RESTful Web Services na platformie Java EE (JAX-RS)

Do realizacji ćwiczenia wymagane jest środowisko NetBeans 12 wraz z serwerem aplikacji Payara i dostarczanym wraz z nim wbudowanym serwerem bazy danych H2. Potrzebne będzie również narzędzie do testowania API dostępnego poprzez protokół HTTP. W opisie ćwiczenia przyjęto, że narzędziem tym będzie Postman.

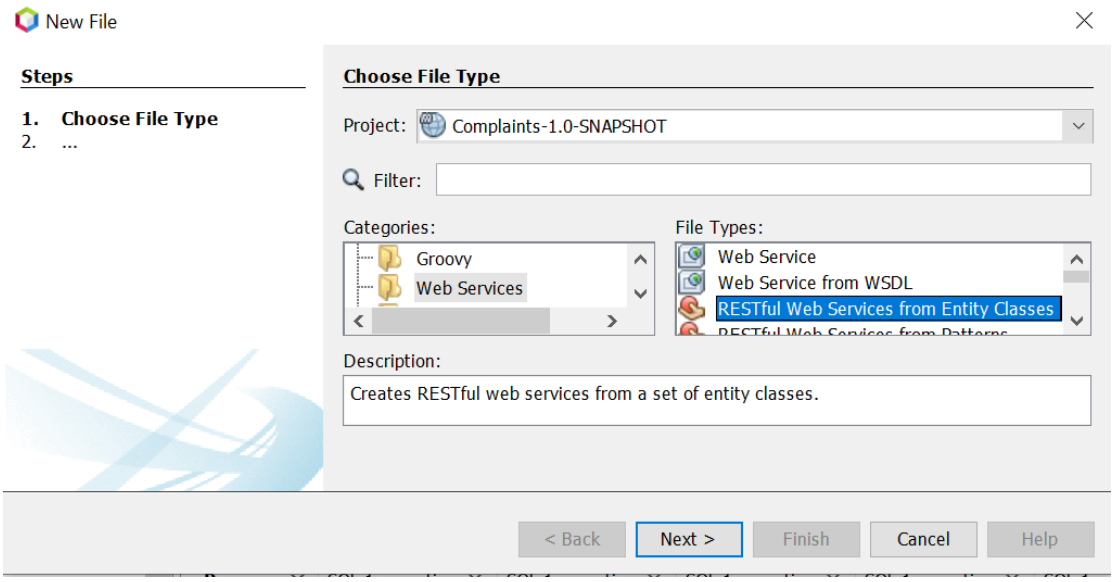
Ćwiczenie 1

Celem ćwiczenia jest przygotowanie usługi sieciowej opartej na klasie Java oznaczonej adnotacjami.

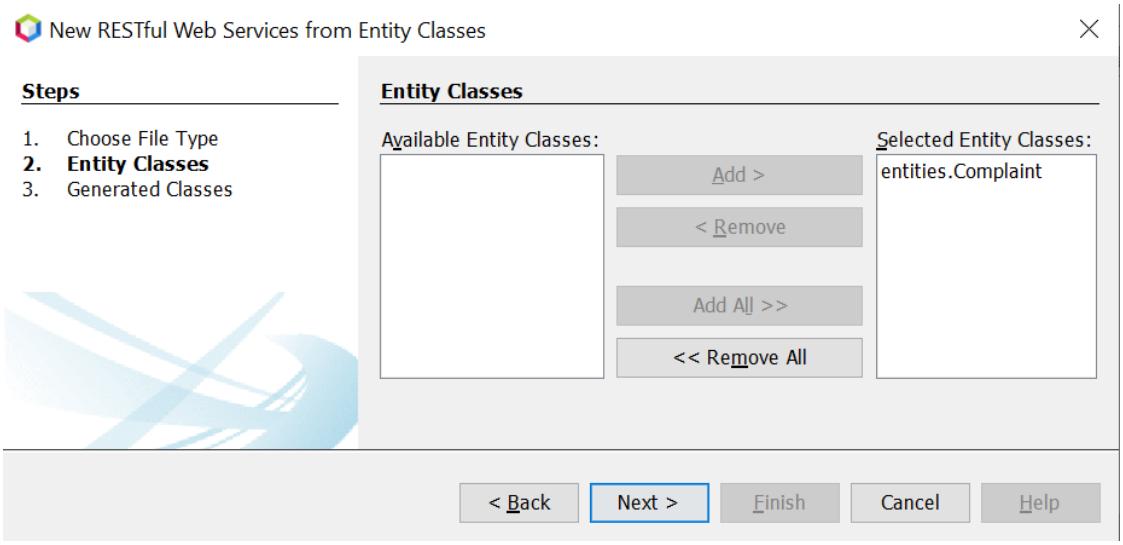
1. Uruchom NetBeans i utwórz nowy projekt opcją File→New Project... W kreatorze projektu z listy kategorii wybierz Java with Maven, a z listy projektów wybierz Web Application. Kliknij przycisk Next >. Jako nazwę projektu podaj **Complaints**, wyczyść pole z nazwą pakietu i kliknij przycisk Next >. Wybierz wersję Java EE Java EE 8 Web i upewnij się, że jako serwer aplikacji wybrany jest serwer Payara (jeśli nie jest dostępny do wyboru, to kliknij Add... i pobierz oraz zainstaluj najnowszą dostępną wersję). Kliknij przycisk Finish.

2. Utwórz klasę encji `Complaint` do reprezentowania skarg.
    a. Kliknij prawym przyciskiem myszy na ikonie projektu i z menu kontekstowego wybierz New → Entity Class.
    b. Jako nazwę klasy podaj `Complaint`, a jako nazwę pakietu `entities`. Pozostaw zaproponowany typ `Long` jako typ identyfikatora encji.
    c. Dodaj w klasie encji następujące prywatne pola: `complaintDate` typu `LocalDate`, `complaintText` typu `String`, `author` typu `String` oraz `status` typu `String`. Dodaj import klasy `java.time.LocalDate`. Wygeneruj publiczne gettery i settery dla dodanych pól (skorzystaj z kreatora Refactor → Encapsulate Fields).
3. Dodaj poniższe ograniczenia dla pól encji `Complaint` za pomocą adnotacji Bean Validation:
    a. @NotNull dla wszystkich 4 dodanych pól
    b. @Size(min = 1, max = 60) dla pól `complaintText` i `author`
    c. @Size(min = 1, max = 6) dla pola `status`

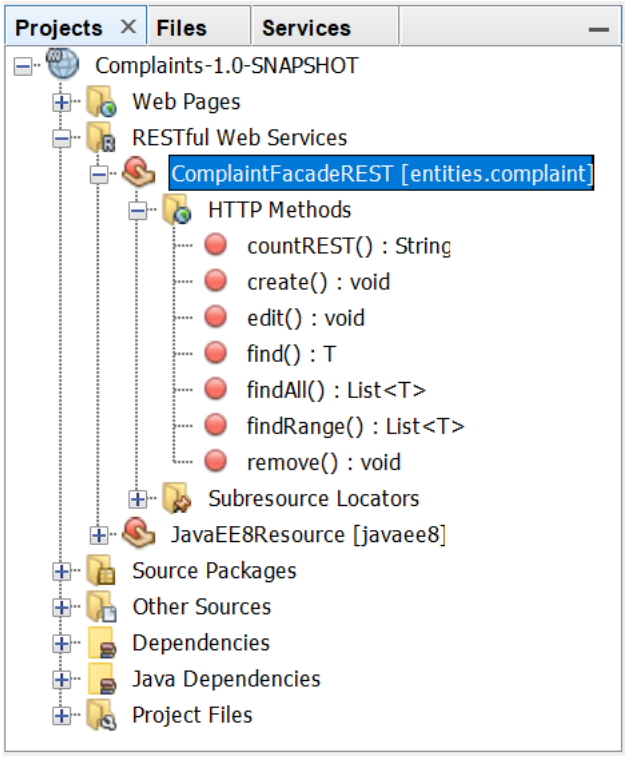
4. Uruchom w projekcie kreator RESTful Web Services from Entity Classes z kategorii Web Services.



5. W kreatorze wybierz klasę encji wygenerowaną w poprzednim kroku ćwiczenia. W ostatnim kroku kreatora pozostaw opcje domyślne.



6. Obejrzyj w strukturze projektu węzeł reprezentujący wygenerowaną usługę REST zwracając uwagę na dostępne operacje. Przejrzyj kod wygenerowanej klasy implementującej usługę zwracając uwagę na adnotacje JAX-RS.



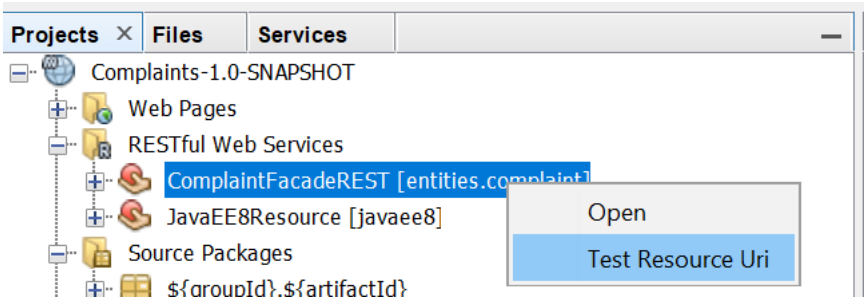
7. Odszukaj w strukturze projektu plik `persistence.xml` (powinien zostać utworzony przez kreator projektu i znajdować się w podkatalogu META-INF). Przejdź do edycji jego źródła otwierając plik, a następnie przełączając edytor na zakładkę Source.

8. Zastąp w kodzie źródłowym pliku `persistence.xml` zawartość elementu <persistence-unit> poniższą wersją, specyfikującą jednostkę trwałości powiązaną ze źródłem danych na serwerze aplikacji i wykorzystującą transakcje w standardzie JTA.

```
    <jta-data-source>jdbc/__default</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property
       name="javax.persistence.schema-generation.database.action"
       value="create"/>
    </properties>
```

9. Uruchom aplikację.

10. Przetestuj usługę wybierając z menu kontekstowego dla niej opcję Test Resource Uri.



Ponieważ w bazie danych nie ma w tej chwili żadnych skarg, do przeglądarki powinna trafić pusta tablica w formacie JSON.

11. Odszukaj w kodzie źródłowym projektu adnotacje odpowiedzialne za elementy ścieżki prowadzącej do zasobu: „resources” i „entities.complaint”.
12. Poprzez modyfikację odpowiedniej adnotacji spraw by zasób Complaints był dostępny pod adresem http://localhost:8080/Complaints/resources/complaints. Ponownie przetestuj usługę aby sprawdzić, że zmiana adresu się powiodła.
13. Z poziomu paska adresu przeglądarki można przetestować odpowiedzi API na żądania GET. Aby przetestować reakcję na inne metody HTTP należy wykorzystać dedykowane do tego celu narzędzia lub samodzielnie zaimplementować aplikacje klienckie. W ćwiczeniu wykorzystamy narzędzie Postman. Uruchom je i na początek przetestuj to same żądanie GET, które sprawdziliśmy w przeglądarce.
14. Przetestuj w narzędziu Postman możliwość tworzenia nowych instancji skarg:
    a. Wprowadź odpowiedni URI
    b. Wybierz metodę POST
    c. Upewnij się, że typem przesyłanej zawartości jest JSON
    d. Jako ciało żądania (w trybie „raw”) wprowadź:

```
{
    "author": "Jim Brown",
    "complaintDate": "2021-04-22",
    "complaintText": "Please check TV in room 242",
    "status": "closed"
}
```

    e. Zwróć uwagę na kod statusu odpowiedzi HTTP. Co on oznacza?
15. Przełącz się na okno przeglądarki i pobierz wszystkie skargi. Powinna zostać pobrana tablica JSON ze skargą dodaną przed chwilą z poziomu narzędzia Postman.
16. Wróć do narzędzia Postman i dodaj poniższe skargi tym samym sposobem co poprzednio:

```
{
    "author": "Marvin Doherty",
    "complaintDate": "2021-04-23",
    "complaintText": "Please fix a tap in room 234",
    "status": "open"
}
```

```
{
    "author": "Arthur McCoy",
    "complaintDate": "2021-04-24",
    "complaintText": "Repair fridge in room 311",
    "status": "open"
}
```

```
{
    "author": "Jim Brown",
    "complaintDate": "2021-04-24",
    "complaintText": "Remove the blood stains on the
wall in room 242",
    "status": "open"
}
```

```
{
    "author": "Johny Bravo",
    "complaintDate": "2021-04-24",
    "complaintText": "Fix air conditioning in room
242",
    "status": "open"
}
```

17. Pobierz wszystkie skargi z poziomu Postmana i przeglądarki.
18. Pobierz z poziomu Postmana i przeglądarki skargę o podanym identyfikatorze. (Identyfikatory zostały nadane automatycznie w bazie danych. Odczytaliśmy je wraz z resztą informacji o skargach w poprzednim punkcie.)
19. Usuń z poziomu Postmana jedną ze skarg o statusie „open”. Zwróć uwagę na kod statusu odpowiedzi HTTP. Następnie pobierz wszystkie skargi aby upewnić się, że usunięcie faktycznie się powiodło.
20. Dodaj możliwość sprawdzenia statusu skargi poprzez adres URI.
    a. Otwórz klasę fasady `ComplaintFacadeREST`.
    b. Dodaj metodę `checkStatus` o następującej treści:

```
public String checkStatus(Long id) {
    return super.find(id).getStatus();
}
```

    c. Metoda ta musi być dostępna przez wywołanie GET. Dodaj odpowiednią adnotację.
    d. Ustaw ścieżkę, pod którą dostępna będzie ta metoda, na „{id}/status”. Ponownie dodaj stosowną adnotację.
    e. Ostatnią adnotacją dla metody zapewnij by status udostępniony był czystym tekstem.

f. Stwórz powiązanie między parametrem „id” w nagłówku metody a polem „{id}” w jej adresie. Wykorzystaj do tego adnotację @PathParam.

g. Uruchom aplikację i przetestuj w przeglądarce odczyt statusu dla kilku skarg.

21. Z poziomu Postmana zaktualizuj jedną ze skarg o statusie „open”, modyfikując coś w jej treści i zmieniając status na „closed”. W tym celu wyślij żądanie metodą PUT pod adres zawierający identyfikator wybranej skargi, podając jako treść żądania odpowiedni JSON. Następnie sprawdź żądaniem GET czy modyfikacja została zrealizowana.

22. Dodaj obsługę filtrowania skarg według statusu:

    a. Oznacz klasę encji przedstawioną poniżej adnotacją `@NamedQuery` definiującą nazwane zapytanie JPQL do wyboru skarg o podanym stutusie.

    ```
    @NamedQuery(name = "Complaint.findByStatus", query =
    "SELECT c FROM Complaint c WHERE c.status = :status"
    )
    ```

    b. W klasie `ComplaintFacadeREST` do metody `findAll` dodaj parametr typu String o nazwie `status`.

    c. Adnotacją `@QueryParam` powiąż go z nazwą parametru query string „status”. Uwaga: Wcześniej wykorzystywaliśmy parametry ścieżkowe. Parametry ścieżkowe są odpowiednie do identyfikacji zasobu. Do filtracji lub sortowania zalecane jest używanie parametrów zawartych w łańcuchu query string. Zwróć uwagę, że dla parametrów typu query nie trzeba zmieniać ścieżki związanej z metodą klasy.

    d. Po dodaniu parametru do metody, przestała ona nadpisywać metodę `findAll` z klasy bazowej. Usuń więc adnotację `@Override`.

    e. Dodaj do metody `findAll` kod, który zwróci dotychczasowy rezultat, jeżeli parametr `status` jest pusty (`null`), a w przeciwnym wypadku zwróci wynik wywołania zapytania nazwanego (`NamedQuery`) „Complaint.findByStatus” przekazując do niego wartość parametru `status`.

```
if (status != null && !"".equals(status))
  return em.createNamedQuery("Complaint.findByStatus")
           .setParameter("status", status)
           .getResultList();
else
  ...
```

23. Uruchom usługę i przetestuj Postmanem działanie filtrowania wg statusu dla zasobu `complaints`.

24. Przetestuj filtrowanie skarg wg statusu bezpośrednio wprowadzając odpowiedni adres w pasku adresu przeglądarki (bez pomocy Postmana).

Ćwiczenie 2

Celem ćwiczenia jest przygotowanie klienta w formie konsolowej aplikacji Java dla usługi REST utworzonej w pierwszym ćwiczeniu.

1. Utwórz w środowisku NetBeans nowy projekt typu Java Application z kategorii Java with Maven. Kliknij przycisk Next >. Jako nazwę projektu podaj **ComplaintsClient**, wyczyść pole z nazwą pakietu i kliknij przycisk Finish.
2. Utwórz w utworzonym przed chwilą projekcie klasę `app.Main`.
3. Dodaj w pliku `pom.xml` projektu klienta bibliotekę Jersey (implementację JAX-RS):

```
    <dependencies>
        <dependency>
            <groupId>org.glassfish.jersey.core</groupId>
            <artifactId>jersey-client</artifactId>
            <version>2.25.1</version>
        </dependency>
        <dependency>
            <groupId>javax.activation</groupId>
            <artifactId>activation</artifactId>
            <version>1.1.1</version>
        </dependency>
    </dependencies>
```

4. Utwórz w klasie `app.Main` metodę `main()` z poniższym kodem:

```
Client client = ClientBuilder.newClient();
String count =
client.target("http://localhost:8080/Complaints/" +
              "resources/complaints/count")
.request(MediaType.TEXT_PLAIN)
.get(String.class);

System.out.println("Count: " + count);

client.close();
```

   Zaimportuj wykorzystywane klasy/interfejsy z pakietów javax.ws.rs.client i javax.ws.rs.core.

5. Uruchom aplikację klienta.
6. Samodzielnie (możesz wzorować się na przykładach np. z Java EE Tutorial) dodaj w metodzie `main()` klienta następujące operacje i przetestuj ich działanie. (Jako format wymiany danych wykorzystaj JSON.)
   a. Pobierz i wyświetl na konsoli wszystkie skargi
   b. Pobierz i wyświetl na konsoli jedną z otwartych skarg (przesyłając jej identyfikator do usługi)
   c. Zmodyfikuj skargę pobraną w poprzednim punkcie zmieniając jej status na zamknięty (poprzez podmianę całego zasobu)
   d. Pobierz i wyświetl na konsoli wszystkie otwarte skargi.