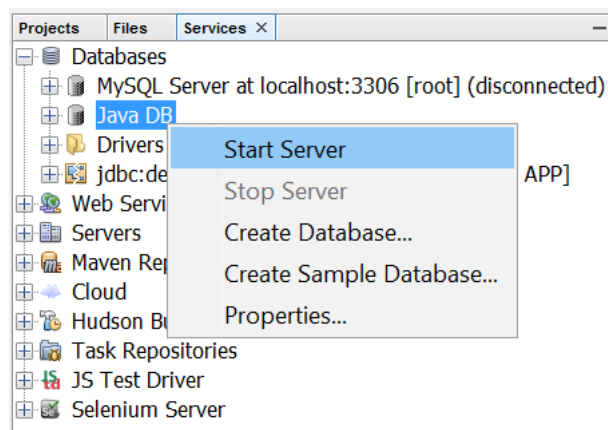


Java EE: JSF + CDI + EJB + JPA + BV

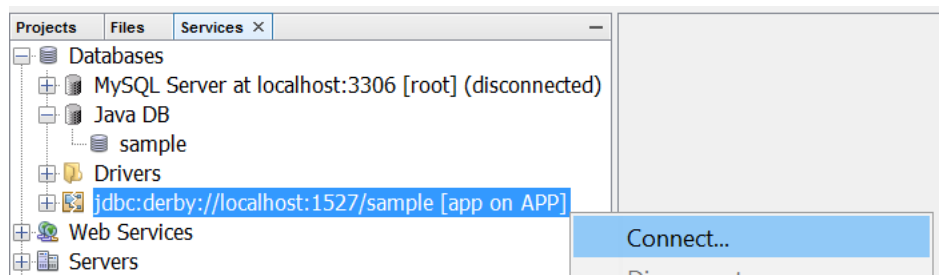
Ćwiczenie 1

Celem ćwiczenia jest utworzenie prostej aplikacji bazodanowej umożliwiającej przeglądanie i dodawanie zleceń serwisowych. Ćwiczenie pokazuje współpracę technologii JSF, CDI, EJB i JPA w ramach platformy Java EE / Jakarta EE. Do realizacji ćwiczenia wymagane jest środowisko NetBeans 11.x wraz z serwerem aplikacji GlassFish i serwerem bazy danych Java DB (dostępny automatycznie wraz z instalacją GlassFisha).

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 **Requests**, 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 GlassFish (jeśli nie jest dostępny do wyboru, to kliknij **Add...** i pobierz oraz zainstaluj najnowszą dostępną wersję). Kliknij przycisk **Finish**.
2. Uruchom serwer bazy danych Java DB korzystając z panelu **Services**.



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



4. 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):

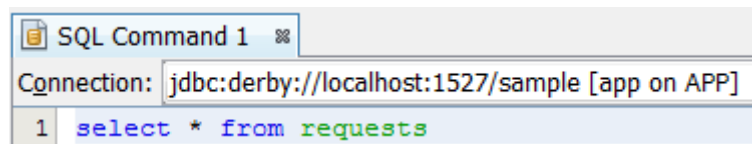
```
DROP TABLE requests;

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);

INSERT INTO requests(request_date, request_text )
VALUES (CURRENT_DATE, 'Please check TV in room 242');

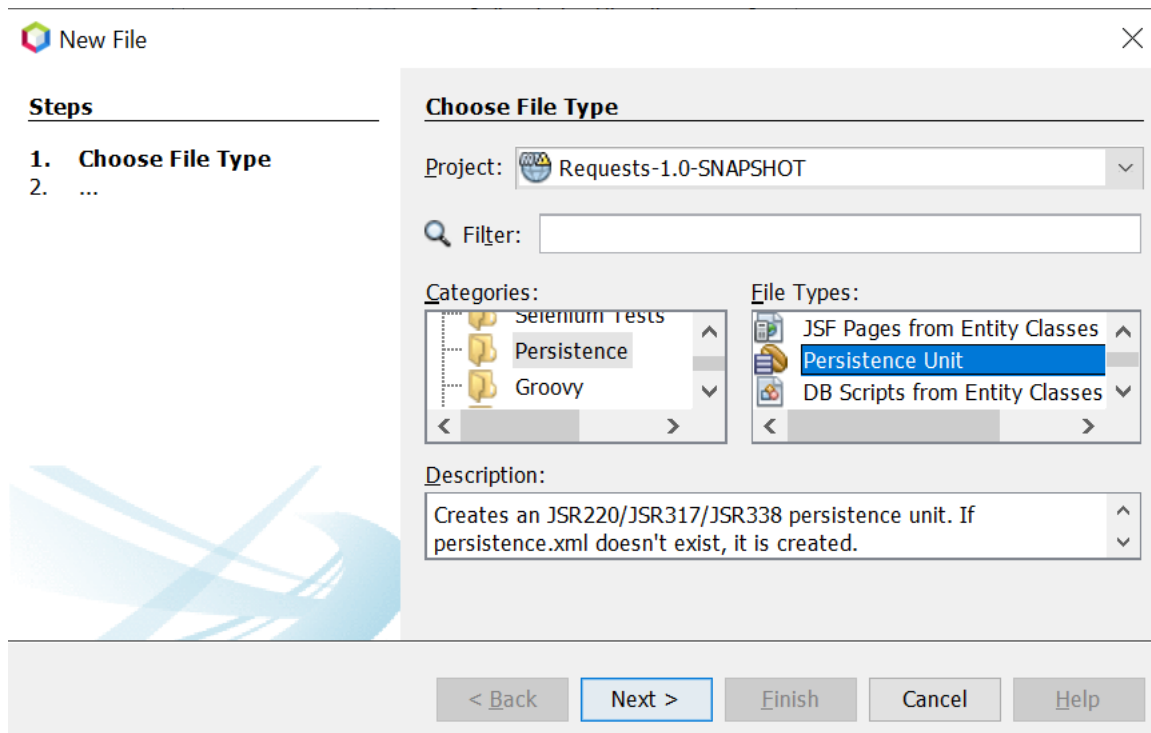
INSERT INTO requests(request_date, request_text )
VALUES (CURRENT_DATE, 'Repair fridge in room 311');
```

- c) Upewnij się odpowiednim zapytaniem, że faktycznie zostały dodane dwa wiersze do tabeli REQUESTS.



5. Utwórz w projekcie jednostkę trwałości, w ramach której obiekty aplikacji będą zachowywane w bazie danych. W tym celu:

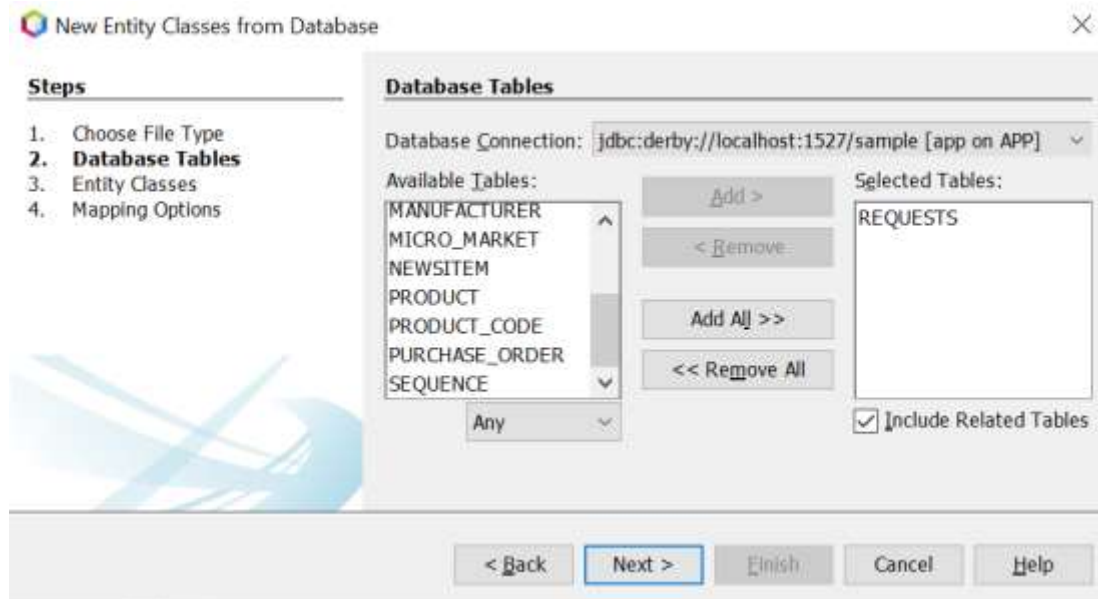
- a) Kliknij lewym przyciskiem myszy na ikonie projektu w panelu projektów i z menu kontekstowego wybierz File → New File. Następnie z kategorii Persistence wybierz typ pliku Persistence Unit. Kliknij przycisk **Next >**.



