Aufgaben "Cheesr Onlineshop" (1)

# Einleitung

Dieses Aufgabenblatt beschreibt den Aufgabensatz „Cheesr Onlineshop“ für das Wicket-Training. Es basiert auf der Beispielanwendung aus „Wicket in Action“, Manning, 2008.

# Übersicht

Im Rahmen der Übung wird ein Online-Shop entwickelt. Der Online-Shop verfügt über zwei Screens („Index“, „Checkout“), in dem Käse ausgewählt und bestellt werden kann.

Die Musterlösungen mit Beispielcode und alle in den Aufgaben referenzierten Artefakte befinden sich online auf github: <https://github.com/anderscore-gmbh/wicket-2019.03>

# Vorgehen

Im weiteren Verlauf der Übungsaufgabe wird ein Online-Shop Schritt für Schritt entwickelt. Jeder Schritt behandelt dabei ein Thema. Für jeden Schritt steht eine Beispiellösung in einem Ordner zur Verfügung.

Verwenden Sie den User Guide (ausgedruckt) oder andere Referenzen (online) um weitere Informationen über die verwendeten Komponenten zu erhalten.

Sollten Sie an einer Stelle nicht weiterkommen oder eine Inspiration benötigen können Sie gerne in die Beispiellösung schauen oder den entsprechenden Code per Copy&Paste übernehmen.

# Aufgabestellung

Für einen Käse-Einzelhandel soll ein Online-Shop in Apache Wicket entwickelt werden. Hierzu wurden von einer Webagentur bereits ein Entwurf erarbeitet (<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/shop-designes>). Er besteht aus:

* Einer Startseite mit Produktauswahl (index.html)
* Einer Bestellseite mit Adresseingabe (checkout.html).

Die Middleware-Abteilung hat bereits eine einfache Schnittstelle zur Anbindung an das Warenwirtschafts- und Abrechnungssystem erstellt. (<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/cheesr-backend>). Über den Service-Client (com.gfu.wicket.backend.BOServices) können die entsprechenden Middleware-Systeme angesteuert werden.

Ihre Aufgabe besteht darin, einen Online-Shop mit der beschriebenen Funktionalität von Grund auf zu entwickeln.

*Tipp: Beachten Sie die Hinweise zu den Aufgaben!*

## Auftrag 1: Init

### Aufgabenstellung

* Laden Sie den Beispielcode von der Projektseite herunter. Installieren Sie das Backend-Projekt (Ordner: cheesr-backend – Kommando: mvn install)
* Erstellen Sie ein neues Wicket-Projekt in einem anderen Ordner. Übernehmen Sie die Abhängigkeit zum cheesr-backend in die Maven pom.xml.
* Übernehmen Sie die HTML-Designs aus dem Beispielcode (Ordner: shop-designs) in ihre Anwendung.
* Erstellen Sie eine WebPage Index, die von der Anwendung als Startseite verwendet wird und die Übersichtsseite zeigt. Erstellen Sie zudem eine WebPage Checkout für den Bestellvorgang

### Hinweise

* Verwenden Sie den maven wicket-archetype-quickstart.
* Die „Homepage“-Dateien (Homepage.html / Homepage.java) werden nicht benötigt. Sie können sie als Vorlage für Index und Checkout verwenden und danach löschen.
* Die backend-services können auch als eclipse-Projekt eingebunden werden.
* Der Name der Page und des zugh. HTML-Template muss exakt gleich sein (case sensitive).

### Beispiellösung

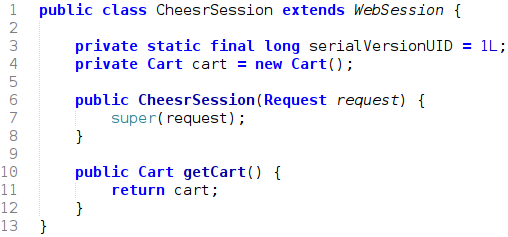
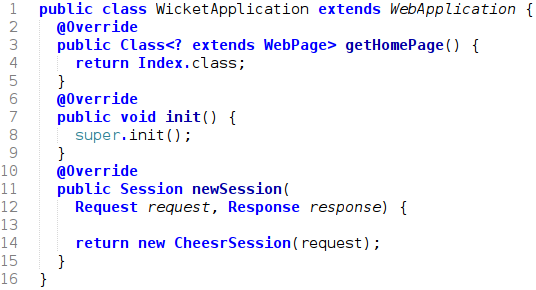
<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-1-CheesrWebApp>

## Auftrag 2: Warenkorb und Session

Der Warenkorb soll in der Session gespeichert werden und allen Teilen der Anwendung zur Verfügung stehen.

