



UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU

Wydział Matematyki i Informatyki

Wprowadzenie do Spring, ORM i usług sieciowych REST



Ostatnim razem

Ostatnio:

- omówiliśmy szczegółowo testy jednostkowe
- wstęp do programowania funkcyjnego

Dziś:

- wprowadzenie do Spring
 - CDI
 - adnotacje
 - dostęp do baz danych
 - zahaczymy o REST
-



Wprowadzenie do Spring

Spring Framework:

- modułowa “kobyła” pozwalająca tworzyć aplikacje biznesowe
 - zapewnie CDI (Container Dependency Injection)
 - posiada moduły dostępu do danych
 - niskopoziomowo: JdbcTemplate
 - “klasycznie”: Hibernate
 - wysokopoziomowo: Spring Data JPA
 - zapewnia moduły do komunikacji sieciowej (i nie tylko)
 - pokażemy tylko Spring Web (REST)
-



Wprowadzenie do Spring

Container Dependency Injection:

- wzorzec projektowy “Wstrzykiwanie zależności”
 - cel: modułowość i skalowalność
 - realizacja wzorca Dependency Inversion (“odwrócenie sterowania”), „D” w akronimie SOLID
 - container, bo Spring dostarcza zarządzanie wstrzykiwaniem:
 - tworzenie obiektów
 - i to instancji konkretnych klas, często bez konieczności wskazywania o które nam chodzi
 - niekiedy potrafi nawet sam wygenerować ww. klasy
 - dla ciekawskich: stosowany jest wzorzec projektowy Proxy
-



Wprowadzenie do Spring

Adnotacje powiązane z CDI:

- stereotypy
 - @Component – komponent, de facto prosta klasa
 - @Service – klasa dostarczająca usługi, na ogół dostarczająca metod wysokopoziomowego dostępu do czegoś (lub przetwarzających dane z wielu komponentów, repozytoriów, etc.)
 - @Controller – kontroler usług internetowych
 - @Repository – repozytorium, dostęp do danych – uwaga często potrzeba przekształcenia tego co da repozytorium na “zjadliwy” model obiektowy – stąd mamy @Service
-



Wprowadzenie do Spring

Adnotacje powiązane z CDI:

- wstrzykiwanie zależności:
 - @Autowired – podstawowa adnotacja, pozwala na automatyczne wstrzyknięcie zależności w danym miejscu
 - @Required – wstrzyknięcie zależności do settera

-



ORM – po co to wszystko?

Niezgodność impedancji (ang. impedance mismatch):

- inny model danych w bazach re(we)lacyjnych niż w Javie
 - brak podstawowych struktur danych (listy, mapy, ...)
 - *teoretycznie* istnieją bazy obiektowe, ale... (znacie Państwo jakieś?)
 - obiektowy model zawiera wiele redundancji (ale istnieją mechanizmy rozwiązujące ten problem)
 - relacyjny model danych w założeniu powstał, aby optymalizować zużycie przestrzeni dyskowej (stąd popularność tzw. baz NoSQL, tj. baz nie-relacyjnych optymalizowanych raczej pod szybkość przetwarzania).
-



ORM – o co chodzi

ORM (Object-Relational Mapping, mapowanie relacyjno-obiektowe):

- mapuje klasy obiektowe na reprezentację w bazie danych
 - robi to teoretycznie automatycznie (po to, żeby nie trzeba było niskopoziomowo konwertować danych)
 - teoretycznie samo generuje potrzebny kod SQL
 - może tworzyć schemat bazy danych (nie polecam)
 - tworzy zapytania, aktualizacje, wstawia rekordy, może je także usuwać (i wszystko to często nieefektywnie)
 - teoretycznie wystarcza do zamodelowania struktury obiektowej
-



ORM – podstawowe pojęcia

Pojęcia powiązane z ORM:

- DAO (Data Access Object):
 - klasa dostępowa do danych implementująca wywołania konkretnych zapytań (i zwracająca dane)
 - w świecie Spring repozytorium (repository)
 - Encja (ang. Entity) – klasa implementująca mapowanie obiektu na konkretną tabelę w bazie
 - DTO (Data Transfer Object):
 - klasa przechodnia mapująca encje do sensownego modelu obiektowego (i z powrotem)
 - możemy ich też używać do transferu danych przez sieć
-



