Projekt komponentenbasierte Softwareentwicklung:

CookieStore

Lukas Völler 620445

Tobias Gels 622452

Hochschule Osnabrück
Fakultät Ingenieurwissenschaften und Informatik
Barbarastr. 16, D-49076 Osnabrück
Tobias.Gels@hs-osnabrueck.de
Lukas.Voeller@ hs-osnabrueck.de

Gliederung

1. verwendete Technologien
2. Aufgabenstellung und initiale Vorgehensweise
3. Definierung der Anforderungen
4. Projektstruktur
5. JPA und Entitätenrelation
6. Verwendung von JSF
7. Verwendung von PrimeFaces
8. Testen der Applikation
9. Festlegung von Fehleingaben
10. Installation
11. Verantwortlichkeiten

Vorwort

In dieser Projektdokumentation wird primär auf die einzelnen Komponenten der beschriebenen Anwendung eingegangen. Dabei werden vereinzelt Probleme, die bei der Entwicklung auftraten kurz beleuchtet.

1. verwendete Technologien

Für die Entwicklung der Anwendung wurde Netbeans in der Version 8.1, sowie als Server der dort eingebundene GlassFish-Server verwendet. Die eigentliche Darstellung geschah über den Browser Firefox

2. Aufgabenstellung und initiale Vorgehensweise

Die Aufgabe war es eine komponentenbasierte Anwendung zu erstellen, die es einem Nutzer ermöglichen soll Produkte zu bestellen und Produkte hinzuzufügen. Die verwendeten Technologien umfassen JSF, CDI, EJB und JPA. Als ersten Schritt wurden die vorliegenden Produkte und deren Eigenschaften genauer definiert. Es soll sich um Kekse handeln, die die eine Bezeichnung haben und von denen nur begrenzt viele in der Anwendung verfügbar sind. Um diese persistent zu speichern ist eine Datenbank notwendig. Desweiteren sollte die Aufgabenstellung insofern erweitert werden, dass die auch die Bestellungen der Kunden gespeichert und verwaltet werden sollen. Am Anfang des Projektes wurde außerdem ein MockUp erstellt, das veranschaulicht wie viele Views und dementsprechende Buttons es geben soll. Dies half bei der Gestaltung der xhtml-Seiten

3. Definierung der Anforderungen

Über die bestehenden Anforderungen hinaus, wurden verschiedene TestCases definiert, die noch einmal genauer auf bestimmte Fuktionalitäten, mögliche Fehleineingaben eines Nutzers und mögliche Probleme bei der Verwendung mit mehreren Nutzern eingehen sollen:

-Bei der Erstellung eines neuen Keks darf die Anzahl und der Name nicht null sein

-Bei der Erstellung eines neuen Keks darf der Name noch nicht vorhanden sein

-Nutzer kann Kekse komplett löschen und hizufügen

-Bei Abschicken der Bestellung wird geprüft ob Produkt und deren Anzahl noch im Store vorhanden sind

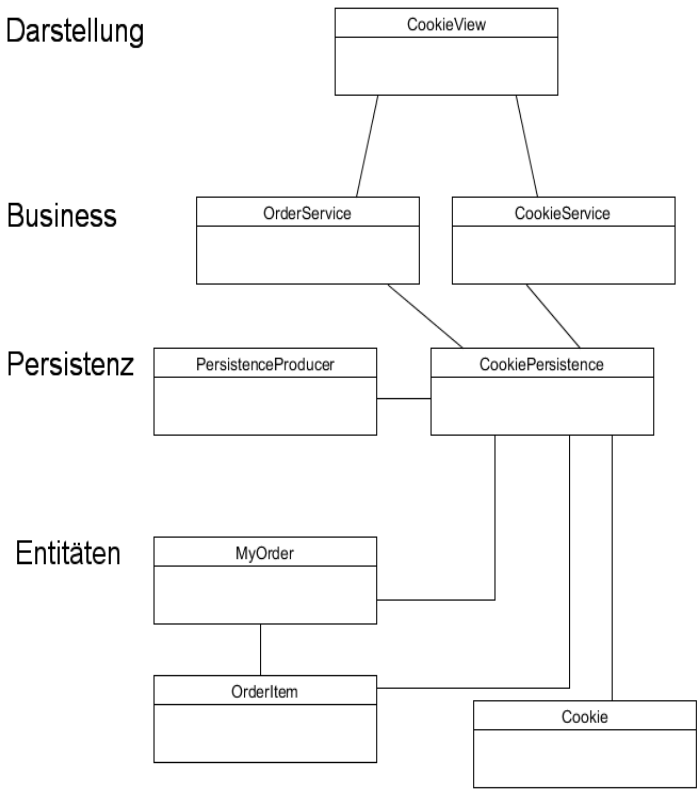
-Bei Abschicken der Bestellung darf der Name des Nutzers nicht null sein

