RESTful Web Services

Ćwiczenie dotyczące platformy Java EE zostało przygotowane z myślą o środowisku NetBeans w wersji 7.3 (do pobrania z <u>http://www.netbeans.org/</u>).

Celem ćwiczenia jest przygotowanie usługi sieciowej w oparciu o klasę Java oznaczoną adnotacjami.

- 1. Uruchom środowisko NetBeans.
- 2. Utwórz nowy projekt typu Web Application z kategorii Java Web.
- 3. W kreatorze nowej aplikacji zmień nazwę aplikacji na RestApp, jako serwer wybierz GlassFish, a dla pozostałych opcji pozostaw wartości domyślne.
- 4. Uruchom serwer bazy danych Java DB korzystając z panelu Services.

Projects		Files	Servi	ces √∎ ≋
Databases				
Java DB				1
Drivers		Start Server		
jdbc:de jdbc:ora		Stop Server		app on AP [scott on §
Web Servic		Create Databa	ase	
Servers		Properties		
Hudson Buik	iers			1
Team Serve	rs			
Issue Tracke	ers			

5. Z poziomu panelu **Services** połącz się z bazą danych "sample" na lokalnym serwerze Java DB (derby).

Projects	Files	Services	≈	
🖃 🗐 Databases				
🗄 폙 Java DB				
🕀 🕖 Drivers				
jdbc:derby:	//localhost:1527/sampl	e [app on APP]		
🗄 🛐 jdbc:oracle:	thin:@localhost:1521:	KE [scott on S	Connect	
🕀 🧟 Web Services			Disconnect	
🗄 🛅 Servers				
🗄 🐞 Hudson Builder	S		Execute Comm	and
🕀 🧮 Team Servers				
⊞ 🙀 Issue Trackers			Refresh	
			Delete	Delete
			Properties	

6. Utwórz w bazie danych tabelę, na której działać będzie tworzona aplikacja, realizując poniższe kroki:

- a. Wywołaj okno poleceń SQL dla bazy danych "sample" wybierając z menu kontekstowego dla węzła reprezentującego połączenie opcję Execute Command.
- b. Korzystając z okna poleceń SQL wykonaj kolejno poniższe polecenia SQL (pierwsze może zakończyć się błędem – służy do ewentualnego usunięcia istniejącej już tabeli):

```
DROP TABLE requests;
CREATE TABLE requests
 (id INT NOT NULL GENERATED ALWAYS AS IDENTITY
  CONSTRAINT requests pk PRIMARY KEY,
  request date DATE,
  request text VARCHAR(60) NOT NULL,
  author VARCHAR(60) NOT NULL,
  closed INT DEFAULT 0,
  CHECK(closed BETWEEN 0 AND 1));
INSERT INTO requests (request date, request text, author)
VALUES (CURRENT DATE, 'Please check TV in room 242',
'Jim Brown');
INSERT INTO requests (request date, request text, author)
VALUES (CURRENT DATE, 'Repair fridge in room 311',
'Arthur McCoy');
INSERT INTO requests (request date, request text, author)
VALUES (CURRENT DATE, 'Remove the blood stains on the
wall in room 242', 'Jim Brown');
```

c. Upewnij się odpowiednim zapytaniem, że faktycznie zostały dodane trzy wiersze do tabeli REQUESTS.



7. W utworzonym projekcie uruchom kreator Entity Classes from Database z kategorii Persistence. Wygeneruj klasę encji odpowiadającą tabeli REQUESTS z bazy danych dostępnej poprzez źródło jdbc/sample.