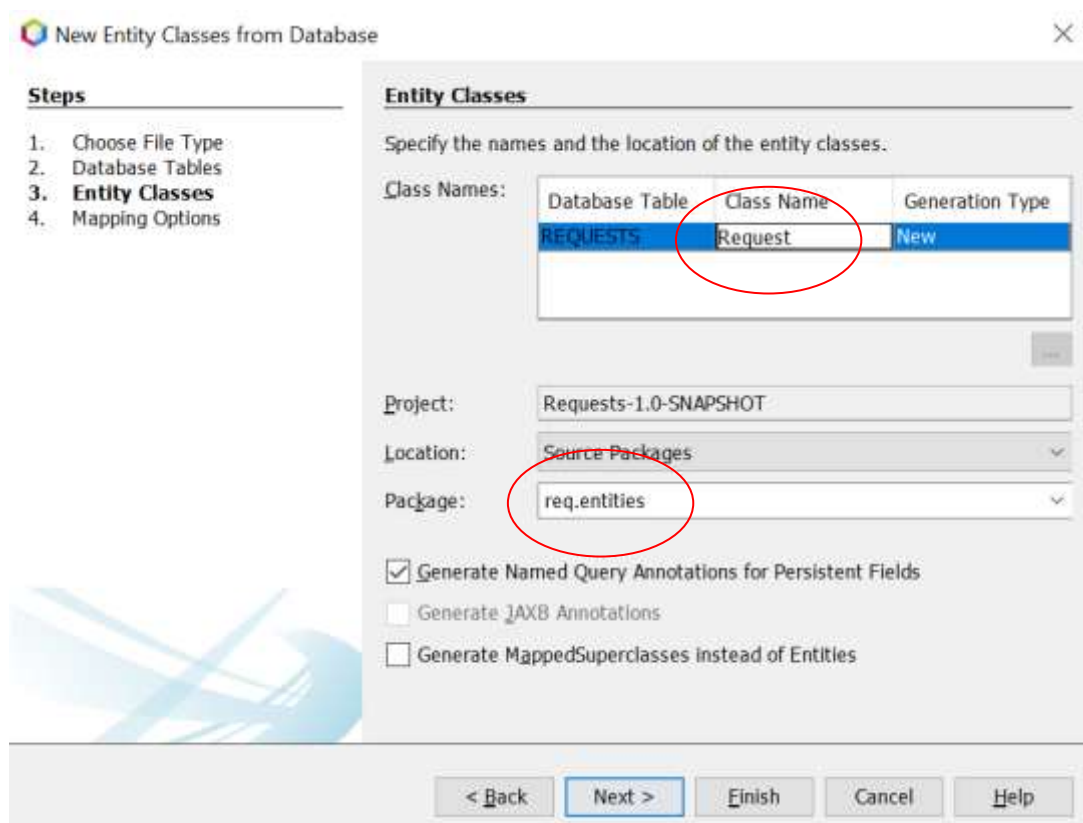
- b) Zmień nazwę jednostki trwałości na **RequestsPU**, a jako połączenie z bazą danych wybierz połączenie do bazy „sample”. Dla pozostałych opcji pozostaw ustawienia domyślne. Niestety w wybranym typie projektu kreator jednostki trwałości nie pozwala na powiązanie jej ze źródłem danych na serwerze aplikacji. Zostanie to poprawione po zakończeniu pracy kreatora. Kliknij przycisk **Finish**.
- c) Przejdź do edycji źródła utworzonego przez kreator pliku `persistence.xml`. przełączając edytor na zakładkę **Source**.
- d) Zastąp w kodzie źródłowym pliku `persistence.xml` cały element `<persistence-unit>` poniższą wersją, specyfikującą jednostkę trwałości powiązaną ze źródłem danych na serwerze aplikacji i wykorzystującą transakcje w standardzie JTA.

```
<persistence-unit name="RequestsPU" transaction-type="JTA">
  <jta-data-source>jdbc/sample</jta-data-source>
  <exclude-unlisted-classes>false</exclude-unlisted-classes>
  <properties>
    <property
      name="javax.persistence.schema-generation.database.action"
      value="create"/>
  </properties>
</persistence-unit>
```

6. Utworzenie za pomocą kreatora zbioru encji na podstawie istniejących tabel w bazie danych.
 - a) Kliknij prawym przyciskiem myszy na ikonie projektu i z menu kontekstowego wybierz **New → Entity Classes from Database**.
 - b) Z listy dostępnych połączeń wybierz połączenie z bazą „sample”, na liście tabel do wyboru w lewym panelu zaznacz myszą tabelę **REQUESTS** i kliknij przycisk **Add >**. Następnie kliknij przycisk **Next >**.



