Wicket Labs

Labs Wicket

Training voorbereiding

Open een terminal en geef de volgende commando's:

```
# cd Documents
# hg clone https://bitbucket.org/infosupport/wicket
Username: cursist
Password: p@ssw0rd          //0 is een nul
# cd wicket
```

Lab 1 – Basics

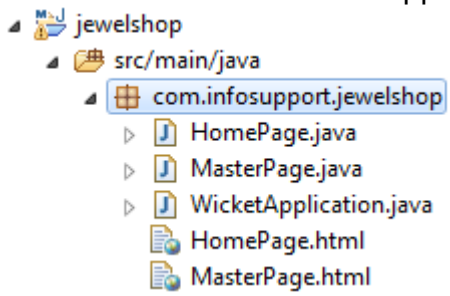
Lab voorbereiding
Importeer project jewelshop-lab1 in Eclipse (File→ import → existing Maven project)

Note: Dit project opent met twee compile errors die in stap 2 opgelost worden.

Stap 1 – Configureer Wicket in web.xml
Om wicket te activeren, dient er eerst een filter geconfigureerd te worden. Dit wordt gedaan in het bestand `src/main/webapp/WEB-INF/web.xml`.
Open dit bestand en bekijk de configuratie.

Stap 2 – Creeer de homepage
Voeg in de package `com.infosupport.jewelshop` een nieuwe *Wicket WebPage* met behulp van de QWickie wizard in Eclipse (file → new → Other → Wicket WebPage).

- Gebruik als naam: `HomePage` en laat de klasse overerven van `com.infosupport.jewelshop.MasterPage`.
- Daarna dient de `com.infosupport.jewelshop` package er als volgt uit te zien:



- Nu moet Wicket weten welke page klasse dient als homepage klasse. Open daarvoor de `WicketApplication` klasse en configureer de `getHomepage()` methode door HomePage.class terug te laten geven.

- In dit lab gaan we aansluiten op een bestaand design. De opmaak van de pagina is gedefinieerd in Masterpage.html en in de HomePage.html gaan we deze opmaak overerven (meer details volgen later). Vervang de inhoud van Homepage.html met de volgende inhoud.

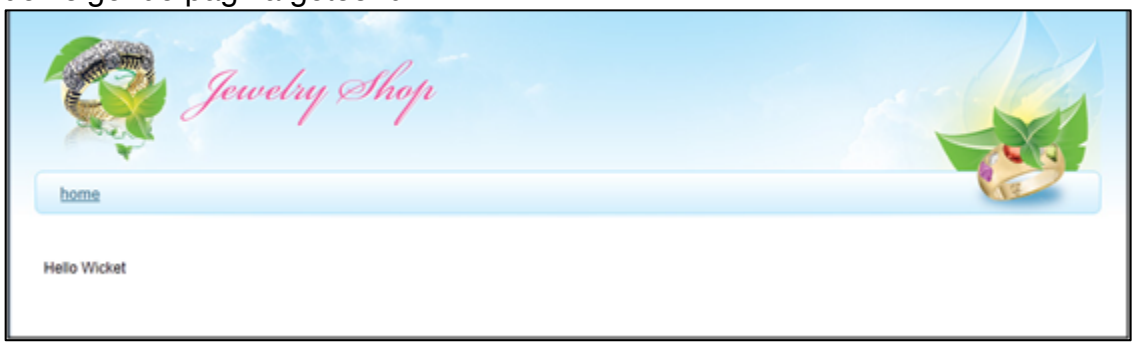
```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:wicket="http://wicket.apache.org">
  <wicket:head>
    <title>Jewel store - Home</title>
  </wicket:head>
<body>
  <wicket:extend>
  </wicket:extend>
</body>
</html>
```

- … en voeg binnen de `wicket:extend` tag een nieuwe span toe. Zorg ervoor dat deze span vanuit code een waarde krijgt (bijv. de huidige tijd of “hello wicket”) weergeeft.
- … en fix de compile errors.

Stap 3 – Run het project

Test de applicatie door de `Start` klasse (is te vinden in de test source folder) als Java applicatie te starten.

Navigeer met een browser naar `http://localhost:8080/js`, en als het goed is wordt de volgende pagina getoond.



