**Lab3**

3. Jak ustawia się informację o braku wstawienia nowego obiektu typu **Produkt1** w metodzie **utworz\_produkt** w **klasie Fasada\_warstwy\_biznesowej**? W jakiej metodzie jest ta informacja ustalana podczas próby wstawienia produktu o atrybutach wstawionego już wcześniej produktu?

dodaj\_produkt2.xhtml zawiera znacznik <h:commandLink action=”#{managed\_produkt.dodaj\_produkt}” value=”OK”/>  
Po jego naciśnięciu wywoływana jest funkcja ( kolejność poniższych podpunktów jest celowa, a zagnieżdżenia wskazują rzeczywiste zagnieżdżenie wywołań funkcji w kodzie produkcyjnym )

1. Managed\_produkt::dodaj\_produkt()
   1. Fasada\_warstwy\_biznesowej::utworz\_produkt(String dane[])
      1. Fasada\_warstwy\_biznesowej::dodaj\_produkt() w której znajduje się instrukcja warunkowa weryfikująca, czy lista obiektów typu Produkt1 zawiera już nowo wstawiany produkt.  
         Jeśli tak to wartość zmiennej Fasada\_warstwy\_biznesowej::stan ustawiana jest na false.
   2. Managed\_produkt::dane\_produktu()
      1. Jeśli Fasada\_warstwy\_biznesowej::dane\_produktu() == Null
         1. Jeśli Fasada\_warstwy\_biznesowej::stan == false -> Zwróć null
      2. Managed\_produkt::stan = 0;
      3. Zwróć „rezultat2”

Wtedy podczas renderowania strony rezultat2.xhtml serwer natrafia na znaczniki

1. <h:outputText escape="false" value="#{bundle['rezultat2.produkt\_istnieje']}" rendered="#{managed\_produkt.stan==0}"/>
   1. Tutaj przy pomocy gettera używana jest wartość zmiennej Managed\_produkt::stan przez co jeśli jej wartość == 0, to renderowany jest komunikat błędu
2. <h:panelGrid columns="2" rendered="#{managed\_produkt.stan!=0}">
   1. Tutaj przy pomocy gettera używana jest wartość zmiennej Managed\_produkt::stan przez co jeśli jej wartość != 0, to renderowany jest znacznik <h:panelGrid/>

( Chyba trochę za dużo opisałem xD )

4. Jak w metodzie **dane\_produktu()** w klasie **Fasada\_warstwy\_biznesowe**j wykonuje się model ostatnio wstawionego produktu, a jak w przypadku braku wstawionego produktu?

W przypadku wstawionego produktu funkcja pobiera ostatni element listy -> konwertuje wartości jego pól to zmiennych typu String, po czym je zapisuje odpowiednio uporządkowane w postaci tablicy zmiennych typu String.

W przypadku braku wstawionego produktu, zwracana jest wartość Null.

5. Jak obiekt typu **Managed\_produk**t ustala sposób prezentowania informacji o wprowadzonym nowym produkcie:

5.1. w przypadku produktu o wartościach atrybutów, których nie zawiera żaden wcześniej wprowadzony

produkt?

Zmienna Managed\_produkt::stan ma wartość 1

5.2. w przypadku produktu o wartościach atrybutów, które zawiera wcześniej wprowadzony produkt?

Zmienna Managed\_produkt::stan ma wartość 0

8. Jak tworzona jest zawartość komponentu **<h:dataTable>** na stronie **lista\_produktow.xhtml**?

Funkcja Managed\_produkt::getItems() zwraca kolekcję obiektów zapisanych w obiekcie typu DataModel.

Parametr item jest iteratorem po tej kolekcji i dla każdego elementu kolekcji, pobierane jest pole. Np. ID ( dostęp poprzez indeks w tablicy nie poprzez nazwe zmiennej ) i dla niego tworzony jest znacznik <h: column> wraz ze znacznikiem <h: outputText/> którego atrybut value przyjmuje wartość odpowiadająca zmiennej ID.

1. <h:column/>
   1. <f:facet/>
   2. <h:outputText/>

Inicjalizowane są wartościami typu String pochodzących pośrednio z pól LinkedList<Produkt1>.

10. Jaka rolę pełni plik **Bundle.properties** w budowie stron xhtml? Jak należy korzystać z tego pliku?

Plik ten jest miejscem na tzw. „Magic Numbers” tj. unikatowe wartości zmiennych pojawiające się w różnych miejscach kodu programu. Używamy je np. na stronie rezultat2.xhtml w celu zainicjalizowania wartości atrybutów value dla znaczników <h: outputLabel/>.

Przykładowe użycie zmiennej

<h:outputLabel value=#{bundle[‘rezultat2.nazwa’]}/>

gdzie rezultat2.nazwa jest nazwą zmiennej zapisanej w pliku bundle.properties