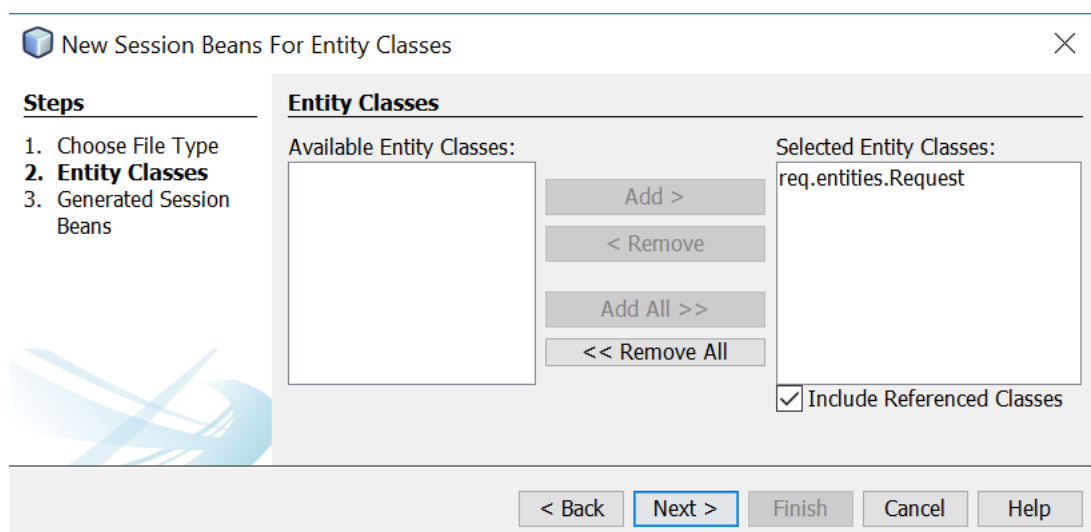
- c) Jako nazwę pakietu podaj `req.entities`. Zmień zaproponowaną nazwę klasy encji na liczbę pojedynczą (`Request`). W pozostałych polach pozostaw wartości domyślne. Kliknij przycisk **Next**.



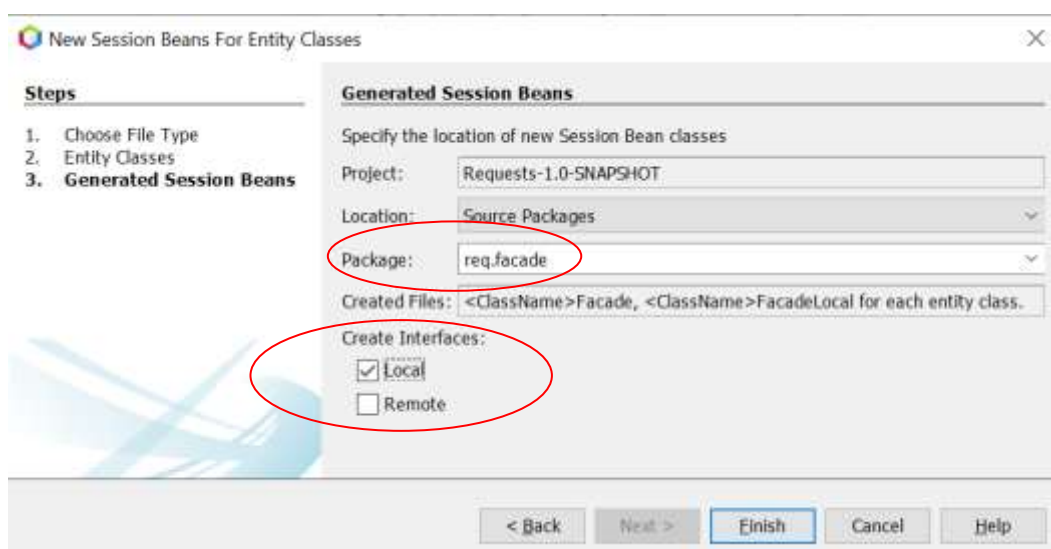
- d) W ostatnim kroku kreatora pozostaw wartości domyślne dla szczegółowych ustawień odwzorowania. Kliknij przycisk **Finish**.

Obejrzyj kod wygenerowanej klasy `Request`. Zwróć uwagę na adnotacje JPA.

7. Sprawdź jakiego typu danych użył kreator do reprezentacji daty zlecenia w klasie encji. Jeśli jest to stara klasa `java.util.Date`, zmień typ właściwości na `java.time.LocalDate`. Nie zapomnij o poprawieniu kodu gettera i settera. Usuń też niepotrzebną po zmianie typu adnotację `@Temporal` dla pola z datą.
8. Utwórz w projekcie nowy bezstanowy komponent sesyjny EJB, który będzie pełnił funkcję fasady (a właściwie „repozytorium” jako szczególnego przypadku „fasady”) udostępniającej operacje na obiektach trwałych. W tym celu:
 - a) Z menu kontekstowego projektu uruchom kreator Session Beans For Entity Classes z kategorii Persistence.
 - b) Wskaż, że tworzony komponent fasadowy ma obsługiwać encję `Request` i kliknij przycisk **Next**.



- c) W ostatnim kroku kreatora zmień proponowany pakiet dla klasy komponentu na `req.facade`. Zaznacz, że komponent ma być dostępny tylko przez interfejs lokalny. Kliknij przycisk **Finish**.



Obejrzyj wygenerowany kod interfejsu i klasy komponentu. Zwróć uwagę, że kreator samodzielnie podjął decyzję, że komponent ma być bezstanowy, co jest typowe dla

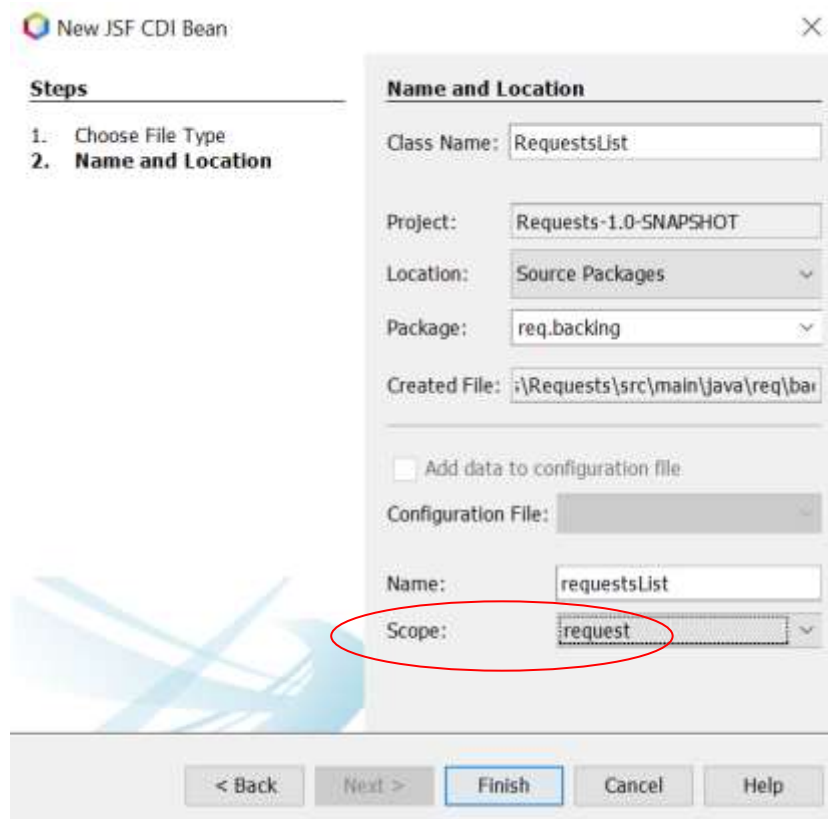
sesyjnych EJB pełniących funkcję fasady. Odszukaj kod realizujący wstrzyknięcie zarządzający encji.

Uwaga: Kreator komponentów fasadowych w NetBeans wykorzystuje abstrakcyjną klasę reprezentującą wspólną dla wszystkich fasad funkcjonalność.