4. Projektstruktur

Die Struktur orientiert sich stark an dem MVC-Pattern. Das heißt es gibt verschiedene Klassen, die getrennt voneineinander jeweils die Datenhaltung, die BusinessLogik und die Darstellung für

den Nutzer behandeln. Dieses Pattern hat den großen Vorteil, das die verwendeten Komponenten wiederverwendet oder auch ausgetauscht werden können. Außerdem bietet es sich für eine parallele Bearbeitung an, da mehrere Personen gleichzeitig an verschiedenen Komponenten arbeiten können.

Auf der untersten Ebene gibt es die Entitäten, die durch auch durch die entsprechende Annotation gekennzeichnet sind. Hier handelt es sich um einfache Java-Objekte(POJO), die nicht viel mehr als Klassenvariabeln und die dazugehörigen Getter und Setter enthalten. Aus den Klassenvariabeln werden dann beim Ausführen SQL-Tabellen und die jeweiligen Spalten erstellt. Eine Ebene darüber befinden sich die Session Bean. Hier werden Funktionen ausgeführt, die direkt mit der Datenbank, also mit den Entitäten in Verbindung stehen. Die verwendete Bean ist zustandslos(Stateless), das heißt sie kann keine übergebenen Variabeln speichern und verfügt so selbst auch über keine Identiät, sondern fungiert als Verbindung zwischen der Datenbank und dereigentlichen Businesslogik. Bei dieser existieren auf Grundlage der verschiedenen Funktionalitäten zwei Klassen. Die erste CookieService kümmert sich um alle logischen Operationen, die mit den Keksen durchgeführt werden, während sich die andere um alle Operationen kümmert die mit den Bestellungen und deren Bestellposten durchgeführt werden. Der genaue Zusammhang der Entitäten ist unter dem Abschnitt 4 erläutert. Bei den Klassen für die Businesslogik kann die Unterscheidung teilweise schwierig sein, da es Methoden gibt die mit mehreren vorkommenden Objekttypen arbeiten. Zum Beispiel wenn eine Liste mit Keksen, die zu einer bestimmten Bestellung gehören zurückgegeben werden soll. Die nächsthöhere Ebene bildet die CookieView, die direkt mit den xHTML-Dateien kommuniziert. Sie hat vor allem die Aufgabe, dass die richtigen Elemente ausgelesen und beschrieben werden und, dass die Navigation korrekt implementiert wird. Auf der obersten Ebene finden sich die xHtml Dateien, die von einem Webbrowser interpretiert und dargestellt werden können. Eine Besonderheit hierbei ist, dass diese Dateien JavaServerFaces nutzen, eine Technologie, die vor allem das Auslesen und korrekte Einfügen in Nutzerschnittstellen implementiert. Zusätzlich wurde hier das Plugin PrimeFaces eingebunden, auf das in Abschnitt 6 eingegangen wird.



5. JPA und Entitätenrelation

In der entwickelten Webanwendung gibt es drei implementierte Persistenz-Entitäten: Cookie, Order und OrderItem. Alle drei haben gemeinsam, dass sie eine, über eine entsprechende Annotation gekennzeichnete, Variable brauchen, die jeder erstellten Instanz dieses Typs eine eindeutige Identität zuweist. Am einfachsten ist es, wie auch hier geschehen, diese automatisch zu generieren. Auf der Datenbankebene bilden jene Variabeln dann den Primärschlüssel der Datenbankzeile. Außerdem verwenden alle Entitäten NamedQueries. Das sind statisch definierte JPQL-Queries, die den Vorteil haben den JPQL-Code vom restlichen Java-Code zu trennen und einen effizienteren Zugang auf die Datenbank zu ermöglichen.

