bad Request Size: 149 Bytes Time: 7 ms

Response Headers 6 Cookies Results Docs

```
{
  "message": [
    "price must not be less than 1",
    "price must be a number conforming to the specified constraints"
  ],
  "error": "Bad Request",
  "statusCode": 400
}
```

POST http://localhost:3000/product Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{"name":"Jafar",
  "price":1234,
  "tags":["tag1","tag2","tag3","tag4"]

}
```

Status: 201 Created Size: 78 Bytes Time: 5 ms

Response Headers 6 Cookies Results Docs

```
{
  "message": {
    "name": "Jafar",
    "price": 1234,
    "tags": [
      "tag1",
      "tag2",
      "tag3",
      "tag4"
    ]
  }
}
```

### 6. Use ParseIntPipe on GET /users/:id route:

```
12
13
14      @Get("/:id")
15      getProduct(@Param('id',ParseIntPipe)id:number){
16  return {ID:id};
17
18      }
19
```

GET http://localhost:3000/product/1 Send

Status: 200 OK Size: 8 Bytes Time: 40 ms

Query Headers 2 Auth Body 1 Tests Pre Run

Response Headers 6 Cookies Results

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "ID": 1
3 }
```

### 7. Use ParseBoolPipe on GET /status?isActive=true:

```
@Get("/status")
getStatus(@Query('isActive',ParseBoolPipe) isActive: boolean):boolean {
  console.log('Received:', isActive);
  return isActive;
}
```

GET http://localhost:3000/product/status?isActive=true Send

Status: 200 OK Size: 4 Bytes

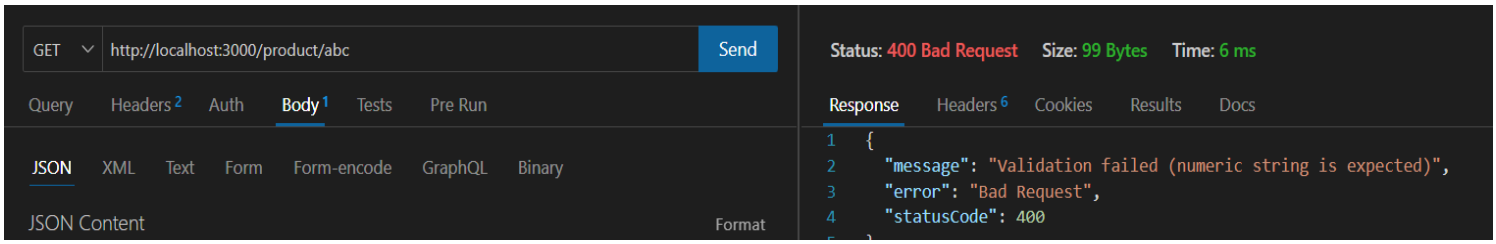
Query Headers 2 Auth Body 1 Tests Pre Run

Response Headers 6 Cooki

JSON XML Text Form Form-encode GraphQL Binary

```
1 true
```

## 8. If 'abc' passed to ParseIntPipe:




## 9. Create custom TrimPipe:

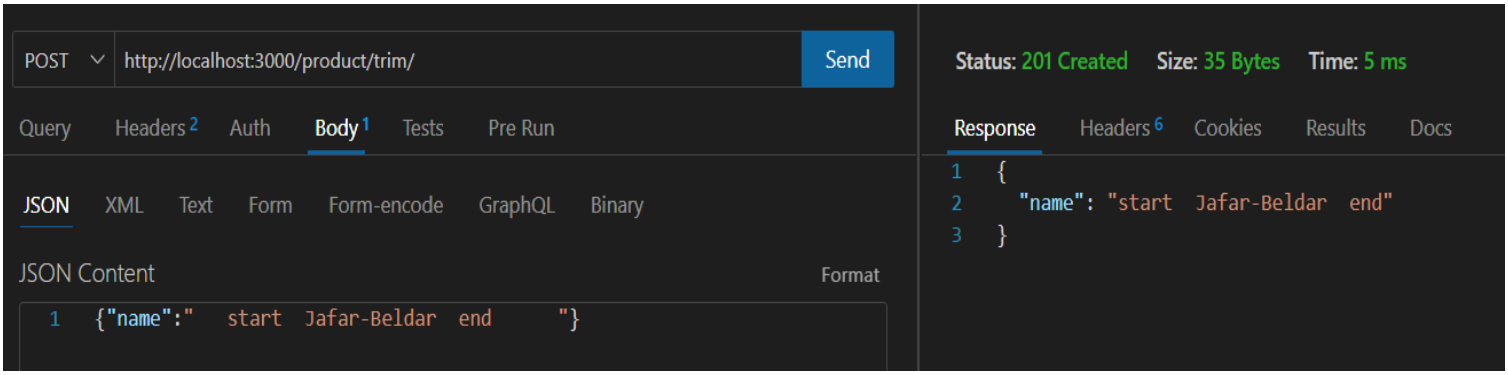
```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9> nest g pipe trim
>>
CREATE src/trim/trim.pipe.ts (228 bytes)
CREATE src/trim/trim.pipe.spec.ts (163 bytes)
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9>
```

src > trim > trim.pipe.ts > TrimPipe > transform

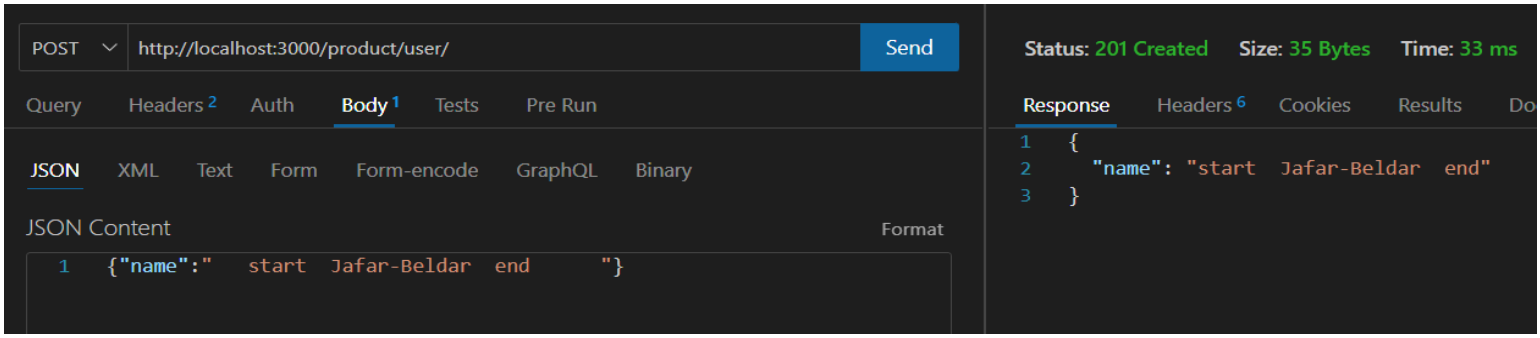
```
import { ArgumentMetadata, Injectable, PipeTransform } from '@nestjs/common';

@Injectable()
export class TrimPipe implements PipeTransform {
  transform(value: string, metadata: ArgumentMetadata) {
    return  value.trim() ;
  }
}
```

```
@Post("/trim")
postName(@Body('name',new TrimPipe()) name:string) {
  console.log('Received:', name);
  return {name};
}
```




## 10. Apply TrimPipe to name field in POST /users:




## 11. Create and use ToUpperCasePipe:

```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9> nest g pipe toUpperCase
CREATE src/to-upper-case/to-upper-case.pipe.ts (235 bytes)
CREATE src/to-upper-case/to-upper-case.pipe.spec.ts (193 bytes)
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9>
```

src > to-upper-case > to-upper-case.pipe.ts > ToUpperCasePipe > transform

```
1  import { ArgumentMetadata, Injectable, PipeTransform } from '@nestjs/common';
2
3  @Injectable()
4  export class ToUpperCasePipe implements PipeTransform {
5    transform(value:string, metadata: ArgumentMetadata) {
6      return value.toUpperCase();
7
8    }
9  }
10
```

```
@Post("/toUpperCase")
convertToUpperCase(@Body('name',new ToUpperCasePipe()) name:string) {
  console.log('Received:', name);
  return {name};
}
```

POST http://localhost:3000/product/toUpperCase/ Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {"name":"   start  Jafar-Beldar  end      "}
```

Status: 201 Created Size: 44 Bytes Time: 35 ms

Response Headers 6 Cookies Results Docs

```
1 {
2   "name": "   START  JAFAR-BELDAR  END      "
3 }
```

## 12.Build a JsonParsePipe that takes a JSON string from a query parameter and converts it into an object:

```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9> nest g pipe jsonParse
CREATE src/json-parse/json-parse.pipe.ts (233 bytes)
CREATE src/json-parse/json-parse.pipe.spec.ts (184 bytes)
```

GET http://localhost:3000/product/parse?name={"first":"Jafar"} Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

Status: 200 OK Size: 17 Bytes Time: 5 ms

Response Headers 6 Cookies Results Docs

```
1 {
2   "first": "Jafar"
3 }
```

```
38  @Get("/parse")
39  parseToJSON(@Query('name',new JsonParsePipe()) name:string) {
40    console.log('Received:', name);
41    return name;
42  }
```

src > json-parse > TS json-parse.pipe.ts > JsonParsePipe > transform

```
import { ArgumentMetadata, Injectable, PipeTransform } from '@nestjs/common';

@Injectable()
export class JsonParsePipe implements PipeTransform {
  transform(value: any, metadata: ArgumentMetadata) {

try{
  return JSON.parse(value);
}catch(err){
  return {"Error While Parsing to JSON":err};
}

  }
}

```

## 13. Write a middleware called RequestLoggerMiddleware that logs the method, URL, and timestamp for each request:

```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9> nest g middleware request-logger
>>
CREATE src/request-logger/request-logger.middleware.ts (213 bytes)
CREATE src/request-logger/request-logger.middleware.spec.ts (224 bytes)
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9>
```

src > TS app.module.ts > AppModule > configure

```
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { ProductModule } from './product/product.module';
import { JsonParsePipe } from './json-parse/json-parse.pipe';
import { RequestLoggerMiddleware } from './request-logger/request-logger.middleware';

@Module({
  imports: [ProductModule],
  controllers: [AppController],
  providers: [AppService,JsonParsePipe
],
})
export class AppModule {
configure(consumer: MiddlewareConsumer) {
  consumer
    .apply(RequestLoggerMiddleware)
    .forRoutes({ path: '*', method: RequestMethod.ALL });
}
}

```

GET http://localhost:3000/product/parse?name="Jafar" Send

Status: 200 OK Size: 5 Bytes Time:

Query Headers 2 Auth Body 1 Tests Pre Run

Response Headers 6 Cookies

1 Jafar

```
{
  'request-type ': 'GET',
  'URL/Path ': '/product/parse?name=%22Jafar%22',
  DateTime: 2025-04-10T12:05:55.771Z
}
Received: Jafar
```

## 14. Apply middleware only to /admin routes:

POST http://localhost:3000/product/admin/toUpperCase Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {"name":"  start  Jafar-Beldar  end     "}
```

Status: 201 Created Size: 44 Bytes Time: 7 ms

Response Headers 6 Cookies Results Docs

```
1 {
2   "name": "  START  JAFAR-BELDAR  END     "
3 }
```

```
{
  'request-type ': 'POST',
  'URL/Path ': '/product/admin/toUpperCase',
  DateTime: 2025-04-10T12:12:48.781Z
}
```

```
50
51    @Post("/admin/toUpperCase")
52    convertToUpperCaseAdmin(@Body('name',new ToUpperCasePipe()) name:string) {
53      console.log('Received:', name);
54      return {name};
55    }
```

## 15. Middleware to check Authorization header:

```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9> nest g middleware auth
CREATE src/auth/auth.middleware.ts (204 bytes)
CREATE src/auth/auth.middleware.spec.ts (187 bytes)
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9>
```

src > auth > TS auth.middleware.ts > AuthMiddleware > use

```
1   import { Injectable, NestMiddleware } from '@nestjs/common';
2
3   @Injectable()
4   export class AuthMiddleware implements NestMiddleware {
5     use(req: any, res: any, next: () => void) {
6       if (!req.headers.authorization) {
7         return res.status(403).send('Forbidden');
8       }
9
10      next();
11    }
12  }
13
```

```
14  })
15  export class AppModule {
16  configure(consumer: MiddlewareConsumer) {
17    consumer
18      .apply(RequestLoggerMiddleware,AuthMiddleware)
19      .forRoutes({ path: '/product/admin/*', method: RequestMethod.ALL });
20  }
21
```

POST http://localhost:3000/product/admin/toUpperCase Send

Query Headers 2 Auth Body 1 Tests Pre Run

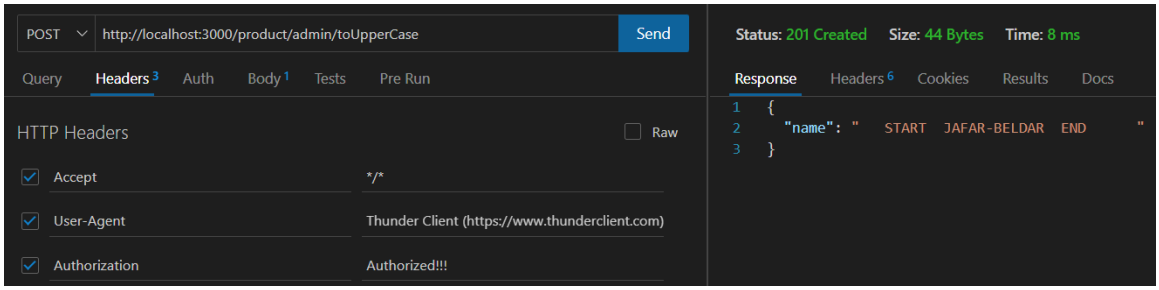
HTTP Headers Raw

| | | |
|---|---|---|
| ☑ | Accept | */* |
| ☑ | User-Agent | Thunder Client (https://www.thunderclient.com) |
| ☐ | header | value |

Status: 403 Forbidden Size: 9 Bytes Time: 35

Response Headers 6 Cookies Results

```
1  Forbidden
```




## 16. Create custom @UserAgent() decorator:

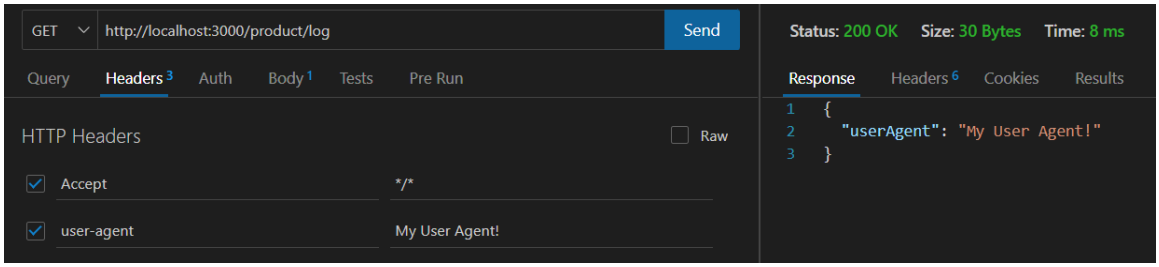
```
src > decorator > TS userAgentDecorator.ts > UserAgent > createParamDecorator() callback
import { createParamDecorator, ExecutionContext } from "@nestjs/common";


export const UserAgent = createParamDecorator(
  (data: unknown, ctx: ExecutionContext) => {
    const request = ctx.switchToHttp().getRequest();
    return request.headers['user-agent'];
  },
);

```

## 17. Use @UserAgent() in controller:

```
    @Get("/log")
logUserAgent(@UserAgent() userAgent: string) {
  console.log('User-Agent:', userAgent);
  return { userAgent };
}
```




## 18. VerifyApiKeyMiddleware (checks x-api-key):

```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9> nest g middleware VerifyApiKeyMiddleware
CREATE src/verify-api-key-middleware/verify-api-key-middleware.middleware.ts (222 bytes)
CREATE src/verify-api-key-middleware/verify-api-key-middleware.middleware.spec.ts (262 bytes)
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9>
```

src > app.module.ts > AppModule

```
import { AppService } from './app.service';
import { ProductModule } from './product/product.module';
import { JsonParsePipe } from './json-parse/json-parse.pipe';
import { RequestLoggerMiddleware } from './request-logger/request-logger.middleware';
import { AuthMiddleware } from './auth/auth.middleware';
import { VerifyApiKeyMiddlewareMiddleware } from './verify-api-key-middleware/verify-api-key-middleware.middleware';
import * as dotenv from 'dotenv';
import { ConfigModule } from '@nestjs/config';

@Module({
  imports: [ProductModule,ConfigModule.forRoot({ isGlobal: true })],
  controllers: [AppController],
  providers: [AppService,JsonParsePipe
],
})
export class AppModule {
configure(consumer: MiddlewareConsumer) {
  consumer
    .apply(RequestLoggerMiddleware,AuthMiddleware,VerifyApiKeyMiddlewareMiddleware)
    .forRoutes({ path: '*', method: RequestMethod.ALL });
}
```

src > verify-api-key-middleware > TS verify-api-key-middleware.middleware.ts > VerifyApiKeyMiddlewareMiddleware > use

```typescript
import { ForbiddenException, Injectable, NestMiddleware } from '@nestjs/common';

@Injectable()
export class VerifyApiKeyMiddlewareMiddleware implements NestMiddleware {
  use(req: any, res: any, next: () => void) {
    const apiKey = req.headers['x-api-key'];
    console.log(`Request IP: ${req.ip}`);

    if (!apiKey || apiKey!==process.env.API_KEY) {
      console.log("Api key is ",apiKey);

      throw new ForbiddenException('Incorrect or missing API key');
    }

    next();
  }
}
```