9. Utwórz w projekcie nową stronę JSF (New File→JavaServer Faces/JSF Page). Jako jej nazwę podaj `requestsList`. Zwróć uwagę aby powstała strona ze składnią Facelets (a nie JSP). Rozszerzenie pliku strony `xhtml` zostanie dodane automatycznie.

10. Utwórz komponent backing bean obsługujący stronę JSF. W tym celu:

- a) Utwórz w projekcie nowy komponent CDI (New File→JavaServer Faces/JSF CDI Bean), który będzie pełnił rolę backing bean dla utworzonej przed chwilą strony JSF. W pierwszym kroku kreatora kliknij przycisk **Next >**.
- b) W drugim kroku kreatora podaj nazwę klasy (`RequestsList`), pakiet (`req.backing`), nazwę komponentu (`requestsList` – zgodna z konwencją, powinna ustawić się automatycznie) i jego zasięg (`request`), a następnie kliknij przycisk **Finish**.



11. Przejdź do edycji kodu klasy komponentu backing bean (`RequestsList`). Wprowadź w nim poniższe modyfikacje:

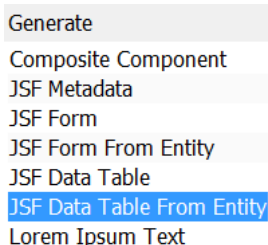
- a) Wstrzyknij referencję do komponentu fasady wstawiając w klasie przed konstruktorem poniższy kod (pamiętaj o niezbędnych importach!).

```
@Inject
private RequestFacadeLocal requestFacade;
```

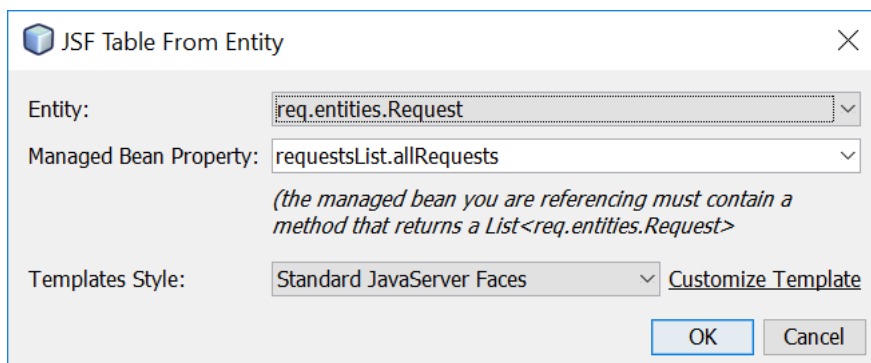
- b) Poniżej konstruktora wklej poniższą metodę, która ma pośredniczyć w dostępie do metody komponentu fasadowego i udostępniać listę wszystkich obiektów encji w formie właściwości komponentu zarządzanego. Samodzielnie uzupełnij kod metody i zaimportuj wykorzystywane klasy (java.util.List oraz naszą klasę encji Request). Zapisz wszystkie zmiany.

```
public List<Request> getAllRequests() {  
    ...  
}
```

12. Przejdź do edycji źródła strony JSF (requestsList.xhtml). Usuń treść zawartą w elemencie <h:body>. Następnie kliknij prawym klawiszem w obszar wewnątrz elementu <h:body> i wybierz z menu kontekstowego opcję Insert Code... Kliknij w wyświetlonym okienku pozycję JSF Data Table From Entity.



W oknie dialogowym które się pojawi po upuszczeniu komponentu na stronie wybierz encję Request i właściwość komponentu backing bean udostępniającą wszystkie instancje encji Request. Naciśnij przycisk OK.