Die Entitäten OrderItem(Bestellposten) und Order(Bestellung) bilden wie der Name schon sagt eine relationale Beziehung. Konkret liegt eine 1-N-Beziehung vor, das heißt n-beliebige Bestellposten gehören zu einer Bestellung. Auf der Datenbankebene heißt das außerdem, dass die ID der Bestellung auch als eigene Spalte bei den Bestellposten auftaucht. Die Beziehung wird durh eine Annotation gekennzeichnet. Dabei ist auf Parent-Seite, also bei MyOrder, noch besonders das noch auf den Variablennamen, den ein Objekt dieser Klasse beim Kind hat verwiesen werden muss. Um bei einem Löschvorgang auch die Kindklasse miteinzubeziehen wird

außerdem das Schlüsselwort orphanRemoval auf true gesetzt. Wichtig ist bei beiden Klassen auch, dass festgelegt werden muss welche JPA-Funktionen kaskadierend Verwendung finden sollen. Falls hierbei nicht PERSIST und MERGE per Annotation aufgeführt werden, kann es später beim Löschen und speichern von Objekten Probleme geben. Zu den Entitätsklassen lässt sich noch sagen, dass hier zu Problemen kommen kann, wenn SQL-spezifische Bezeichner für diese verwendet werden. Dies ist auch der Grund, dass die Klasse myOrder und nicht Order heißt.

In der Klasse CookiePersistence sind Funktionen mit JPA-spezifischen Schlüsselwörtern zu finden. Hierbei gibt es wieder einige Auffälligkeiten, die durch die Beziehung von Order und OrderItem zustande kommen. So ist es wichtig, dass wenn eine Bestellung gelöscht werden soll, es zuersteinmal notwendig ist alle zu dieser Entität gehörenden Unterobjekte zu löschen. Diese müssen dann jeweils einzeln über den remove()-Befehl gelöscht und anschließend noch von der Liste in der Bestellung gelöscht werden. Zu diesem Zweck gibt es in der Bestellung auch Hilfsmethoden, um auf die Liste zuzugreifen.

Bei der Verwendung von JPA wird die Datei persistence.xml benötigt, welche die verwendete Persistenzeinheit und den Namen der Datenbank definiert. Die Persistenzeinheit wird per Injection in die Persistenzklasse eingebunden.

6. Verwendung von JSF

Bei der Webapplikation wurde das JavaServletFaces-Framework verwendet. Dieses unterstützt die Verwendung von bestimmten Annotationen. Um zu definieren über welchen Zeitraum die Applikation Daten verwalten soll, wurde ApplicationScoped verwendet. Das bedeuteteut Daten werden über die komplette Laufzeit verwaltet. In der CookieView-Klasse ist es dann möglich auf die Klassenvariablen zuzugreifen. Dabei ist es wichtig, dass jede Variable einen Getter und Setter besitzt. Dieser sollte generiert sein, um vom Framework interpretiert werden zu können. Sollte dies ist nicht der Fall sein, ist es nämlich nicht möglich mit folgender Schreibweise auf diese zuzugreifen: #{var}. Um Listen mit Daten zu füllen ist es außerdem möglich eine Variable zu

definieren. Auf dessen Objektparameter kann dann in den verschachtelten UI-Elementen zugegriffen werden. Bei der Erstellung der Applikation war dabei auffällig, dass nur das Schlüsselwort „item“ korrekt verarbeitet wird.