New Entity Classes from Databas	X
Steps 1. Choose File Type 2. Database Tables	Database Tables Data Source: jdbc/sample
Chity Classes Mapping Options	Available Tables: CUSTOMER DISCOUNT_CODE MANUFACTURER MICRO_MARKET PRODUCT PRODUCT_CODE PURCHASE_ORDER SEQUENCE Add All >> < <remove all<="" td=""></remove>
	Any
	< Back Next > Einish Cancel Help

8. Zmień nazwę wygenerowanej klasy na liczbę pojedynczą i ustaw pakiet na "entities". Pozostałe opcje pozostaw domyślne.

New Entity Classes from Database	e				x
Steps	Entity Classe	5			
1. Choose File Type 2. Database Tables	Specify the names and the location of the entity classes.				
3. Entity Classes 4. Manning Ontions	Class Names:	Database Table	Class Name	Generation Type	
		REQUESTS	Request	New	
	Project:	RestOdZera			
	Location:	Source Packages		•	•
	Package:	entities		•	•
	Generate N	amed Query Appotations f	or Persistent Fields		
	Generate <u>J</u>	AXB Annotations			
< <u>B</u> ack Next > Einish Cancel Help					

- 9. Otwórz nowoutworzoną klasę Request. Zweryfikuj:
 - a. Które ograniczenia z bazy danych są wzięte pod uwagę?
 - Długości stringów?
 - Ograniczenie na wartości pola "CLOSED"?
 - b. Czy metody hashCode i equals biorą pod uwagę wszystkie pola?

10. W utworzonym projekcie uruchom kreator RESTful Web Services from Entity Classes z kategorii Web Services. W kreatorze wybierz klasę encji wygenerowaną w poprzednim kroku ćwiczenia. W ostatnim kroku kreatora pozostaw opcje domyślne.

New RESTful Web Services from I Steps	Intity Classes	×
Choose File Type Entity Classes Generated Classes	A <u>v</u> ailable Entity Classes:	Selected Entity Classes: entities.Request
	Add >	re
	Add All >	>
	< <u>B</u> ack Next	> Einish Cancel Help

11. 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.

Projects %	Files	Services	Debugging			
🖃 🛞 RestOdZera						
۱ 🎝 🗄	Web Pages					
😑 🔂 🤅	Source Pack	ages				
	entities					
	🔤 🔂 Req	uest.java				
<u> </u>	service					
	- 🖄 Abst	tractFacade.jav	a			
	🦾 💰 Req	uestFacadeRES	T.java			
🖶 - 🌆 🤉	Test Packag	es				
🖶 🔂 I	libraries					
🖶 - 🍋 T	Test Librarie	s				
🖶 🔥 E	Enterprise B	eans				
📥 - 属 F	RESTful Web	o Services				
🖃 🗞 RequestFacadeREST [entities.request]						
🖶 🐻 HTTP Methods						
🗄 😡 Subresource Locators						
🗄 🖟 👧 Configuration Files						
(MANIFE:	ST.MF				
i (persister	nce.xml				

- 12. Uruchom aplikację.
- 13. Przetestuj usługę wybierając z menu kontekstowego dla niej opcję Test Resource Uri.



- 14. Popraw adres w przeglądarce dodając do niego człon identyfikujący konkretny produkt np. "/1".
- 15. Otwórz plik service.ApplicationConfig i zbadaj jego zawartość. Jakie dwie funkcje spełnia dla Twojej aplikacji?
- 16. Adres URL zasobu jest długi i skomplikowany. Wykonaj zmiany, by zasób Request był dostępny pod adresem http://localhost:8080/RestApp/requests.
- 17. Po stworzeniu klasy encji, kreator dodał pewną ilość NamedQueries. Stwórz interfejs do wykorzystania zapytania Request.findByAuthor.

```
@Entity
@Table(name = "REQUESTS")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Request.findAll",
            query = "SELECT r FROM Request r"),
    @NamedQuery(name = "Request.findById",
            query = "SELECT r FROM Request r WHERE r.id = :id"),
    @NamedQuery(name = "Request.findByRequestDate",
            query = "SELECT r FROM Request r WHERE r.requestDate = :requestDate"),
    @NamedQuery(name = "Request.findByRequestText",
            query = "SELECT r FROM Request r WHERE r.requestText = :requestText"),
    @NamedQuery(name = "Request.findByAuthor",
            query = "SELECT r FROM Request r WHERE r.author = :author"),
    @NamedQuery(name = "Request.findByClosed",
            query = "SELECT r FROM Request r WHERE r.closed = :closed")})
public class Request implements Serializable {
```