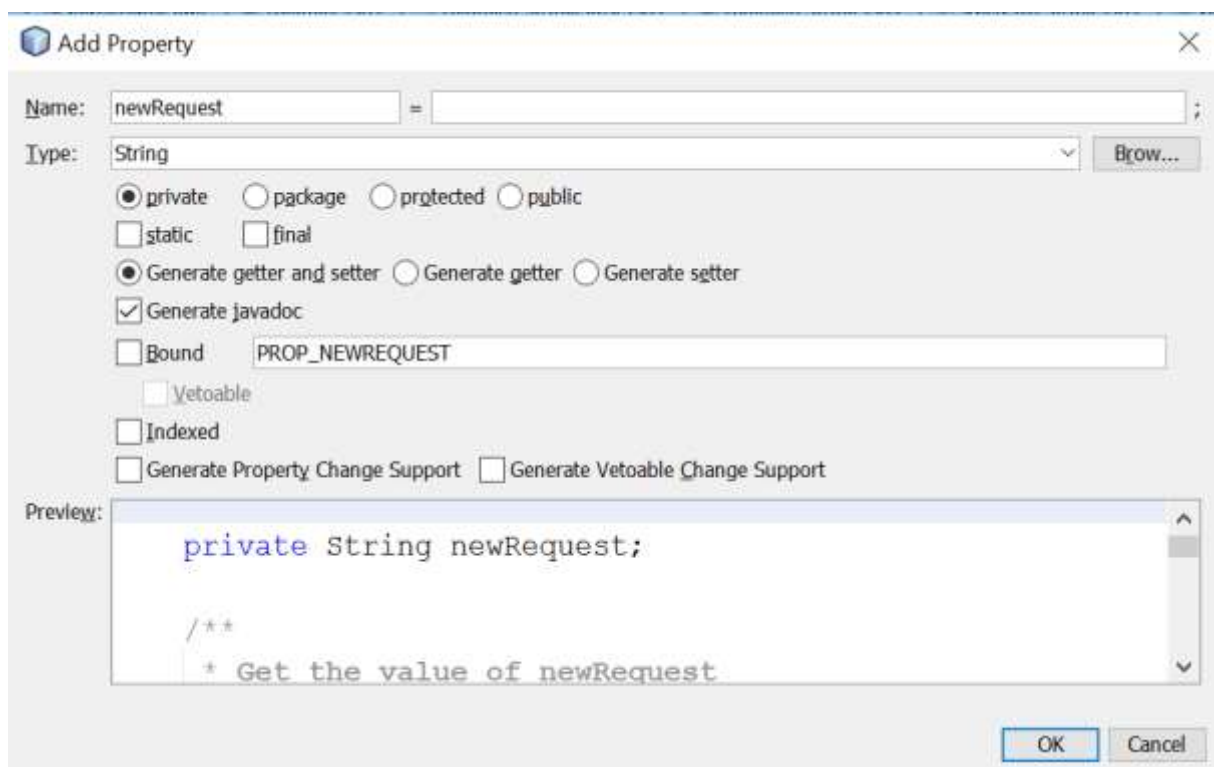


Uwaga: Może się zdarzyć, że nasza encja nie pojawi się na liście encji do wyboru. W takiej sytuacji powinno pomóc przebudowanie projektu, zapisanie zmian, a następnie zamknięcie i ponowne uruchomienie środowiska NetBeans.

13. Obejrzyj kod wygenerowany przez kreator i dokonaj następujących poprawek:

- a) Usuń źle umiejscowione znaczniki <f:view> i </f:view> (pozostawiając zawartość między nimi!)
- b) Dodaj w znaczniku <h:dataTable> atrybut border z wartością 1

- c) Usunąć nagłówek `<h1>` (wraz z zawartością) poprzedzający element `<h:dataTable>`
14. Dodaj do projektu plik konfiguracyjny JSF (`faces-config.xml`) korzystając z kreatora New File → JavaServer Faces/JSF Faces Configuration. Nie będziemy edytować zawartości tego pliku. Celem dodania go jest aktywacja technologii JSF w projekcie, która jest aktualnie wymagana by można było korzystać na stronach JSF z beanów CDI.
 15. Otwórz do edycji plik `web.xml`. Odszukaj w nim wskazanie strony startowej aplikacji i popraw ją na: `/faces/requestsList.xhtml`.
 16. Zapisz wszystkie zmiany i uruchom aplikację. W przeglądarce powinna wyświetlić się strona z tabelką z listą zleceń.
 17. Przejdź do edycji źródła utworzonej strony JSF i wstaw wewnątrz elementu `<h:form>` przed elementem `<h:dataTable>` poniższe trzy elementy (korzystając z podpowiedzi i autouzupełniania edytora tekstowego):
 - element `<h:outputText>` z atrybutem `value` o wartości „New request”
 - element `<h:inputText>` z atrybutem `id` o wartości „newReqInputText”
 - element `<h:commandButton>` z atrybutem `value` o wartości „Add”
 18. Przejdź do edycji kodu klasy komponentu backing bean (`RequestsList`). Wywołaj poprzez menu kontekstowe edytora kreator Insert Code → AddProperty. Dodaj prywatną właściwość `newRequest` typu `String` z publicznymi metodami getter i setter. Zwróć uwagę, aby dodana właściwość nie zaburzyła poprawnej struktury klasy (aby własność nie została wstawiona bezpośrednio po adnotacji).