Spring Data JPA

Literatura:

[Spring Data JPA documentation](#)

[Accessing Data with JPA Tutorial](#)

Raczej nie chcemy (jako projekt akademicki OK, ale...):

[Accessing JPA Data with REST Tutorial](#)

Co to takiego:

- JPA: Java Persistence API – standard dostępu do danych
 - Spring Data JPA – moduł dostępu do danych Springa
 - Spring ORM – to co faktycznie implementuje mapowanie
 - Hibernate – to co (niestety) siedzi pod spodem Spring ORM
-



Wprowadzenie do Spring

Adnotacje JPA:

- wstrzykiwanie zależności:
 - @Autowired – podstawowa adnotacja, pozwala na automatyczne wstrzyknięcie zależności w danym miejscu
 - @Required – wstrzyknięcie zależności do settera
-



Przykład – Spring Data JPA

```
x 05-06 – PolicyService.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
05-06 src > main > java > pl > edu > amu > demo > life > service > PolicyService > countPolicies
UnicodeTest
PolicyController.java application.properties PolicyDTO.java PolicyStatus.java PersonDTO.java PolicyService.java PersonRepository.java PolicyRepository.java
Project
19
20 @Service
21 @Slf4j
22 public class PolicyService {
23
24     private final PolicyRepository policyRepository;
25     private final PersonRepository personRepository;
26
27
28     @Autowired
29     public PolicyService(PolicyRepository policyRepository, PersonRepository personRepository) {
30         this.policyRepository = policyRepository;
31         this.personRepository = personRepository;
32     }
33
34     public long countPolicies() {
35         log.info("Repository class: {}", policyRepository.getClass().getCanonicalName());
36         var policies:Iterable<Policy> = policyRepository.findAll();
37         var policiesCount:long = StreamSupport.stream(policies.spliterator(), parallel:false).count();
38         log.info("{} policies found in the repository.", policiesCount);
39         return policiesCount;
40     }
41
42     // Don't do that!!!
43     // NEVER, EVER YOU SHOULD QUERY FOR ALL!!!!
44     // Just an example!
45     public List<PolicyDTO> getAllPolicies() {
46         return StreamSupport.stream(policyRepository.findAll().spliterator(), parallel:false) Stream<Policy>
47             .map(PolicyDTO::fromPolicy) Stream<PolicyDTO>
48             .collect(toList());
49     }
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
Structure Persistence Bookmarks Maven Database Notifications
Git Run Debug TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 18:21)
34:66 LF UTF-8 4 spaces master
```



Przykład – Spring Data JPA

```
05-06 - PolicyController.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
05-06 src > main > java > pl > edu > amu > demo > life > web > rest > PolicyController > putNewPolicy
PolicyController.java x UnicodeTest.java x PolicyDTO.java x PolicyStatus.java x PersonDTO.java x PolicyService.java x PolicyRepository.java x pom.xml (demo-jpa) x
10 @RestController
11 public class PolicyController {
12
13     final PolicyService service;
14
15     @Autowired
16     public PolicyController(PolicyService service) {
17         this.service = service;
18     }
19
20     @GetMapping(path = "/policies/count")
21     public long getCount() {
22         return service.countPolicies();
23     }
24
25     @GetMapping(path = "/policies/list")
26     public List<PolicyDTO> getAllPolicies() {
27         // AGAIN, DO NOT DO THAT IN REAL LIFE!!!
28         // YOU ALWAYS NEED WHERE CLAUSE WHILE QUERYING A DATABASE!!!
29         return service.getAllPolicies();
30     }
31
32     @PostMapping(path = "/policies/add")
33     public void putNewPolicy(@RequestBody PolicyDTO policyDTO) {
34         service.savePolicy(policyDTO);
35     }
36
37 }
38

Structure Persistence Bookmarks Database Notifications
Git Run Debug TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (20 minutes ago) 33:42 LF UTF-8 4 spaces master
```