- a. Otwórz klasę fasady RequestFacadeREST.
- b. Dodaj metodę findByAuthor o następującej sygnaturze:

```
public List<Request> findByAuthor(String author) {
    return getEntityManager().createNamedQuery(...)
    .setParameter(..., ...).getResultList();
}
```

- c. Metoda ta musi być dostępna przez wywołanie GET. Dodaj odpowiednią adnotację.
- d. Ustaw ścieżkę, pod którą dostępna będzie ta metoda, na "byauthor/{author}". Ponownie dodaj adnotację.
- e. Ostatnią adnotacją dla metody zapewnij, by zasób dostępny był w formatach XML oraz JSON.
- f. Stwórz powiązanie między parametrem "author" w nagłówku metody a polem "{author}" w jej adresie. Wykorzystaj do tego adnotację @PathParam.
- g. Wypełnij treść metody.
- h. Uruchom aplikację i przetestuj działanie dla autora "Jim Brown".
- 18. Stwórz aplikację do testowania usługi. W tym celu stwórz nowy projekt typu WebApplication i nadaj mu nazwę RestAppTest.
- 19. Z menu kontekstowego projektu RestApp wybierz Test RESTful Web Services.



20. W kreatorze wybierz opcję utworzenia testu w istniejącym projekcie.

Configure REST Test Client	x
Locally Generated Test Client (suitable for Internet Explorer)	
Web Test Client in Project Browse	
Target <u>F</u> older:	
Notice:	
The security restrictions of some browsers might block access to remote (REST) resources from the local filesystem. To access REST resources in such cases, deploy a project that contains the test client to the sar domain as the resource. If your browser permits access to remote resources, you can choose your local filesystem as the target and you do not need to deploy the test client to a server.	me
OK Can	icel

21. Wybierz przez Browse... nowoutworzony projekt i zatwierdź.

Select Project	×
Select REST Resource:	
RestAppTest	
(Resupp	
ОК	Cancel

22. By uruchomić stronę do przeprowadzania testów, z menu kontekstowego pliku testresbeans.html wybierz Run File.

🖃 🛞 RestAppTest		
🖨 🔥 Web Pages		
····· 🚺 WEB-INF		
🕀 🕖 images		
📄 app.gif		
📄 cc.gif		
📴 cg.gif		
📄 collapse.gif		
፝ዀ_ css_master-all.css		
📄 expand.gif		
🐻 index.html		
····· 📄 item.gif		
📄 og.gif		
፝ዀ_ test-resbeans.css		
····· 🐻 test-resbeans.htm		
test-resbeans.js	Open	
🖨 🚡 Source Packages	View	
<default package<="" th=""><th></th><th></th></default>		
🖨 🔂 Libraries	Run File	Shift+F6
🕀 🖳 JDK 1.7 (Default)	Debug File	Ctrl+Shift+F5
🗄 🔚 GlassFish Server 4		

23. W oknie przeglądarki otworzyła się strona do testowania. Najpierw sprawdź źródło danych, na podstawie którego została automatycznie wygenerowana. W tym celu kliknij na pole WADL w lewym górnym rogu strony.



- 24. WADL to Web Application Description Language. Pozwala na automatyczne wykrywanie usług sieciowych opartych o http. Dzięki takiemu opisowi można automatycznie wygenerować klienta dla aplikacji REST.
- 25. Jakie metody, wg WADL, dostępne są dla utworzonego zasobu by-author/{author}?
- 26. "Przeklikaj" się przez interfejs. Spróbuj:
 - a. wyświetlić wszystkie zgłoszenia, w postaci XML i JSON,
 - b. wyświetlić zgłoszenie o id 3,
 - c. wyświetlić liczbę zgłoszeń,
 - d. wyświetlić zgłoszenia od 2 do 3. Jakie parametry trzeba podać, by faktycznie pokazały się 2 wpisy?
 - e. wyświetl wpisy Arthura McCoy'a.
- 27. Interfejs testowy nie jest doskonały. Spróbuj wykonać "złe" operacje:
 - a. pobierz listę zgłoszeń w formacie HTML
 - b. wyszukaj wg autora nie podając żadnego parametru

c. spróbuj wykorzystać metodę PUT na zasobie "by-author", zarówno podając autora jak i pomijając ten parametr