.env

```
API_KEY=correctKey
```

GET http://localhost:3000/product/log Send

Query | Headers 3 | Auth | Body 1 | Tests | Pre Run

HTTP Headers Raw

| | |
|---|---|
| Accept | */* |
| user-agent | My User Agent! |
| Authorization | Authorized!!! |

Status: 403 Forbidden Size: 79 Bytes Time: 34 ms

Response | Headers 6 | Cookies | Results | Docs

```
{
  "message": "Incorrect or missing API key",
  "error": "Forbidden",
  "statusCode": 403
}
```

GET http://localhost:3000/product/log Send

Query | Headers 4 | Auth | Body 1 | Tests | Pre Run

HTTP Headers Raw

| | |
|---|---|
| Accept | */* |
| user-agent | My User Agent! |
| Authorization | Authorized!!! |
| x-api-key | correctKey |

Status: 200 OK Size: 30 Bytes Time: 9 ms

Response | Headers 6 | Cookies | Results | Do

```
{
  "userAgent": "My User Agent!"
}
```

### Bonus: Log IP address to /reports:

console.log(`Request from IP: ${req.ip}`);

```
{
  'request-type ': 'GET',
  'URL/Path ': '/product/log',
  DateTime: 2025-04-10T12:58:41.934Z
}
Request IP: ::ffff:127.0.0.1
User-Agent: My User Agent!
```

## 19. DTO + pipes for email and age:

```
PS C:\Users\Jafar\Desktop\Nodejs\js files\day9\src> nest g class userDto
CREATE user-dto/user-dto.ts (25 bytes)
CREATE user-dto/user-dto.spec.ts (159 bytes)
```

src > user > user-dto > TS user-dto.ts > ...

```
1
2   import { Transform } from 'class-transformer';
3   import {IsNumber,IsEmail, Min, IsInt} from 'class-validator'
4
5   export class UserDto {
6
7       @Transform(({value})=>value.trim().toLowerCase())
8       @IsEmail()
9       email: string;
10
11      @Transform(({value})=>parseInt(value))
12      @IsInt()
13      age: number;
14
15
16  }
17
```

```
17      @Post()
18      transform(@Body() userdto:UserDto){
19          return userdto;
20      }
21
```

## 20. Use ValidationPipe, TransformUserPipe, and custom header decorator:

```
@Post("/validations")
@UsePipes(TransformUserPipe)
transformUser(@Body() userdto:UserDto,@UserAgent() useragent:string){
    return {userdto,useragent};
}
```

src > transform-user-input > TS transform-user-input.pipe.ts > ...

```
import { PipeTransform, Injectable, BadRequestException } from '@nestjs/common';
import { UserDto } from 'src/user/user-dto/user-dto';


@Injectable()
export class TransformUserPipe implements PipeTransform {
  transform(value: UserDto) {
    const email = value.email?.trim().toLowerCase();
    const age = Number(value.age);

    return value;
  }
}
```

src > user > user-dto > TS user-dto.ts > ...

```
import { Transform } from 'class-transformer';
import {IsNumber,IsEmail, IsInt} from 'class-validator'

export class UserDto {

    @Transform(({value})=>value.trim().toLowerCase())
    @IsEmail()
    email: string;

    @Transform(({value})=>parseInt(value))
    @IsInt()
    age: number;


}
```

src > decorator > TS userAgentDecorator.ts > UserAgent > createParamDecorator() callback

```
1   import { createParamDecorator, ExecutionContext } from "@nestjs/common";
2
3
4   export const UserAgent = createParamDecorator(
5       (data: unknown, ctx: ExecutionContext) => {
6         const request = ctx.switchToHttp().getRequest();
7         return request.headers['user-agent'];
8       },
9     );
10
```

POST http://localhost:3000/user/validations Send

Query Headers 4 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {"email": "example@email.com",
2 "age": "27"
3 }
```

Status: 201 Created Size: 79 Bytes Time: 40 ms

Response Headers 6 Cookies Results Docs

```
1 {
2   "userdto": {
3     "email": "example@email.com",
4     "age": 27
5   },
6   "useragent": "My User Agent!"
7 }
```