Przykład – Spring Data JPA

The screenshot shows the Insomnia REST client interface. The top bar indicates a new document is open. The main area displays a PUT request to `http://localhost:8080/policies/add` with a status of `200 OK`, a response time of `192 ms`, and `0 B` of data. The request body is a JSON object:

```
1 {
2   "id": null,
3   "number": "TST-00112233",
4   "insuredSum": 123456.78,
5   "startDate": "2022-01-01",
6   "endDate": "2024-12-31",
7   "insuredPerson": {
8     "id": null,
9     "displayName": "SmurfMaruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14   "status": "ACTIVE",
15   "statusChangeDate": "2022-01-01"
16 }
17
```

The response preview shows "No body returned for response". The interface includes tabs for JSON, Auth, Query, Headers, and Docs, and a "Beautify JSON" button at the bottom.



Przykład – Spring Data JPA

The screenshot shows the Insomnia REST client interface. The top bar indicates the current document is "Insomnia - New Document - New Request". The main area displays a GET request to the URL `http://localhost:8080/policies/list`. The response status is `200 OK` with a response time of `110 ms` and a body size of `263 B`. The response is displayed in JSON format, showing a list of policy objects. The JSON is beautified, and the response body is visible in the Preview pane.

```
1 {
2   "id": null,
3   "number": "TST-00112233",
4   "insuredSum": 123456.78,
5   "startDate": "2022-01-01",
6   "endDate": "2024-12-31",
7   "insuredPerson": {
8     "id": null,
9     "displayName": "SmurfMaruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14  "status": "ACTIVE",
15  "statusChangeDate": "2022-01-01"
16 }
17
```

```
1 [
2  {
3    "id": 1,
4    "number": "TST-00112233",
5    "insuredSum": 123456.78,
6    "startDate": "2022-01-01",
7    "endDate": "2024-12-31",
8    "insuredPerson": {
9      "id": 2,
10     "displayName": "SmurfMaruda",
11     "firstName": "Maruda",
12     "lastName": "Smurf",
13     "clientId": 121
14   },
15   "status": "ACTIVE",
16   "statusChangeDate": "2022-01-01"
17 }
18 ]
```

At the bottom of the interface, there are two panes: "Beautify JSON" and "\$.store.books[*].author".



Przykład – Spring Data JPA

The screenshot shows the Insomnia REST client interface. The top bar indicates the application is in the 'DESIGN' mode. The main area displays a 'POST' request to the endpoint `://localhost:8080/policies/update/1`. The request body is a JSON object with the following structure:

```
1 {
2   "id": null,
3   "number": "TST-00112233",
4   "insuredSum": 123456.78,
5   "startDate": "2022-01-01",
6   "endDate": "2023-12-31",
7   "insuredPerson": {
8     "id": null,
9     "displayName": "Smurf Maruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14  "status": "ACTIVE",
15  "statusChangeDate": "2022-01-01"
16 }
17
```

The response is a `200 OK` status with a response time of `13.1 ms` and `0 B` of data. The response body is empty, indicated by the text 'No body returned for response'. The interface also shows a 'Send' button and a 'Beautify JSON' option at the bottom.



Przykład – Spring Data JPA

The screenshot shows the Insomnia REST client interface. The top bar indicates the current document is 'New Document' and the request method is 'DELETE' for the endpoint '://localhost:8080/person/delete/2'. The response status is '500 Internal Server Error' with a duration of 44.2 ms. The response body is displayed in the 'Preview' tab, showing a JSON object with an error message and a stack trace.

```
1 {
2   "id": null,
3   "number": "TST-00112233",
4   "insuredSum": 123456.78,
5   "startDate": "2022-01-01",
6   "endDate": "2023-12-31",
7   "insuredPerson": {
8     "id": null,
9     "displayName": "Smurf Maruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14  "status": "ACTIVE",
15  "statusChangeDate": "2022-01-01"
16 }
17
```

```
1 {
2   "timestamp": "2022-12-06T18:44:46.106+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "trace":
  "org.springframework.dao.DataIntegrityViolationException: could not execute statement; SQL [n/a]; constraint [\"FKF9QLWFX74HUKBRXL95279PEDJ: PUBLIC.POLICY FOREIGN KEY(INSURED_PERSON_PERSON_ID) REFERENCES PUBLIC.PERSON(PERSON_ID) (CAST(2 AS BIGINT))\"; SQL statement:\ndelete from person where person_id=? [23503-214]]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement\n\tat org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:276)\n\tat org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:233)\n\tat org.springframework.orm.jpa.JpaTransactionManager.doCommit(JpaTransactionManager.java:566)\n\tat org.springframework.transaction.support.AbstractPlatformTransactionManager.processCommit(Abstrac
```



Przykład – Spring Data JPA

Insomnia - New Document (LifInsurance) – add a policy

Application Edit View Window Tools Help

Star 25,174 Insomnia / New Document

DESIGN DEBUG TEST

Setup Git Sync

LifInsurance Cookies

Filter

- GET list policies
- PUT add a policy
- POST update the policy
- GET del maruda

PUT http://localhost:8080/policies/add

Send 500 Internal Server Error 8.81 ms 12.4 KB 2 Minutes Ago

JSON Auth Query Headers 1 Docs

```
1 {
2   "id": null,
3   "number": "TST-00002222",
4   "insuredSum": 123456.78,
5   "startDate": "2019-01-01",
6   "endDate": "2021-12-31",
7   "insuredPerson": {
8     "id": 2,
9     "displayName": "Smurf Maruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14  "status": "EXPIRED",
15  "statusChangeDate": "2022-01-01"
16 }
17
```

Preview Headers 4 Cookies Timeline

```
1 {
2   "timestamp": "2022-12-06T18:55:04.675+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "trace":
  "org.springframework.dao.InvalidDataAccessApiUsageException:
  detached entity passed to persist:
  pl.edu.amu.demo.life.jpa.Person; nested exception is
  org.hibernate.PersistentObjectException: detached entity passed
  to persist: pl.edu.amu.demo.life.jpa.Person\n\tat
  org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertH
  ibernateAccessException(HibernateJpaDialect.java:297)\n\tat
  org.springframework.orm.jpa.vendor.HibernateJpaDialect.translat
  eExceptionIfPossible(HibernateJpaDialect.java:233)\n\tat
  org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.tr
  anslateExceptionIfPossible(AbstractEntityManagerFactoryBean.jav
  a:551)\n\tat
  org.springframework.dao.support.ChainedPersistenceExceptionTran
  slator.translateExceptionIfPossible(ChainedPersistenceException
  Translator.java:61)\n\tat
  org.springframework.dao.support.DataAccessUtils.translateIfNeces
  sary(DataAccessUtils.java:242)\n\tat
  org.springframework.dao.support.PersistenceExceptionTranslation
  Interceptor.invoke(PersistenceExceptionTranslationInterceptor.j
  ava:152)\n\tat
  org.springframework.aop.framework.ReflectiveMethodInvocation.pr
  oceed(ReflectiveMethodInvocation.java:186)\n\tat
  org.springframework.data.jpa.repository.support.CrudMethodMetad
  ataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor.
  invoke(CrudMethodMetadataPostProcessor.java:174)\n\tat
  org.springframework.aop.framework.ReflectiveMethodInvocation.pr
```

Beautify JSON

\$.store.books[*].author



Przykład – Spring Data JPA

```
05-06 - PolicyService.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
05-06 src > main > java > pl > edu > amu > demo > life > service > PolicyService > savePolicy
UnicodeTest
PolicyController.java application.properties PolicyDTO.java PolicyStatus.java PersonDTO.java PolicyService.java PersonRepository.java PolicyRepository.java
49
50 @
51 public void savePolicy(PolicyDTO policyDTO) {
52     policyRepository.save(policyDTO.toPolicy());
53 }
54 @
55 public void updatePolicy(Long policyId, PolicyDTO policyDTO) {
56     Policy policy = policyRepository.findById(policyId)
57         .orElseThrow(() -> new IllegalStateException(String.format("Policy not found: %d", policyId)));
58     Policy updated = policyDTO.toPolicy();
59     if (nonNull(updated.getInsuredSum()) && !policy.getInsuredSum().equals(updated.getInsuredSum())) {
60         policy.setInsuredSum(updated.getInsuredSum());
61     }
62     if (nonNull(updated.getEndDate()) && !policy.getEndDate().equals(updated.getEndDate())) {
63         if (!updated.getEndDate().isAfter(policy.getStartDate())) {
64             throw new IllegalStateException("End date must be after start date!");
65         }
66         policy.setEndDate(updated.getEndDate());
67     }
68     if (nonNull(updated.getStatus()) && !policy.getStatus().equals(updated.getStatus())) {
69         policy.setStatus(updated.getStatus());
70         policy.setStatusChangeDate(LocalDate.now());
71     }
72     policyRepository.save(policy);
73 }
74 @
75 public void updatePerson(PersonDTO personDTO) { personRepository.save(personDTO.toPerson()); }
76
77
78 public void deletePerson(Long personId) {...}
79
80
81
82
83
84
85
Git Run Debug TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 18:21) 51:10 (53 chars, 1 line break) LF UTF-8 4 spaces master
```



Przykład – Spring Data JPA

```
05-06 - PolicyService.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
05-06 / src / main / java / pl / edu / amu / demo / life / service / PolicyService
UnicodeTest
PolicyController.java application.properties PolicyDTO.java PolicyStatus.java PersonDTO.java PolicyService.java PersonRepository.java PolicyRepository.java
47
48 }
49
50 @
51 public void savePolicy(PolicyDTO policyDTO) {
52     Policy policyToAdd = policyDTO.toPolicy();
53     if (nonNull(policyToAdd.getId())) {
54         log.error("Attempt to add policy with id {}", policyToAdd.getId());
55         throw new IllegalStateException("New policies must not have an id!");
56     }
57     if (nonNull(policyToAdd.getInsuredPerson().getId())) {
58         Person person = findPerson(policyToAdd.getInsuredPerson().getId());
59         policyToAdd.setInsuredPerson(person);
60     }
61     policyRepository.save(policyToAdd);
62 }
63
64 private Person findPerson(Long personId) {
65     return personRepository.findById(personId)
66         .orElseThrow(() -> new IllegalStateException(String.format("Person id not found: %d", personId)))
67 }
68 @
69 public void updatePolicy(Long policyId, PolicyDTO policyDTO) {
70     Policy policy = policyRepository.findById(policyId)
71         .orElseThrow(() -> new IllegalStateException(String.format("Policy not found: %d", policyId)));
72     Policy updated = policyDTO.toPolicy();
73     if (nonNull(updated.getInsuredSum()) && !policy.getInsuredSum().equals(updated.getInsuredSum())) {
74         policy.setInsuredSum(updated.getInsuredSum());
75     }
76     if (nonNull(updated.getEndDate()) && !policy.getEndDate().equals(updated.getEndDate())) {
77         if (!updated.getEndDate().isAfter(policy.getStartDate())) {
78             throw new IllegalStateException("End date must be after start date!");
79         }
80     }
81 }
Git Run Debug TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 18:21) 67:1 LF UTF-8 4 spaces master
```



Przykład – Spring Data JPA

The screenshot shows the Insomnia REST client interface. The active document is titled "Lifeline - New Document (Lifeline) - add another policy". The request is a PUT to `http://localhost:8080/policies/add` with a status of `200 OK`, a response time of `22.4 ms`, and `0 B` of data. The response body is a JSON object:

```
1 {
2   "id": null,
3   "number": "TST-00002222",
4   "insuredSum": 123456.78,
5   "startDate": "2019-01-01",
6   "endDate": "2021-12-31",
7   "insuredPerson": {
8     "id": 2,
9     "displayName": "Smurf Maruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14   "status": "EXPIRED",
15   "statusChangeDate": "2022-01-01"
16 }
17
```

The interface also shows a list of endpoints on the left, including "list policies", "add a policy", "add another policy", "update the policy", and "del maruda". The "Preview" tab on the right shows "No body returned for response".