### Aufgabenstellung

* Erstellen Sie eine Klasse CheesrSession, abgeleitet von org.apache.wicket.protocol.http.WebSession – sie enthält den Warenkorb als Attribut (com.gfu.wicket.backend.bo.Cart).
* Überschreiben Sie den Hook   
  public Session newSession(Request request, Response response)  
  in der Klasse WicketApplication in der Sie eine neue CheesrSession zurückgeben.
* Erstellen Sie ein abstrakte Basisklasse CheesrPage für Index und Checkout. Erstellen Sie die Methode getCart(), die den Warenkorb aus der Session zurückgibt.



### Hinweise

* Die Basisklasse ist nicht direkt erforderlich, vereinfacht aber den Code, da alle Pages auf den Warenkorb zugreifen müssen.
* In Webpages – und damit auch in der CheesrPage – können Sie mittels getSession() auf die aktuelle Session zugreifen.

### Beispiellösung

<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-2-CheesrWebApp>

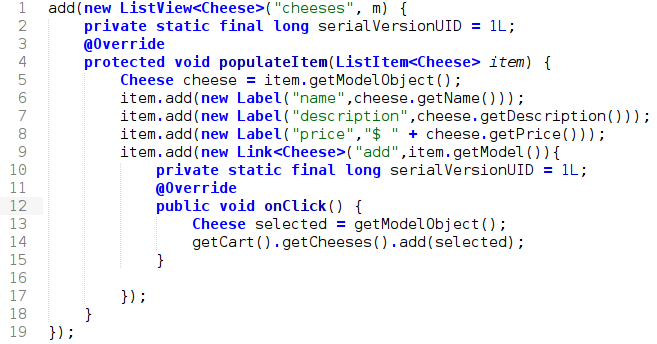
## Auftrag 3: Verfügbare Käsesorten anzeigen

Die im Backend gespeicherten Käsesorten sollen dem Benutzer angezeigt werden.

### Aufgabenstellung

* Erstellen Sie ein Klasse CheeseListModel extends Model<ArrayList<Cheese>>. Beim Aufruf der überschriebenen Methode getObject() soll die Liste aus dem Backend-System zurückgegeben werden.
* Verwenden Sie die Wicket-Komponente ListView um die Liste der Käsesorten auf der Index-Seite darzustellen. Bei einem Click auf den Link *addToCart* soll der Käse in den Warenkorb übernommen werden (vgl. Abbildung 1).

Abbildung 1: Einbettung einer ListView in einer Page



* Vergeben Sie passende Wicket-IDs im HTML.

### Hinweise

* HTML-Code wird ausgeblendet, wenn er von <wicket:remove>…</wicket:remove> umschlossen ist.
* Weitere Informationen zur ListView finden Sie unter: <https://ci.apache.org/projects/wicket/guide/8.x/single.html#_displaying_multiple_items_with_repeaters>
* Optional: Verwenden Sie eine PageableListView-Komponente mit Pagination.

### Beispiellösung

<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-3-CheesrWebApp>

## Auftrag 4: Warenkorb anzeigen.

Der Inhalt des Warenkorbs muss angezeigt werden.

### Aufgabenstellung

* Verwenden Sie erneut eine ListView um den Inhalt des Warenkorbs darzustellen
* Bei einem Click auf Checkout soll auf die Checkout-Seite weitergeleitet werden.
* Der Checkout-Button soll nur angezeigt werden, wenn der Warenkorb tatsächlich gefüllt ist.

### Hinweise

* Der Wert des Warenkorbs kann über Cart.getTotal() ermittelt werden.
* Beachten sie die API der Link-Komponente:
  + Via setResponsePage(newCheckout()) wird die Seite festgelegt, auf die weiter geleitet werden soll.
  + Durch überschreiben von isVisible() können Sie festlegen, wann der Link sichtbar sein soll.

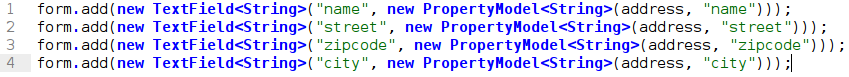
### Beispiellösung

<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-4-CheesrWebApp>

## Auftrag 5: Bestellvorgang umsetzen

Eingegebene Adressdaten müssen an das Backend-System weitergeleitet werden. Die Bestellung muss erfolgen.

### Aufgabenstellung

* Instanziieren Sie wicket-Objekte für das Formular und alle Felder. Bei einem Click auf „Order!“ soll die Bestellung an das backend übertragen und in der Warenkorb in der Session gelöscht werden.
* Bei einem Click auf „Cancel“ soll die Übersicht (Index) angezeigt werden – der Warenkorb muss erhalten bleiben.
* Verwenden Sie ein PropertyModel um das Eingabefeld und die Adresse des Warenkorbs miteinander zu verbinden.  
  

### Hinweise

* Beachten Sie die API-Dokumentation zu den Wicket-Klassen Form, TextField und Button.
* Das Backend-System gibt erhaltene Bestellungen auf der Konsole (System.err) aus.
* Die Klasse Cart hält eine Referenz auf die Adresse.

### Beispiellösung

<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-5-CheesrWebApp>

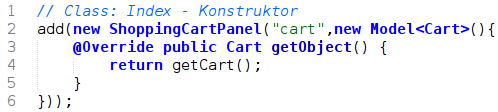
## Auftrag 6: Wiederverwendbare Komponenten: Panels

Im Akzeptanz-Test ist aufgefallen, dass die Benutzer den Inhalt des Warenkorbs sehen möchten, wenn Sie ihre Adressdaten eingeben. Hierzu soll der Warenkorb auch auf der Checkout-Seite angezeigt werden.

### Aufgabenstellung

* Erstellen Sie eine Klasse ShoppingCartPanel extends Panel mit einem Konstruktor ShoppingCartPanel(String id, IModel<Cart> cart). Verschieben Sie den Java-Code zum Aufbau der entsprechenden ListView und des Gesambetrags („Total“) in diese Klasse.
* Erstellen Sie eine neue HTML-Seite ShoppingCartPanel.html im gleichen Paket. Verschieben Sie den HTML-Code analog zu Java-Code
* Instanziieren Sie in den Klassen Index und Checkout ein neues ShoppingCartPanel und fügen Sie es dem Komponenten Baum hinzu.   
  Erstellen Sie im HTML-Code ein DIV-Element mit passender Wicket-ID an der Stelle, an der das Panel erscheinen soll.

### Hinweise

* Alle innerhalb von <wicket:panel>..</wicket:panel> aufgeführten Teile des DOM-Baums gehören zum Panel. So können bspw. oberhalb und unterhalb HTML / CSS / JavaScript eingebunden werden, die nicht Bestandteil des Panels sind.
* Weitere Informationen zu Panels finden Sie im Wicket-Guide:  
  „Kapitel 5 Wicket as page layout manager“
* An den Konstruktor des ShoppingCartPanel sollte ein Model mit Generic-Type Cart übergeben werden (best-pratice). In der Index-Page kann das Panel dann wie folgt initialisiert und eingebettet werden:  
    
    
  Welcher Bug steckt in dieser Variante?  
  

### Beispiellösung

<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-6-CheesrWebApp>

## Auftrag 7: Validierung

Im Test ist aufgefallen, dass die Benutzer zum Teil leere Adressangaben machen. Das Eingabe-Formular muss entsprechend validiert werden.

### Aufgabenstellung

* Markieren Sie die TextFields als „required“ indem Sie.setRequired(true) aufrufen.
* Instanziieren Sie ein Feedbackpanel um Fehler anzuzeigen – erstellen Sie im HTML einen DIV-Container mit passender ID um das Panel aufzunehmen.  
  add(newFeedbackPanel("feedback"));

### Hinweise

* TextFields verwenden das Builder Pattern. Bei setRequired(.) wird das Textfield zurückgegeben.

### Beispiellösung

<https://github.com/anderscore-gmbh/wicket-2019.03/tree/master/Aufgabe-7-CheesrWebApp>