

Programowanie Bazodanowe

Celem sprawozdania jest zaprezentowanie efektów pracy na ćwiczeniach z przedmiotu.

Działanie kontrolerów dla metod EF:

1. Dodanie produktu do koszyka:

POST /api/OrderEF

Parameters

Cancel

Name	Description
productId	integer(\$int32) (query)
userId	integer(\$int32) (query)

ExecuteClear

Responses

Curl

```
curl -X 'POST' \
'http://localhost:5011/api/OrderEF?productId=1&userId=1' \
-H 'accept: */*' \
-d ''
```

Request URL

```
http://localhost:5011/api/OrderEF?productId=1&userId=1
```

Server response

CodeDetails

200

Response headers

```
content-length: 0
```

	Id	ProductId	UserId	Amount
▶	1	1	1	0

2. Wygenerowanie zamówienia:

GET /api/OrderEF

Parameters

Cancel

Name	Description
userId	
integer(\$int32)	1
(query)	

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:5011/api/OrderEF?userId=1' \
-H 'accept: text/plain'
```

Request URL

http://localhost:5011/api/OrderEF?userId=1

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 0, "itemDTOs": [{ "product": { "id": 1, "name": "test", "price": 20, "groupName": "test" }, "quantity": 15 }], "total": 300 }</pre>

Download

3. Zaktualizowanie ilości:

PUT /api/OrderEF

Cancel

Parameters

Name	Description
productId	
integer(\$int32)	1
(query)	
basketId	
integer(\$int32)	1
(query)	
quantity	
integer(\$int32)	15
(query)	

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
'http://localhost:5011/api/OrderEF?productId=1&basketId=1&quantity=15' \
-H 'accept: */*'
```

Request URL

http://localhost:5011/api/OrderEF?productId=1&basketId=1&quantity=15

Server response

Code	Details
200	<p>Response headers</p> <pre>content-length: 0</pre>

	Id	ProductId	UserId	Amount
▶	1	1	1	15

4. Usunięcie z koszyka:

DELETE /api/OrderEF

Parameters

Cancel

Name	Description
productId	
integer(\$int32)	1
(query)	
basketId	
integer(\$int32)	1
(query)	

ExecuteClear

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:5011/api/OrderEF?productId=1&basketId=1' \
-H 'accept: */*'
```

Request URL

```
http://localhost:5011/api/OrderEF?productId=1&basketId=1
```

Server response

Code	Details
200	Response headers

	Id	ProductId	UserId	Amount
🗑️	NULL	NULL	NULL	NULL

5. Zapłacenie za zamówienie:

POST /api/OrderEF/pay

Parameters

Cancel

Name	Description
orderId	
integer(\$int32)	1
(query)	
payment	
number(\$double)	300
(query)	

ExecuteClear

Responses

Curl

```
curl -X 'POST' \
'http://localhost:5011/api/OrderEF/pay?orderId=1&payment=300' \
-H 'accept: */*' \
-d ''
```

Request URL

```
http://localhost:5011/api/OrderEF/pay?orderId=1&payment=300
```

Server response

Code	Details
200	Response headers

content-length: 0
date: Tue, 14 May 2024 19:37:24 GMT
server: Kestrel

	Id	UserId	DateTime	isPaid
	1	1	14.05.2024 21:1...	True

6. Dodanie produktu:

POST

/api/ProductsControllerEF

Parameters

Cancel

Name	Description
name string (query)	test
price number(\$double) (query)	20
groupId integer(\$int32) (query)	1

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5011/api/ProductsControllerEF?name=test&price=20&groupId=1' \
  -H 'accept: */*' \
  -d ''
```

Request URL

```
http://localhost:5011/api/ProductsControllerEF?name=test&price=20&groupId=1
```