Przykład – Spring Data JPA

Insomnia - New Document (LifInsurance) – list policies

Application Edit View Window Tools Help

Star 25,174 Insomnia / New Document

DESIGN DEBUG TEST

Setup Git Sync

LifInsurance Cookies

GET http://localhost:8080/policies/list

Send 200 OK 17.7 ms 528 B

Just Now

Filter

JSON Auth Query Headers 1 Docs

Preview Headers 3 Cookies Timeline

```
GET list policies
PUT add a policy
PUT add another policy
POST update the policy
GET del maruda
```

```
1 {
2   "id": null,
3   "number": "TST-00112233",
4   "insuredSum": 123456.78,
5   "startDate": "2022-01-01",
6   "endDate": "2023-12-31",
7   "insuredPerson": {
8     "id": null,
9     "displayName": "Smurf Maruda",
10    "firstName": "Maruda",
11    "lastName": "Smurf",
12    "clientId": 121
13  },
14  "status": "ACTIVE",
15  "statusChangeDate": "2022-01-01"
16 }
17
```

```
2 {
3   "id": 1,
4   "number": "TST-00112233",
5   "insuredSum": 123456.78,
6   "startDate": "2022-01-01",
7   "endDate": "2024-12-31",
8   "insuredPerson": {
9     "id": 2,
10    "displayName": "Smurf Maruda",
11    "firstName": "Maruda",
12    "lastName": "Smurf",
13    "clientId": 121
14  },
15  "status": "ACTIVE",
16  "statusChangeDate": "2022-01-01"
17 },
18 {
19   "id": 3,
20   "number": "TST-00002222",
21   "insuredSum": 123456.78,
22   "startDate": "2019-01-01",
23   "endDate": "2021-12-31",
24   "insuredPerson": {
25     "id": 2,
26     "displayName": "Smurf Maruda",
27     "firstName": "Maruda",
28     "lastName": "Smurf",
29     "clientId": 121
30   },
31   "status": "EXPIRED",
32   "statusChangeDate": "2022-01-01"
33 }
```

Beautify JSON

\$.store.books[*].author



Przykład – Spring Data JPA

The screenshot displays the Insomnia REST client interface. The top bar shows the application name "Insomnia - Life Insurance (Lifelsurance) – del maruda" and menu options: Application, Edit, View, Window, Tools, Help. Below the top bar, there are tabs for DESIGN, DEBUG, and TEST, and a "Setup Git Sync" button. The main interface is divided into three sections:

- Left Sidebar:** A list of endpoints for the "LifeInsurance" collection. The selected endpoint is "DEL del maruda". Other endpoints include "GET list policies", "PUT add a policy", "PUT add another policy", and "POST update the policy".
- Center Panel:** Shows a DELETE request to "http://localhost:8080/person/delete/2". Below the URL, there are tabs for "Body", "Auth", "Query", "Headers", and "Docs". The "Body" tab is active, displaying a large bug icon and the text "Enter a URL and send to get a response". Below this, it says "Select a body type from above to send data in the body of a request" and provides a link to "Introduction to Insomnia".
- Right Panel:** A list of keyboard shortcuts for various actions:
 - Send Request: Ctrl + Enter
 - Focus URL: Ctrl + L
 - Edit Cookies: Ctrl + K
 - Show Environment Editor: Ctrl + E
 - Show Keyboard Shortcuts: Ctrl + Shift + /



Przykład – Spring Data JPA

```
05-06 - Policy.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
05-06 / src / main / java / pl / edu / amu / demo / life / jpa / Policy / InsuredPerson UnicodeTest Git:
Project
  application.properties
  PolicyDTO.java
  PolicyStatus.java
  PersonDTO.java
  PolicyService.java
  PersonRepository.java
  PolicyRepository.java
  Policy.java
  14 @ToString
  15 public class Policy {
  16
  17     @Id
  18     @GeneratedValue(strategy = GenerationType.AUTO)
  19     @Column(name = "policy id")
  20     private Long id;
  21
  22     @Column(name = "policy number", unique = true, nullable = false, updatable = false)
  23     private String number;
  24
  25     @Column(name = "policy insured sum", nullable = false, updatable = false)
  26     private BigDecimal insuredSum;
  27
  28     @Column(name = "policy start date", nullable = false, updatable = false)
  29     private LocalDate startDate;
  30
  31     @Column(name = "policy end date", nullable = false)
  32     private LocalDate endDate;
  33
  34     @ManyToOne(cascade = CascadeType.ALL, optional = false)
  35     private Person insuredPerson;
  36
  37     @Column(name = "policy status", nullable = false)
  38     private PolicyStatus status;
  39
  40     @Column(name = "policy")
  41     private LocalDate statusChangeDate;
  42
  43 }
  Structure Persistence Bookmarks
  Git Run Debug TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies
  Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 18:21) 34:59 (16 chars) LF UTF-8 4 spaces P master
```