Stap 5 – Valideer met unittesten

Maak in de test-source map een nieuwe testklasse genaamd `TestHomePage` met een test `homepageRendersSuccessfully()` die valideert dat de pagina correct gerenderd wordt.

bonus Schrijf een test om te valideren dat de span de correcte waarde krijgt.

Lab 2 – Databinding en DataTables

Lab voorbereiding
Importeer project jewelshop-lab2 in Eclipse (File→ import → existing Maven project)

In dit lab gaan we onze Juwelenshop uitbreiden met nieuwe features. Wederom focussen we ons op de Java code. Het HTML design is reeds gedaan.

Stap 1 – Forms en property models

- Open `ContactPage.html` en bekijk de HTML die reeds gemaakt is.
- Maak in de package com.infosupport.jewelshop.forms een nieuwe klasse genaamd ContactForm en laat deze overerven van Form. Gebruik als generic type de Contact klasse.
- Voeg een constructor toe:

```
public ContactForm(String id, Contact contact) {
  super(id, new CompoundPropertyModel<Contact>(contact));
}
```

- Gebruik TextFields en een CompoundPropertyModel om de velden Name, Email, en Phone te binden aan de corresponderende textfields.
- Override de onSubmit() methode en laat de inhoud van het contact modelobject printen naar de console wanneer deze uitgevoerd wordt.

Stap 2 – Creëer een Juwelenlijst pagina

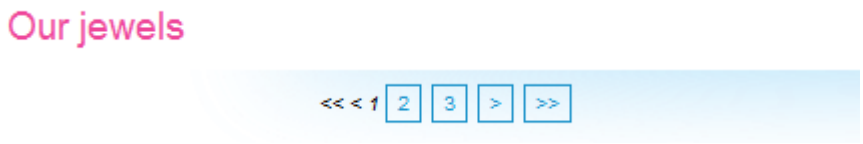
Open HomePage.html en bekijk de HTML structuur. Onze designer heeft reeds HTML code bedacht voor de weergave van één juweel (dit is de inhoud van de div met als class feat_prod_box). Deze HTML dient als template gebruikt te worden voor alle juwelen uit de lijst.
Voor ieder juweel dient een plaatje met beschrijving getoond te worden

- Voeg Java code toe in de HomePage klasse om alle juwelen te tonen. Probeer het eerst zo simpel mogelijk door bijvoorbeeld eerst alleen de omschrijving te tonen in de template. Gebruik als Java type een ListView. Om alle juwelen op te halen, kan gebruik gemaakt worden van de JewelInventorySrv.getJewels() methode.
  Enkele tips:
  - De plaatjes staan in images/;
  - Gebruik het Java type `ContextImage` om een plaatje te tonen;
  - Gebruik de `jewel.getImage()` methode om de bestandsnaam van het plaatje te construeren.
- Sommige juwelen zijn zeer bijzonder en dienen daarom een speciaal icoon te krijgen. Schrijf javacode die dit icoon alleen gaat tonen als `Jewel.isSpecialProduct` de waarde `true heeft`.
  Tip: Gebruik de `setVisible()` methode.

Hierna dient de pagina er ongeveer als volgt uit te zien.

- Net als in een echte applicatie willen we paging functionaliteit. Voeg daarvoor een nieuwe div toe onder de regel `<div class=”center_content”>`.
  ```
  <div class="pagination" wicket:id="navigator"></div>
  ```
- Verander het type ListView in een PageableListView en voeg een PagingNavigator toe in de HomePage klasse.

Daarna is paging toegevoegd:



Stap 3 – Maak een detachable model

De applicatie is nu nog niet geschikt voor Enterprise situaties en grote hoeveelheiden juwelen zal ongetwijfeld tot geheugenproblemen leiden.

- Om dit te vermijden, maak een nieuwe klasse genaamd `DetachableJewelModel` in de package `com.infosupport.jewelshop.models`.
- Maak deze klasse verder af en zorg ervoor dat de juwelenlijst dit model gebruikt.
- Test de applicatie.

Stap 4 – Gebruik de data table

In deze stap maak je kennis met een data table klasse. Dit is een rijk component waarop sortering, filtering, etc. allemaal aanwezig is.