19. Analogicznie do poprzedniego punktu ćwiczenia dodaj w klasie backing bean właściwość `requestsDataTable` typu `javax.faces.component.html.HtmlDataTable`.

20. Przejdź do edycji strony JSF. Dokonaj powiązania jej komponentów z komponentem backing bean dodając następujące atrybuty znaczników:

- w elemencie `<h:inputText>`:
`value="#{requestsList.newRequest}"`
- w elemencie `<h:commandButton>`:
`action="#{requestsList.addRequest}"`
- w elemencie `<h:dataTable>`:
`binding="#{requestsList.requestsDataTable}"`

21. Dodaj w klasie komponentu backing bean metodę, do której odnosi się przycisk na stronie JSF. Uzupełnij kod metody o operacje:

- utworzenia instancji klasy `Request`
- ustawienia w nowo utworzonym obiekcie daty zlecenia na bieżącą i tekstu zlecenia na wprowadzony do formularza
- utrwalenia obiektu poprzez fasadę

```
public String addRequest()
{
    ...
    setNewRequest("");
    return null;
}
```

22. Zapisz wszystkie zmiany. Uruchom aplikację. Przetestuj dodawanie nowych zleceń serwisowych.

23. Dodaj w klasie komponentu backing bean metodę do usuwania obiektu encji `Request` reprezentowanego przez bieżący wiersz w komponencie `Data Table`. Uzupełnij kod metody o operację usunięcia trwałej reprezentacji obiektu za pomocą fasady.

```
public String deleteRequest() {
    Request req =
        (Request) getRequestsDataTable().getRowData();
    ...
    return null;
}
```

24. Przejdź do edycji strony JSF. Dodaj w komponencie Data Table kolumnę z przyciskami umożliwiającymi usunięcie zlecenia wyświetlanego w bieżącym wierszu (jako ostatnią kolumnę tabeli). Użyj w odpowiedni sposób znacznika `<h:commandButton>` i jego własności `value` i `action`.

Uwaga: W rzeczywistej aplikacji kliknięcie przycisku powinno prowadzić do strony z prośbą o potwierdzenie decyzji o usunięciu zlecenia.

25. Zapisz wszystkie zmiany. Uruchom aplikację i przetestuj ją usuwając kilka zleceń.

Ćwiczenie 2

Celem ćwiczenia jest pokazanie sposobu walidacji danych z wykorzystaniem technologii Bean Validation.

1. Oznacz pole w beanie CDI podpięte pod pole formularza do wprowadzania nowego zlecenia adnotacją `@Size` standardu Bean Validation. Ustaw minimalną długość tekstu na 3 znaki, a maksymalną na 60 znaków.

2. Na stronie JSF z formularzem dodaj komponent `<h:message>` (obok przycisku do dodawania wpisów) do wyświetlania komunikatu o błędzie walidacji dla pola do wprowadzenia nowego zlecenia.

3. Przetestuj dodawanie nowych zleceń zwracając uwagę na standardowy komunikat o błędzie walidacji.

4. Zastąp standardowy komunikat komunikatem „Request text must be from 3 to 60 characters long.” wykorzystując odpowiedni atrybut adnotacji `@Size`.

5. Sprawdź czy nadal poprawnie działa usuwanie zleceń.

6. Spraw aby możliwe było usuwanie zleceń gdy pole z treścią nowego zlecenia jest puste.

7. Zawarcie komunikatu o błędzie w atrybucie adnotacji jest złą praktyką, m.in. z punktu widzenia możliwości internacjonalizacji aplikacji. W celu poprawy tego mankamentu wykonaj poniższe kroki:

a) Utwórz projekcie, w węźle `Other Sources / src/main/resources` plik `properties` korzystając z kreatora `Properties File` z kategorii `Other`. Nazwij plik `ValidationMessages.properties`.

b) Umieść w utworzonym pliku `properties` poniższy wpis klucz=wartość:

```
request.size=Request text must be from {min} to {max} characters long.
```

c) W adnotacji `@Size` w kodzie klasy komponentu backing bean zastąp treść komunikatem jego kodem w nawiasach klamrowych (`{request.size}`).

d) Jeśli masz czas, możesz przetestować internacjonalizację komunikatu o błędzie walidacji przygotowując drugi, zlokalizowany, plik komunikatów.