Przykład – Spring Data JPA

The screenshot displays the Insomnia REST client interface. The top bar shows the application name "Insomnia - Life Insurance (Lifeline) - del maruda" and menu options: "Application", "Edit", "View", "Window", "Tools", "Help". The main interface is divided into several sections:

- Left Panel:** A sidebar with a "Filter" input and a list of endpoints for the "LifeInsurance" collection:
 - GET list policies
 - PUT add a policy
 - PUT add another policy
 - POST update the policy
 - DEL del maruda (selected)
- Request Section:** Shows a "DELETE" request to "http://localhost:8080/person/delete/2". The "Send" button is highlighted in purple. Below the URL, there are tabs for "Body", "Auth", "Query", "Headers", and "Docs".
- Response Section:** Shows a successful response with a status of "200 OK", a response time of "28.3 ms", and "0 B" of data. The "Preview" tab is active, displaying the message "No body returned for response".
- Bottom Panel:** A large area with a blue background and a white bug icon. It contains the text "Enter a URL and send to get a response" and "Select a body type from above to send data in the body of a request". A link "Introduction to Insomnia" is also visible.



Przykład – Spring Data JPA

```
05-06 - Policy.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
05-06 / src / main / java / pl / edu / amu / demo / life / jpa / Policy / InsuredPerson UnicodeTest
Project
  application.properties x PolicyDTO.java x PolicyStatus.java x PersonDTO.java x PolicyService.java x PersonRepository.java x PolicyRepository.java x Policy.java x
14 @ToString
15 public class Policy {
16
17     @Id
18     @GeneratedValue(strategy = GenerationType.AUTO)
19     @Column(name = "policy id")
20     private Long id;
21
22     @Column(name = "policy number", unique = true, nullable = false, updatable = false)
23     private String number;
24
25     @Column(name = "policy insured sum", nullable = false, updatable = false)
26     private BigDecimal insuredSum;
27
28     @Column(name = "policy start date", nullable = false, updatable = false)
29     private LocalDate startDate;
30
31     @Column(name = "policy end date", nullable = false)
32     private LocalDate endDate;
33
34     @ManyToOne(cascade = CascadeType.ALL)
35     private Person insuredPerson;
36
37     @Column(name = "policy status", nullable = false)
38     private PolicyStatus status;
39
40     @Column(name = "policy")
41     private LocalDate statusChangeDate;
42
43 }
Structure Persistence Bookmarks
Maven Database Notifications
Git Run Debug TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 18:21) 34:41 LF UTF-8 4 spaces P master
```



Przykład – Spring Data JPA

Browser address bar: <https://start.spring.io> | 120% | Search

spring initializr

Project

Gradle - Groovy Gradle - Kotlin Java Kotlin Groovy

Maven

Spring Boot

3.0.1 (SNAPSHOT) 3.0.0 2.7.7 (SNAPSHOT) 2.7.6

Project Metadata

Group:

Artifact:

Name:

Description:

Package name:

Packaging: Jar War

Dependencies ADD DEPENDENCIES... CTRL + B

- Lombok** DEVELOPER TOOLS −
Java annotation library which helps to reduce boilerplate code.
- Spring Web** WEB −
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- Spring Data JPA** SQL −
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
- H2 Database** SQL −
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

GENERATE CTRL + ↵ **EXPLORE** CTRL + SPACE **SHARE...**



Dodatek: Zaawansowane funkcje ORM (TODO: zrobię i zaktualizuję)



UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU

Wydział Matematyki i Informatyki

To wszystko na dziś.