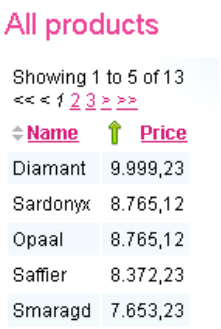
- Maak een nieuwe WebPage genaamd **JewelsOverview.** Laat deze pagina overerven van MasterPage en geef de volgende HTML als inhoud:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:wicket="http://wicket.apache.org">
<body>
	<wicket:extend>
		<div class="title">All products</div>
		<div class="clear"></div>
		<div class="center_content">
			<table wicket:id="jewelTable"/>
		</div>
	</wicket:extend>
</body>
</html>
```

- Maak een linkje in de MasterPage klasse naar de gemaakte WebPage.
- Maak in de JewelsOverview klasse een dataTable met de volgende kolommen:
  - Een kolom voor de naam

- Een kolom voor de prijs;

- Voeg sortering toe (omhoog en omlaag)
- Daarna dient de pagina er als volgt uit te zien:



Lab 3 – Forms

Lab voorbereiding
Importeer project jewelshop-lab3 in Eclipse (File→ import → existing Maven project)

In dit lab gaan we het contactformulier uitbreiden met validatie en conversie.

Stap 1 – Binden van het Contact formulier
Open het ContactPage.html bestand.

Kies een aantal input velden en bind die met de corresponderende properties in het Contact object. Validatie en conversie is nu nog niet nodig.
Tips:

- Interested options kunnen opgehaald worden met:
```
List<String> interestedOptions =
      java.util.Arrays.asList("Edelstenen","Juwelen","Oud goud");
```
- Een lijst van countries kan opgehaald worden met: `Country.getPoplulairCountries()`

- **Bonus**: bind het "I'm interested in" veld met het Contact object.
  Voor een lijst van ContactType values gebruik: `Arrays.asList(ContactType.values())`

Stap 2 – Conversie

- Verander het datatype van het zip code veld in de Contact klasse van String naar DutchZipCode. Genereer opnieuw getters en setters voor deze property.
- Schrijf een Converter die de String waarde converteert naar een DutchZipCode object en omgekeerd.
- Verander de declaratie van het zip textField van TextField<String> naar TextField<DutchZipCode> en registreer de converter in de WicketApplication class.

Stap 3 – Validatie

- Maak alle velden verplicht.
- Valideer dat het birthdate veld een valide datum is.
- Zorg ervoor dat het naamveld alleen karakters bevat.

- Schrijf een custom validator genaamd: ZipCodeInRangeValidator die alleen postcodes accepteert tussen: 2000 en 5000.

Stap 4 – Feedback

- Zorg ervoor dat de postcode validator de volgende melding geeft:
```
This zipcode ${numbers} {letters} with range is out of the range
${range_min} to ${range_max}.
```
- Verander de required melding naar:
```
Field with name '<fieldname>' is required.
```

Stap 5 – Bonus

- Voeg CSS code toe die :
  - error messages rood weergeeft
  - info messages blauw weergeeft
  - warnings oranje laat zien.

Lab 4 – Pagina's en navigatie

Lab voorbereiding
Importeer project jewelshop-lab4 in Eclipse (File→ import → existing Maven project)

In dit lab gaan we een Juweel-detail pagina maken.

Stap 1 – Aanmaken van nieuwe pagina
Maak een nieuwe WebPage genaamd `JewelDetailsPage` in de package `com.infosupport.jewelshop`. Laat deze overerven van MasterPage.
De markup voor de detailpagina kun je copy/pasten uit het bestand JewelDetailsPage.txt.

Zorg ervoor dat de details van het gegeven juweel getoond worden.
Tips:

- Gebruik `JewelInventorySrv.getInstance().getJewelById(jewelId)` om een juweel op basis van id op te halen.
- **Let op:** deze methode gooit een exception wanneer het juweel niet gevonden kan worden. Implementeer en nette foutafhandeling.

Stap 2 – Bookmarkable links

- Maak een link van de HomePage.html naar de JewelDetailsPage.
- Zorg ervoor dat deze link bookmarkable is zoals bijvoorbeeld:
  localhost:8080/jewelshop/wicket/bookmarkable/com.infosupport.jewelshop.JewelDetailsPage?jewelId=1

Hierna dient de detailspagina er ongeveer als volgt uit te zien: