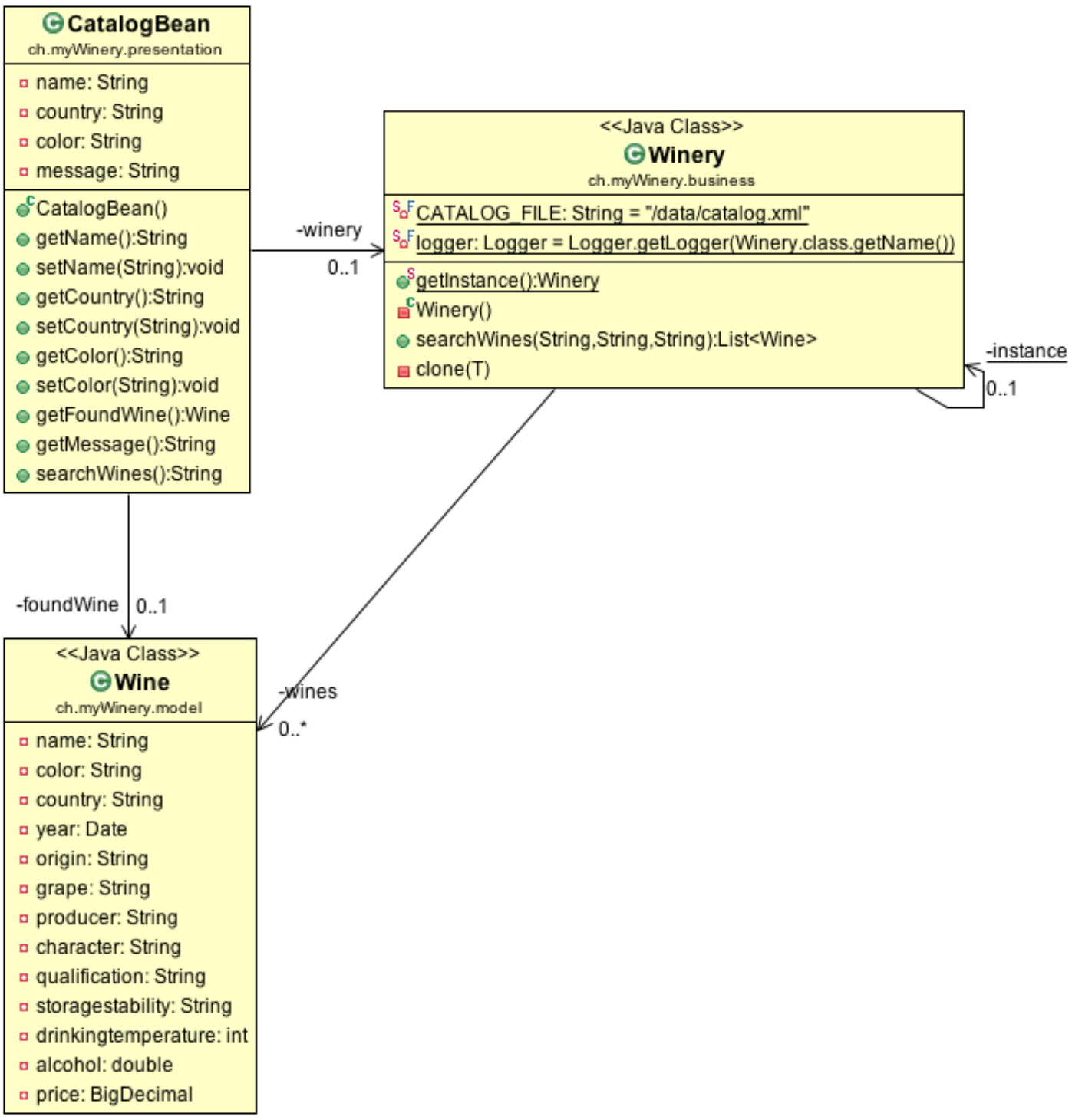
Exercise 01: Catalog Bean

Beschreibung

Die Aufgabe dieser Übung besteht darin, ein Managed Bean und eine JSF Seite zu erstellen welche es dem Benutzer erlaub nach Wein zu suchen.



Aufgabe

1. Extrahiere das bereitgestellte Archive in den Workspace als "myWinery01"
2. Implementiere das Managed Bean (CatalogBean)
   `src/main/java/ch/myWinery/presentation/CatalogBean.java`
   mit einer Methode welche Wein anhand der Kriterien sucht
3. Erstelle die JSF Seite:
   `src/main/webapp/catalog.xhtml`
   welche mittels Suchfeldparameter den gefundenen Wein anzeigt
4. Deploye die Applikation und Teste sie

myWinery

Catalog

Name	Solabal
Country	
Color	

Search

Name	Solabal Reserva
Country	Spanien
Year	Mon Jan 01 00:00:00 CET 2007
Origin	DOCa Rioa
Grape	100% Tempranillo
Producer	Bodegas Solabal
Storagestability	2013 - 2018
Drinking Temperature	18
Alcohol	14.5
Price	28.30

Exercise 02: Navigation

Beschreibung

Die Aufgabe dieser Übung besteht darin, zwei JSF Seiten zu erstellen und mittels Navigation zu Verbinden.

Aufgabe

1. Definiere im bereits bestehenden Managed Bean `src/main/java/ch/myWinery/presentation/CatalogBean.java`
   ein "outcome" der Winesuche welche auf die nächstte Seite verweist
2. Erstelle die JSF Seite
   `src/main/webapp/wineDetails.xhtml`
   welche die Details des gefundenen Weines anzeigt
3. Deploye die Applikation und Teste sie

myWinery

Catalog

Name Solabal

Country

Color

Search

myWinery

Wine Details

Name	Solabal Reserva
Country	Spanien
Year	Mon Jan 01 00:00:00 CET 2007
Origin	DOCa Rioa
Grape	100% Tempranillo
Producer	Bodegas Solabal
Storagestability	2013 - 2018
Drinking Temperature	18
Alcohol	14.5
Price	28.30

Back

Exercise 03: Standard Component

Beschreibung

Die Aufgabe dieser Übung besteht darin, eine Daten Tabelle mit allen resultaten einer Wein suche anzuzeigen. Der Benutzer kann mittels Klick auf ein Objekt in der Tabelle die Details ansehen.

Aufgabe

1. Im Managed Bean:
   `src/main/java/ch/myWinery/presentation/CatalogBean.java`·
   speichere die resultate einer Weinsuche in eine Liste und implementiere eine Methode zum selektieren eines Weines davon.
2. Erstelle eine JSF Seite:
   · `src/main/webapp/searchResults.xhtml`·
   welche die Tabelle der Weinsuch-Resultate anzeigt und es dem Benutzer erlaubt ein Wein zu selektieren (mittels Action-Parameter)
3. Formatiere die Wein Details Seite gemäss Printscreen
4. Deploye und Teste die web Applikation

Abgabestatus

myWinery

Catalog

Name

Country

Color Rot

Search

myWinery

Search Results

Title	Price
Casa Illana Tradicion	15,00 CHF
Solabal Reserva	28,30 CHF
Elego Amarone	45,00 CHF

Back

myWinery

Wine Details

Name	Casa Illana Tradicion
Country	Spanien
Year	2006
Origin	DO Ribera del Jucar
Grape	Bobal 45%, Tempranillo 30%, Syrah 25%
Producer	Casa Illana
Storagestability	2010 - 2015
Drinking Temperature	18° C
Alcohol	13,5 % Vol.
Price	15,00 CHF

Back

Exercise 04: Facelets Templating

Beschreibung

Das Ziel dieser Übung besteht darin, Facelets Templating bei der MyWinery Applikation einzusetzen.

Aufgabe

1. Schreibe ein Template mit zwei Bereichen
   `src/main/webapp/template.xhtml`
2. Verwende ein ui:param (e.g. für den Seiten Titel)
3. "Verschönere" die Web-Applikation mittels Bildern und CSS
4. Deploye und Teste die Web-Applikation

Exercise 05: Internationalization

Beschreibung

Ziel dieser Übung ist, die MyWinery App zu "Internationalisieren"

Aufgabe

1. Schreibe ein resource bundle
   `src/main/resources/bundles/texts_*.properties`
   welche die Englische und Deutsche texts entält
2. In den JSF Seiten
   `src/main/webapp/*.xhtml`
   ersetze alle literalen texts mit entsprechenden value expressions
3. Erweitere das JSF configuration file
   `src/main/webapp/WEB-INF/faces-config.xml`
   mit den unterstützten locales und den dazugehörigen resource bundle
4. Schreibe ein "language selector" welche es ermöglicht die Sprache zwischen Deutsch und Englisch mittels Fähnchen zu wählen
5. Deploye und Teste die App

Exercise 06: Messages

Beschreibung

Das Ziel dieser Aufgabe ist, Fehlermeldungen mittels eines message bundels zu verwenden

Aufgabe

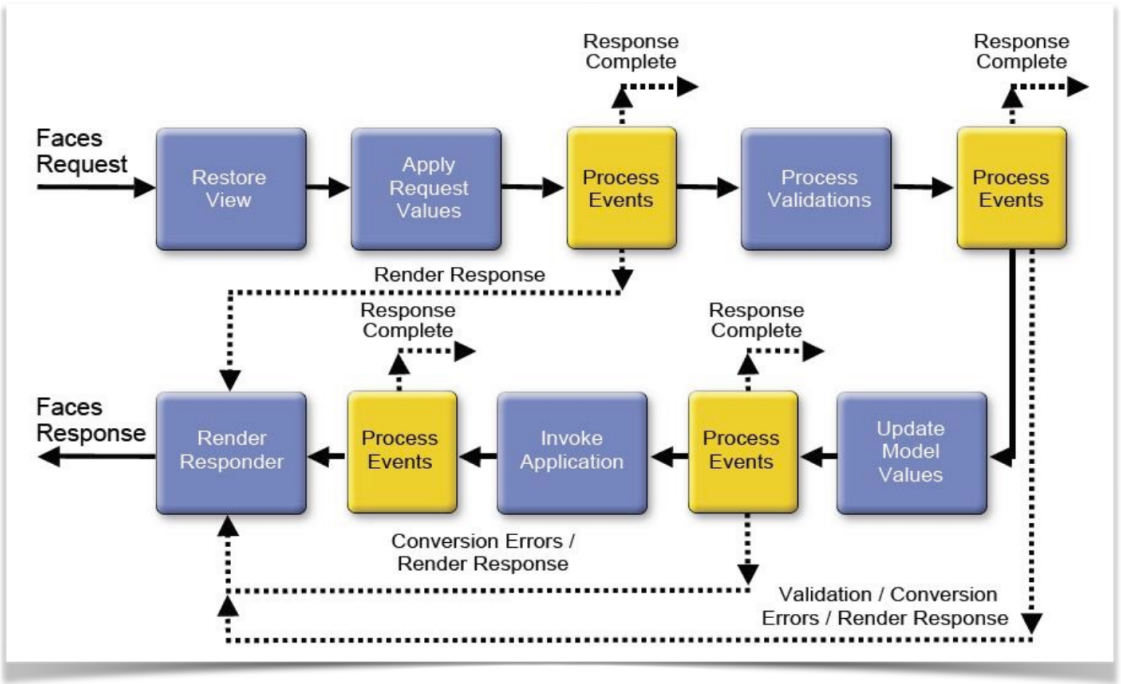
1. Schreibe ein message bundle
   `src/main/resources/bundles/messages_*.properties`
   welche die englischen und deutschen fehlermeldungen beinhaltet
2. Im Managed Bean
   `src/main/java/ch/myWinery/presentation/CatalogBean.java`
   soll eine fehlermeldung ausgegeben werden wenn die Weinsuche keine Ergebnisse liefert oder keine Suchkriterien angegeben wurden
3. Setze eine Message Komponente in der JSF Seite
   `src/main/webapp/catalog.xhtml`
   ein welche die Fehlermeldung anzeigt
4. Deklariere das message bundle im faces-config
   `src/main/webapp/WEB-INF/faces-config.xml`
5. Deploy the web application and test it

Abgabestatus

JAVA SERVER FACES

Chapter 07 - JSF Lifecycle

LEBENSZYKLUS

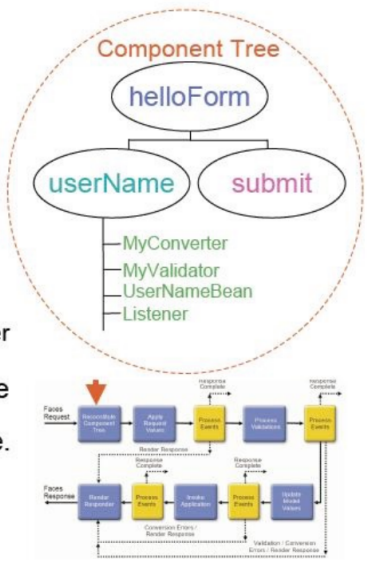


REQUEST LIFECYCLE

1. Restore View: Wiederherstellung des Komponentenbaumes
2. Apply Requests: Übernahme der Daten der Anfrage
3. Process Validations: Verarbeitung der Validierung
4. Update Model Values: Aktualisierung der Managed-Bean Daten
5. Invoke Applikation: Aufruf der Applikation
6. Render Response: Darstellen (rendern) der Antwort

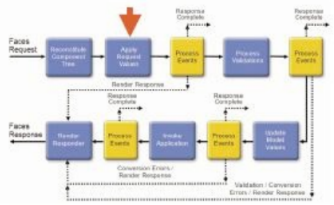
RESTORE VIEW

- Wird ausgelöst, wenn eine Seite vom Client angefordert wird.
- Der Component Tree wird aus der zugehörigen JSF Seite erzeugt, Event Handlers und Validators werden verknüpft.
  - Sofern es sich um ein initiales Request handelte, wird ein leeres View der Seite erzeugt und im weiteren Verlauf gefüllt.
  - Ansonsten existiert bereits ein korrespondierendes View, welches mit Hilfe von Zustandsinformationen, die auf dem Client oder Server gespeichert sind, wieder hergestellt wird.
- Entspricht der Verschachtelung der Tags, wie sie auf der JSP Seite eingetragen wurden, ist also die serverseitige Repräsentation einer JSF-Seite.
- Speichert den View in den *Faces-Context*.



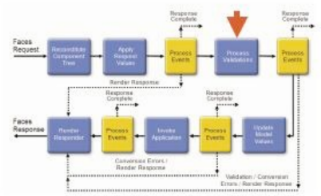
APPLY REQUEST VALUES

- Nach dem Aufbau des Komponentenbaums holt sich jede Komponente den neuen Wert aus dem Request.
  - Wert wird **lokal** in der Komponente gespeichert.
  - Mögliche Fehler werden im *FacesContext* gespeichert.
- Außerdem finden hier Datentypkonvertierungen der Werte statt.
- Registrierte Event-Listener werden in dieser Phase bereits benachrichitgt.



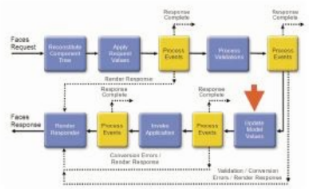
PROCESS VALIDATIONS

- Alle Validatoren, welche in der Apply Request Phase auf den Komponenten registriert wurden, werden ausgeführt.
- Überprüfung, ob lokal gespeicherte Komponentenwerte den Regeln der individuell gesetzten Validatoren entsprechen.
- Falls Validation fehlschlägt,
  - Fehlermeldung wird im *FacesContext* gespeichert
  - Es wird direkt zur Response Phase verzweigt.
  - Beispiel: Wenn eingegebene Zahl zwischen 0 und 10 liegen sollte und dieses nicht erfüllt ist.



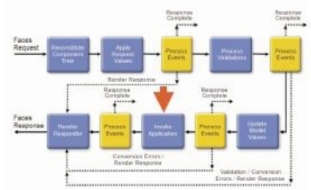
UPDATE MODEL VALUES

- Die JSF-Implementierung durchwandert den Component Tree und setzt die korrespondierenden serverseitigen Objekteigenschaften auf die lokalen Werte der Komponenten.
  - Updaten der Bean-Eigenschaften, die auf einen Eingabewert der Komponente zeigen.



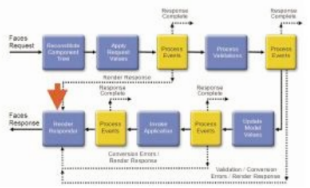
INVOKE APPLICATION

- In dieser Phase wickelt die JSF-Implementierung alle Events auf Anwendungsebene ab, wie z.B. das Abschicken eines Formulars oder die Verlinkung zu einer anderen Seite.



RENDER RESPONSE

- Durch die *encode*-Methode wird der Komponentenbaum, der in *FacesContext* gespeichert ist, gerendert.
- Wird nun durch alle Komponenten iteriert, stellt sich so Stück für Stück die Antwortseite zusammen.
- Traten in irgendwelchen anderen Phasen Fehler auf, wird die alte Seite neu gerendert mit entsprechend generierten Fehlermeldungen.



Exercise 08: Validation

Beschreibung

Ziel dieser Übung ist...

- Die Weine in einer Datenbank zu speichern
- Eingaben mittels Bean Validation zu überprüfen
- Fehlerseiten zu definieren

Aufgabe

1. Erstelle ein myWinery Datenbank (HSQLDB)
2. Stelle die Business-Klasse von XML-Daten auf DB um
3. Implementiere 'Bean Validation' in den Modell Klassen
   `src/main/java/ch/myWinery/model/*`
4. Erstelle eine JSF Seite welche es dem User ermöglicht neue Weine zu erstellen
   `src/main/webapp/catalogadmin.xhtml`
   stelle sicher dass die Eingaben validiert werden
5. Erweitere die Business Klasse 'Winery.java' mit einer Methode 'saveNewWine' welche es ermöglicht neue Weine in der Datenbank zu speichern.
   `src/main/java/ch/myWinery/business/Winery.java`
6. Erstelle eine Fehlerseite welche bei ViewExpired angezeigt wird
7. Deploye und Teste die Applikation

Exercise 09: (Custom)Converter

Description

The objective of this exercise is to implement a 'Custom Converter'.

Assignment

1. create a custom converter to handle the producers (implements Converter)
   src/main/java/ch/myWinery/converter/ProducerConverter.java
2. implement*getAsObject()*and*getAsString()*
3. add*@FacesConverter*with a valid *ID*to the Converter
4. add the custom converter to the JSF Component for selecting a producer (*f:converter* ...)
5. Deploy the web application and test it

Exercise 10: CompositComponent

Description

The objective of this exercise is to implement a 'Composit Component'.

Assignment

1. create a configurable 'inputSpinner' composit component *src/main/webapp/resources/util/inputSpinner.xhtml*
2. use the a Javascript to increment/decrement the input value (inputSpinner.js)
3. create a custom CSS file (inputSpinner.css)
4. create a messagebundle for the default texts (inputSpinner.properties)
5. implement a new modle class "StockItem" which represent a Wine in your cellar
6. create a cellar site where you collect the information about your cellar
7. implement the inputSpinner component at the cellar- and catalog site for setting the amount of wines (catalog 6 Step / cellar 1 Step)
8. Deploy the web application and test it

Exercise 11: Ajax

Description

The objective of this exercise is to implement Ajax to the catalog search.

Assignment

1. implement Ajax to the inputText components to search for a wine by tiping
   src/main/webapp/catalog.xhtml
2. extend the catalogBean with the action 'searchWineByTiping()' and make sure the search will not star until at least tree caracters are tiped
3. *implement Ajax to the selectOneRadio component for selecting 'new wine' or 'new producer'*
   src/main/webapp/catalogAdmin.xhtml
4. *implement Ajax to the inputText components for immediate validation*
   src/main/webapp/catalogAdmin.xhtml
5. Deploy the web application and test it

Release Notes

myWinery 1.0

Schule:	Höhere Fachschule für Technik Mittelland
Kurs:	2521 Web Applikationen I (FS15)
Dozent:	Dominik Tschumi
Autor:	Student
Datum:	13. November 2015

Server Konfiguration

- Datasource:
- Database User:
- webroot:
- DB Content:

Basis Funkionen

Lorem ipsum dolor:

- consetetur sadipscing
- sed diam nonumy
- eirmod tempor invidunt

Lorem ipsum dolor:

- consetetur sadipscing
- sed diam nonumy
- eirmod tempor invidunt

Zusätzliche Funktionen

Lorem ipsum dolor:

- consetetur sadipscing
- sed diam nonumy
- eirmod tempor invidunt

Verwendete JSF Features

Lorem ipsum dolor:

- consetetur sadipscing
- sed diam nonumy
- eirmod tempor invidunt

Präsentation

- Technische Präsentation (keine Leidensgeschichte)
- Zeit: 10-15 min (inkl. Demo)
- Themen:
  - Ausgangslage
  - Basis Funktionen
  - Zusatz Funktionen
  - Highlight
  - Ausblick (wie könnte das Projekt ausgebaut werden)
  - Fazit