7. Verwendung von PrimeFaces

Das Plugin PrimeFaces basiert auf Grundlage von JSF und hat für eine Web-Applikation sowohl aus Design-, als auch aus Funktionssicht Vorteile. Für die Verwendung muss die PrimeFaces Bibliothek in das Projekt eingebunden werden und ein Link zu PrimeFaces muss im Html-Tag integriert werden. Aus Funktionssicht ist vor allem die Einbindung von Ajax zu nennen. Durch diese können bestimmte UI-Elemente zur Laufzeit aktualisiert werden. Aus Designsicht bietet PrimeFaces viele vorgefertigte Elemente, wie zum Beispiel, Buttons, die vom Design auffälliger, als die standardisierten Elemente sind und sich auch einfacher individualisieren lassen. Außerdem gibt es vorgefertigte Menü-Layouts, wie die hier verwendete Möglichkeit, mit Title-Cards zu arbeiten.

8.Testen der Applikation

Um die Applikation zu testen wurden verschiedene kurze Testszenarien definiert, die jeweils eine Funktionalität der Anwendung betrachten. Dabei wurde vor allem auch Wert darauf gelegt, dass die Daten sowohl auf der untersten, also der Datenbank, sowie auf der obersten Schicht, der View, korrekt dargestellt werden können. Somit ist gewährleistet, dass die Daten die komplette Applikation und all ihre Schichten problemlos durchlaufen können. Beim Testen ist es außerdem wichtig, dass immer vom gleichen Ausgangszustand ausgegangen werden kann. Um den zu geährleisten werden beim Starten alle Entitäten gelöscht und es werden einige initiale Cookies erstellt. Die folgenden Tests beschreiben die Aktionen des Nutzers und gehen danach auf den erwarteten Zustand auf der Datenbankschicht ein.

Szenario 1: initialer Start der Applikation

Datenbank: leer
Nutzer öffnet Applikation
Nutzer ist auf index.xhtml und drückt „Zum Cookiestore“
Nutzer ist auf main.xhtml
Nutzer sieht 5 Cookies
Datenbankzustand
Cookie: -name: Schoko, price: 1.99, count: 64 -name: Halbkorn, price: 2.49, count: 64 -name: Osmania, price 2.99, count: 64 -name: Schokomilch, price: 1.49, count: 64 -name: Vollkorn, price: 0.99, count: 64 MyOrder: -customer: $nutzer

Szenario 2: Hinzufügen von Cookie **Datenbank enthält initiale Cookies und Bestellung**
Nutzer ist auf main.xhtml
Nutzer gibt ein: name: Zimtstern, price: 1.99, count: 40
Nutzer drückt „Hinzufügen“
Datenbankzustand
Cookie: -name: Schoko, price: 1.99, count: 64 -name: Halbkorn, price: 2.49, count: 64 -name: Osmania, price 2.99, count: 64 -name: Schokomilch, price: 1.49, count: 64 -name: Vollkorn, price: 0.99, count: 64 -name: Zimtstern, price: 1.99, count: 40 MyOrder: -customer: $nutzer

Szenario 3: Entfernen von Cookie **Datenbank enthält initiale Cookies und Bestellung**
Nutzer ist auf main.xhtml
Nutzer gibt id des dargestellten Cookies Halbkorn ein
Nutzer drückt „Löschen“
Datenbankzustand
Cookie: -name: Schoko, price: 1.99, count: 64 -name: Osmania, price 2.99, count: 64 -name: Schokomilch, price: 1.49, count: 64 -name: Vollkorn, price: 0.99, count: 64 myOrder: -customer: $nutzer

Szenario 4: Bestellen von Cookie Datenbank enthält initiale Cookies und Bestellung
Nutzer ist auf main.xhtml
Nutzer ist auf Feld Schoko und gibt bei Anzahl 16 ein
Nutzer drück Button „Bestellen“ auf Feld Schoko
Datenbankzustand
Cookie: -name: Schoko, price: 1.99, count: 64 -name: Halbkorn, price: 2.49, count: 64 -name: Osmania, price 2.99, count: 64 -name: Schokomilch, price: 1.49, count: 64 -name: Vollkorn, price: 0.99, count: 64 -name: Zimtstern, price: 1.99, count: 40 MyOrder: -customer: $nutzer, OrderItem: -cookieId: $Cookie.Id(name=Schoko), count: 16, status: false , myorder_id: $MyOrder.Id(customer=$nutzer)

Szenario 5: Abschicken von Bestellung Datenbank enthält initiale Cookies und Bestellung
Nutzer ist auf main.xhtml und führt Szenario 4 aus
Nutzer drückt „Bestellung“
Nutzer sieht Bestellung und drückt „Bestellung bestätigen“
Nutzer sieht final.xhtml und drückt „zum Store“
Nutzer ist auf main.xhtml und sieht bei der Anzahl von Schoko nun 48
Datenbankzustand
Cookie: -name: Schoko, price: 1.99, count: 48 -name: Halbkorn, price: 2.49, count: 64 -name: Osmania, price 2.99, count: 64 -name: Schokomilch, price: 1.49, count: 64 -name: Vollkorn, price: 0.99, count: 64 -name: Zimtstern, price: 1.99, count: 40 MyOrder: -customer: $nutzer -customer: $nutzer OrderItem: -cookieId: $Cookie.Id(name=Schoko), count: 16, status: true , myorder_id: $MyOrder.Id(customer=$nutzer)

Szenario 6: Löschen von Bestellposten

Datenbank enthält initiale Cookies und Bestellung
Nutzer ist auf main.xhtml und führt Szenario 4 aus
Nutzer drückt „Bestellung“
Nutzer sieht Bestellung und drückt bei Schoko auf „Löschen“
Nutzer sieht dass seine Bestellung nun leer ist
Datenbankzustand
Cookie: -name: Schoko, price: 1.99, count: 48 -name: Halbkorn, price: 2.49, count: 64 -name: Osmania, price 2.99, count: 64 -name: Schokomilch, price: 1.49, count: 64 -name: Vollkorn, price: 0.99, count: 64 -name: Zimtstern, price: 1.99, count: 40 MyOrder: -customer: $nutzer

9. Festlegung von Fehleingaben

Bei der Interaktion mit dem Nutzer muss das Programm Nutzereingaben abfangen, die zu fehlerhaften Datenbankzuständen führen würden. Für den Nutzer wird als Hinweis eine Textfeld eingeblendet, das ihn auf seinen Fehler aufmerksam macht. Fehlerausgaben sind in der folgenden Liste aufgeführt:

Aktion	Nachricht
Cookie bestellen ohne Anzahl	„Bitte Menge angeben“
Cookie löschen mit falscher ID	„Cookie nicht gefunden“
Cookie hinzufügen ohne Name	„Angaben unvollständig“
Cookie hinzufügen ohne Preis	„Angaben unvollständig“
Cookie hinzufügen ohne Anzahl	„Angaben unvollständig“
Bestellung bestätigen ohne bestellte Cookies	„Bestellung ist leer“
Cookie bestellen mit zu hoher Anzahl	„Vorrat reicht nicht aus“
Cookie wurde während Bestellvorgang gelöscht	„Cookie 01 existiert nicht mehr“
Cookie wurde währen Bestellvorgang von anderer Person bestellt und hat nun zu geringe Anzahl	„Cookie 01 existiert nicht mehr in der Stückzahl“

Auf erfolgreiche Aktionen gibt es folgende mögliche Ausgaben:

Aktion	Nachricht
Cookie erfolgreich hinzugefügt	„Cookie hinzugefügt“
Cookie erfolgreich gelöscht	„Cookie gelöscht“
Cookie erfolgreich zur Bestellung hinzugefügt	„Zur Bestellung hinzugefügt“

Bestellung erfolgreich abgeschlossen	„Bestellung erfolgreich
Bestellposten erfolgreich gelöschr	„Bestellposten gelöscht“

10. Installation

Die Installation weist keine Besonderheiten zu gewöhlichen Java-EE Projekten auf. Die Applikation sollte über Netbeans gestartet werden und benutzt dort den internen GlassFish-Server. Die in den config-Files verwendete Datenbank hat den Namen Voeller_GelsDB mit User: app und Passwort: app.

11. Verantwortlichkeiten

Themenbereich	**Verantwortlicher**
JPA	Tobias Gels
JSF	Tobias Gels
Einbindung Prime Faces	Lukas Völler
Dokumentation	Tobias Gels
Testen	Lukas Völler
Gestaltung/UI	Lukas Völler