28. Spróbuj zmienić wpisy:

a. W węźle "request" wybierz metodę POST i wklej poniższy XML:

- b. Zwróć uwagę na kod odpowiedzi. Co on oznacza?
- c. Spróbuj zaktualizować ten wpis, zmieniając numer pokoju na 432. W tym celu wykonaj metodę PUT dla identyfikatora 4, podając w treści odpowiedni XML. Pamiętaj, że jeżeli w treści XMLa nie podasz węzła <id>, utworzony zostanie nowy wpis!
- d. Usuń wpis Arthura McCoya metodą DELETE.
- 29. Dodasz teraz nową metodę, pozwalającą na zamykanie wpisów (ustawianie pola "closed" na 1).
- 30. Utwórz w pliku RequestFacadeREST.java nową metodę:

```
@PUT
@Path("{id}/close")
public void closeRequest(@PathParam("id") Integer id)
{
    Request req = ...
    req.setClosed(1);
    ...
}
```

- 31. Wykorzystując metody klasy bazowej (super...) najpierw odnajdź wpis o zadanym id, a po ustawieniu flagi zapisz go w bazie.
- 32. Uruchom ponownie aplikację.
- 33. W interfejsie testowym odnajdź nowy zasób i wywołaj go dla wpisu o identyfikatorze 1. Pamiętaj, by usunąć treść żądania!

Resource: request/{id}/close (request/{id}/close)		
Choose method to test: PUT v MIME:	application/xml	Add Parameter Test
id: 1		_
Content:		

- 34. Spróbuj zmienić wartość pola "closed" w metodzie closeRequest na 2. Uruchom ponownie usługę i zweryfikuj efekty. W logu serwera GlassFish znajdź wpis "org.apache.derby.client.am.SqlException:..." i zweryfikuj jego treść. Dlaczego żądanie się nie powiodło?
- 35. Za pomocą ograniczeń na poziomie REST można wprowadzić ograniczenie na wartości parametrów. Zmodyfikuj metodę closeRequest tak, by przyjmowała parametr, który ma być ustawiony w polu closed danego obiektu. W tym celu:
 - a. zmodyfikuj ścieżkę, pod którą dostępny jest ten zasób, na "{id}/close/{value}"
 - b. dodaj parametr typu Integer i powiąż go z "value"
 - c. wykorzystaj go w wywołaniu setClosed.
 - d. uruchom usługę
 - e. przetestuj działanie zasobu
 - f. spróbuj podać zarówno poprawną jak i niepoprawną wartość parametru value.
- 36. Dodaj ograniczenie do elementu "{value}" zmieniając odpowiednią adnotację na @Path("{id}/close/{value: [0-1]}")
- 37. Uruchom usługę, przetestuj zasób z poprawną i niepoprawną wartością.
- 38. W ostatnim zadaniu dodasz opcjonaln parametr wywołania do metody. Pozwoli on na wyświetlenie tylko otwartych zadań.
- 39. W pliku RequestFacadeREST, do metody findAll dodaj parametr typu boolean o nazwie onlyOpen.
 - a. za pomocą adnotacji @DefaultValue ustaw jego domyślną wartość na "false"
 - b. adnotacją @QueryParam (uwaga! Zwróć uwagę, że poprzednio wykorzystywaliśmy @PathParam!) powiąż go z elementem "onlyOpen"
 - c. zwróć uwagę, że dla parametrów typu query nie trzeba zmieniać ścieżki
 - d. po dodaniu parametru do metody, przestała ona nadpisywać metodę findAll z klasy bazowej usuń więc adnotację @Override

- 40. Dodaj do metody kod, który:
 - a. zwróci dotychczasowy rezultat, jeżeli parametr onlyOpen ma wartość false.
 - b. w przeciwnym wypadku zwróci wynik wywołania NamedQuery "Request.findByClosed" z parametrem "closed" ustawionym na 0.
- 41. Uruchom usługę i przetestuj działanie zasobu request.



42. Spróbuj wywołać ten zasób z przeglądarki. W jaki sposób podasz parametr "onlyOpen"?