Server response

Code	Details
200	<div>Response headers</div> <pre>content-length: 0</pre>

7. Wyświetlenie listy produktów:

GET

/api/ProductsControllerEF

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5011/api/ProductsControllerEF' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5011/api/ProductsControllerEF
```

Server response

Code	Details
200	<div>Response body</div> <pre>{ "id": 1, "name": "test", "price": 20, "groupName": "" }</pre> <div>Download</div>

8. Usunięcie produktu:

DELETE

/api/ProductsControllerEF

Cancel

Parameters

Name	Description
productId	
integer(\$int32)	
(query)	
deletePermanently	
boolean	
(query)	

ExecuteClear

Responses

Curl

curl -X 'DELETE' \

'http://localhost:5011/api/ProductsControllerEF?productId=1&deletePermanently=false' \

-H 'accept: */*'

Request URL

http://localhost:5011/api/ProductsControllerEF?productId=1&deletePermanently=false

Server response

Code	Details
200	Response headers

GET

/api/ProductsControllerEF

Cancel

Parameters

No parameters

ExecuteClear

Responses

Curl

curl -X 'GET' \

'http://localhost:5011/api/ProductsControllerEF' \

-H 'accept: text/plain'

Request URL

http://localhost:5011/api/ProductsControllerEF

Server response

Code	Details
200	Response body

[]

Download

9. Reaktywacja produktu:

PUT

/api/ProductsControllerEF

Cancel

Parameters

Name	Description
productId	

integer(int32) 1

(query)

ExecuteClear

Responses

Curl

curl -X 'PUT' \

'http://localhost:5011/api/ProductsControllerEF?productId=1' \

-H 'accept: */*'

Request URL

http://localhost:5011/api/ProductsControllerEF?productId=1

Server response

Code	Details
200	Response headers

GET

/api/ProductsControllerEF

Cancel

Parameters

No parameters

ExecuteClear

Responses

Curl

curl -X 'GET' \

'http://localhost:5011/api/ProductsControllerEF' \

-H 'accept: text/plain'

Request URL

http://localhost:5011/api/ProductsControllerEF

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "name": "test",
  "price": 20,
  "groupName": ""
}
```

Download

Działanie kontrolerów dla metod DB:

1. Dodanie produktu do koszyka:

POST /api/OrderDB

Parameters Cancel

Name	Description
productId integer(\$int32) (query)	12
userId integer(\$int32) (query)	5

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5011/api/OrderDB?productId=12&userId=5' \
  -H 'accept: */*' \
  -d ''
```

Request URL

```
http://localhost:5011/api/OrderDB?productId=12&userId=5
```

Server response

Code	Details
200	Response headers

	Id	ProductId	UserId	Amount
▶	5	12	5	1
⊕	NULL	NULL	NULL	NULL

2. Wygenerowanie zamówienia:

GET /api/OrderDB

Parameters Cancel

Name	Description
userId integer(\$int32) (query)	5

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5011/api/OrderDB?userId=5' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5011/api/OrderDB?userId=5
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "itemDTOs": [
    {
      "product": {
        "id": 12,
        "name": "test",
        "price": 10,
        "groupName": "test"
      },
      "quantity": 1
    }
  ],
  "total": 10
}
```

Download

3. Zaktualizowanie ilości:

PUT

/api/OrderDB

Parameters

Cancel

Name	Description
productId	integer(\$int32)
integer(\$int32)	12
(query)	
basketId	integer(\$int32)
integer(\$int32)	5
(query)	
quantity	integer(\$int32)
integer(\$int32)	13
(query)	

Execute

Clear

Responses

Curl

curl -X 'PUT' \

'http://localhost:5011/api/OrderDB?productId=12&basketId=5&quantity=13' \

-H 'accept: */*'

Request URL

http://localhost:5011/api/OrderDB?productId=12&basketId=5&quantity=13

Server response

Code	Details
200	Response headers
	content-length: 0

	Id	ProductId	UserId	Amount
▶	5	12	5	13
⚙	NULL	NULL	NULL	NULL

4. Usunięcie z koszyka:

DELETE

/api/OrderDB

Parameters

Cancel

Name	Description
productId	integer(\$int32)
integer(\$int32)	12
(query)	
basketId	integer(\$int32)
integer(\$int32)	5
(query)	

Execute

Clear

Responses

Curl

curl -X 'DELETE' \

'http://localhost:5011/api/OrderDB?productId=12&basketId=5' \


-H 'accept: */*'

Request URL

http://localhost:5011/api/OrderDB?productId=12&basketId=5

Server response

Code	Details
200	Response headers
	content-length: 0
	date: Tue, 14 May 2024 19:28:00 GMT
	server: Kestrel

	Id	ProductId	UserId	Amount
	NULL	NULL	NULL	NULL

5. Zapłacenie za zamówienie:

POST /api/OrderDB/pay

Parameters

Cancel

Name	Description
orderId	integer(\$int32)
payment	number(\$double)

1

10

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5011/api/OrderDB/pay?orderId=1&payment=10' \
  -H 'accept: */*' \
  -d ''
```

Request URL

http://localhost:5011/api/OrderDB/pay?orderId=1&payment=10

Server response

Code	Details
200	<div>Response headers</div> <div>content-length: 0</div>

	Id	UserId	DateTime	isPaid
	1	5	14.05.2024 21:1...	True

6. Dodanie produktu:

POST

/api/ProductsDB

Parameters

Name

Description

name

string

(query)

test

price

number(\$double)

(query)

10

groupId

integer(\$int32)

(query)

1

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5011/api/ProductsDB?name=test&price=10&groupId=1' \
  -H 'accept: */*' \
  -d ''
```

Request URL

http://localhost:5011/api/ProductsDB?name=test&price=10&groupId=1

Server response

Code

Details

200

Response headers

content-length: 0

date: Tue, 14 May 2024 19:15:33 GMT

server: Kestrel

7. Wyświetlenie listy produktów:

GET

/api/ProductsDB

Parameters

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5011/api/ProductsDB' \
  -H 'accept: text/plain'
```

Request URL

http://localhost:5011/api/ProductsDB

Server response

Code

Details

200

Response body

```
[
  {
    "id": 12,
    "name": "test",
    "price": 10,
    "groupName": "test"
  }
]
```

Download

8. Usunięcie produktu:

DELETE

/api/ProductsDB

Cancel

Parameters

Name	Description
productId integer(\$int32) (query)	12
deletePermanently boolean (query)	false

Execute

Clear

Responses

Curl

curl -X 'DELETE' \

'http://localhost:5011/api/ProductsDB?productId=12&deletePermanently=false' \

-H 'accept: */*'

Request URL

http://localhost:5011/api/ProductsDB?productId=12&deletePermanently=false

Server response

Code	Details
200	Response headers

GET

/api/ProductsDB

Cancel

Parameters

No parameters

Execute

Clear

Responses

Curl

curl -X 'GET' \

'http://localhost:5011/api/ProductsDB' \

-H 'accept: text/plain'

Request URL

http://localhost:5011/api/ProductsDB

Server response

Code	Details
200	Response body

[]

Download

9. Reaktywacja produktu:

PUT

/api/ProductsDB

Parameters

Cancel

Name	Description
productId	12

Integer(\$int32)
(query)

ExecuteClear

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5011/api/ProductsDB?productId=12' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5011/api/ProductsDB?productId=12
```

Server response

Code	Details
200	<div>Response headers</div> <div>content-length: 0 date: Tue, 14 May 2024 19:17:28 GMT server: Kestrel</div>

GET

/api/ProductsDB

Parameters

Cancel

No parameters

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5011/api/ProductsDB' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5011/api/ProductsDB
```

Server response

Code	Details
200	<div>Response body</div> <div>[{"id": 12, "name": "test", "price": 10, "groupName": "test"}]</div> <div>Download</div>

Porównując obie metody możemy zauważyć że wykorzystanie procedur jest zdecydowanie szybsze od entity frameworka. Podobna zależność zachodzi również w przypadku frameworka Hibernate który jest odpowiednikiem entity frameworka w środowisku Javy. Tam problem wydajności bierze się z faktu, że w celu bycia uogólnionym narzędziem do wszystkich możliwych konfiguracji bazy danych Hibernate nie jest w stanie wysłać do bazy danych zoptymalizowanych query i jest zmuszony wysłać wiele prostych co znacząco spowalnia jego pracę. Tutaj niemalże na pewno mamy do czynienia z podobnym problemem. Jednakże należy wspomnieć, że ze względu na to że procedury musiały być każdorazowo wgrane do bazy danych, a środowisko na którym odbywała się praca w laboratorium resetowało się przy każdym uruchomieniu implementacja ich była uciążliwa. W warunkach normalnych można jednak spodziewać się mniejszej dynamiki sprzętu więc najpewniej problem ten nieistniałby.