Übungsblatt 4 (03.11.2015)

Weboberflächen-Programmierung mit GWT, Systemfunktionalität, Systemtestfälle, White-Box-Test, Issue-Tracker

In dieser Übung:

- ✓ Erweitern Sie Ihre GWT-Weboberfläche um Interaktionen.
- ✓ Erstellen Sie Systemtestfälle für Ihre GWT-Weboberfläche.
- ✓ Erstellen Sie einen Kontrollflussgraph.
- ✓ Untersuchen Sie Issue-Tracker-Systeme von zwei Open-Source-Projekten

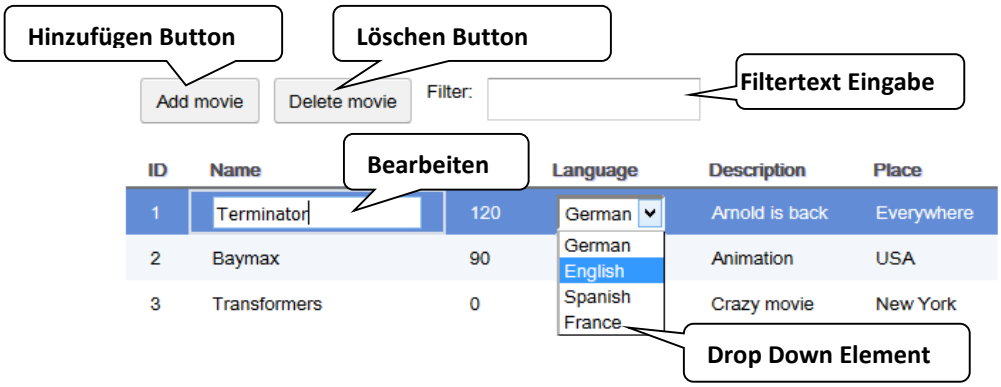
Aufgabe 4.1: Weboberflächenprogrammierung mit GWT - Interaktionen

Präsenz: Ja	**Punkte:** 12	**Team:** Nein	**Projekt:** Nein	**Testat**

In dieser Aufgabe werden Sie Ihre Weboberfläche aus Blatt 3 um Interaktionen erweitern. Die nachfolgend aufgelisteten Systemfunktionen und GUI-Details beschreiben die Interaktionen der NutzerInnen mit der Webanwendung „MovieManager“:

1. Bearbeiten der Daten eines Filmes
   - Die einzelnen Attribute eines Filmes können durch einen Klick in die entsprechende Zelle des Attributes bearbeitet werden.
   - Die Sprache wird über ein Drop-Down-Element aus einer vorgegebenen Liste (Deutsch, Englisch, Spanisch, Französisch) auswählt.
   - Die ID eines Filmes darf nicht bearbeitet werden.
   - Die Zeit eines Filmes ist eine positive ganze Zahl.
2. Anlegen von Filmen
   - Das Anlegen von Filmen wird über einen Button oberhalb der Tabelle durchgeführt. Durch diese Aktion erzeugt das System eine neue leere Zeile in der Tabelle und vergibt eine ID.
3. Löschen von Filmen
   - Filme werden über einen Klick in die Tabelle ausgewählt und werden anschließend durch einen Klick auf einen Button oberhalb der Tabelle gelöscht. Das System löscht den Film, der in der Tabelle ausgewählt wurde.
4. Filtern der Filme
   - Über die Eingabe eines Filtertextes werden nur die Filme angezeigt, deren Titel den Filtertext beinhalten. Als Filtertext ist eine beliebige Zeichenkette erlaubt. Ist der Filtertext leer, zeigt das System alle Filme an.
5. Speichern der Filme
   - Das System speichert die Filme jeweils beim Anlegen und Bearbeiten.
6. Start der Webanwendung
   - Beim Start der Webanwendung lädt das System alle gespeicherten Filme und zeigt sie an.

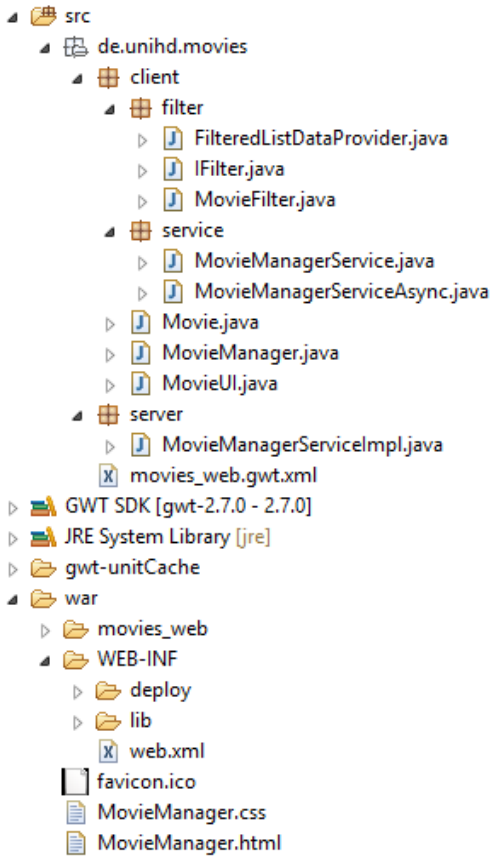
Die nachfolgende Abbildung zeigt die Details für die Umsetzung der Interaktionselemente:



Erweitern Sie Ihre Webanwendung, sodass die oben genannte Spezifikation der Systemfunktionen erfüllt ist. D.h. implementieren Sie alle geforderten Systemfunktionen und GUI-Details.

***Hinweise zur Implementierung**:*

- *In Moodle finden Sie das Archiv `04-movies-service.zip`, das Ihnen bei der Bearbeitung dieser Aufgabe hilft. Laden Sie sich das Archiv aus Moodle herunter und entpacken Sie es. Führen Sie die folgenden Schritte durch, um die Inhalte des Archivs in Ihre Webanwendung zu integrieren:*
  - *Ersetzen Sie die Datei `web.xml` in dem Ordner `war/WEB-INF/` mit der Datei aus dem Archiv.*
  - *Kopieren Sie die Ordner `filter` und `service` in das Paket `client` Ihres Projekts.*
  - *Kopieren Sie den Ordner `server` in das Paket `de.uhd.movies` Ihres Projekts.*
  - *Ihr Projekt sollte danach wie folgt aussehen:*



- *Verwenden Sie die Klasse `SelectionCell` für die Anzeige des Drop-Down-Menüs.*
- *Verwenden Sie die Klasse `SingleSelectionModel` um eine Zeile in der Tabelle auszuwählen.*
- *Verwenden Sie die Operation `setFieldUpdater()` zum Hinzufügen eines `FieldUpdaters` um Zellen in der Tabelle bearbeiten zu können.*
- *Verwenden Sie die Operation `addClickHandler()` zum Hinzufügen eines `ClickHandlers` zu den entsprechenden Buttons.*
- *Verwenden Sie die Klasse `FilteredListDataProvider` zum Umsetzten der Filterfunktionalität. Mit der Operation `setFilter()` können Sie den Filtertext setzen. Verwenden Sie die Operation `addValueChangeHandler()` zum Reagieren auf Eingaben in dem Filtertextfeld.*
- *Verwenden Sie für die Umsetzung der dauerhaften Speicherung der Filme sowie zum Laden der Filme die Klassen `MovieManagerServiceAsync` und `AsyncCallback`. Verwenden Sie den Aufruf `GWT.create(MovieManagerService.class)` um die Klasse `MovieManagerServiceAsync` zu initialisieren.*

Ergebnis:
Bitte speichern Sie Ihre Änderungen am Quellcode auf dem Git Server.
Speichern Sie bitte das exportierte Eclipse Projekte als .zip-Datei bis **Montag 9.11.2015 um 10.00 Uhr** in Moodle.

Aufgabe 4.2: Systemtestfälle

Präsenz: Nein	**Punkte:** 8	**Team:** Ja(2)	**Projekt:** Nein	

Erstellen Sie basierend auf der Spezifikation der Funktionen aus Aufgabe 4.1 geeignete *logische und konkrete* Systemtestfälle durch Black-Box-Test. Verwenden Sie zum Spezifizieren der Testfälle die Tabelle „04-Systemtests.xlsx“ (Datei zu finden in Moodle). Beachten Sie für die Namensgebung der Testfälle, so dass durch den eindeutigen Namen des Testfalls zu erkennen ist, was getestet wird.

Verwenden Sie für die Testfallableitung Ein-/Ausgabe- und Grenzwertüberdeckung (siehe auch Arbeitsblatt Black-Box der Vorlesung).

Identifizieren Sie für die Eingabe/das Bearbeiten von Daten zu Filmen etwaige Einschränkungen aus der Spezifikation und leiten Sie daraus die gültigen und ungültigen Äquivalenzklassen ab. Berücksichtigen Sie dabei ggf. Grenzwerte.
Ungültige Eingaben durch inkorrekte Worte (z.B. leerer String, Tippfehler) oder inkorrekte Datentypen (z.B. Text statt Zahl) müssen Sie nicht betrachten. Wird eine falsche Eingabe durch die vorgegebenen GUI-Details ausgeschlossen (z.B. falsche Sprache), müssen Sie diese Eingabe auch nicht betrachten. Vernachlässigen Sie für das Filtern von Filmen, an welcher Position der Filmliste der Film dem Filterkriterium entspricht und an welcher Position die Filme nach dem Filtern in der Ergebnis-Filmliste angezeigt werden.

Erstellen Sie nun Äquivalenzklassen für die Eingaben (d.h. Aufrufe mit ggf. Daten) der anderen Funktionen. Verfeinern Sie diese Äquivalenzklassen wenn nötig, um wichtige Ausgaben abzudecken. Betrachten Sie dabei auch den Systemzustand (d.h. die Menge der vorhandenen Filme).

Beschreiben Sie für alle identifizierten Äquivalenzklassen logische Testfälle und konkrete Testfälle mit entsprechenden Repräsentanten für die folgenden Funktionen:

- Zeit bearbeiten, Sprache ändern, Laden von Filmen und Filtern von Filmen.

Beschreiben Sie die Äquivalenzklassen durch Angabe der betroffenen Daten bzw. Funktionen und einer Benennung, die deutlich macht, ob gültig oder ungültig und einer Beschreibung, z.B.

Ort: Bearbeiten des Ortes eines Filmes
- *Gültig: GOrtK1: beliebige Zeichenkette*
- *Ungültig: keine*

Ergebnis:

Speichern Sie bitte eine .zip-Datei bis **Montag 9.11.2015 um 10.00 Uhr** in Moodle bestehend aus:
- PDF mit den Äquivalenzklassen für die verschiedenen Fälle
- Tabelle **04-Systemtests.xlsx** mit den spezifizierten und protokollierten Systemtestfällen

Aufgabe 4.3: White-Box-Test: Kontrollflussgraph & Anweisungsüberdeckung

Präsenz: Nein	**Punkte:** 8	**Team: Ja(2)**	**Projekt:** Nein	

1. Laden Sie sich das Projekt **04-moviemanager.zip** aus Moodle herunter und importieren Sie das Projekt in Eclipse. Erstellen Sie einen Kontrollflussgraph für den Code der Operation `overallRating` der Klasse `Movie` im Ordner `src`.
2. Überlegen Sie mithilfe des Kontrollflussgraphen, welche Testfälle zusätzlich zu den Testfällen durch die Äquivalenzklassenbildung **aus Aufgabe 3.2** für eine 100%ige Anweisungsüberdeckung fehlen. Ergänzen Sie diese bei Ihren spezifizierten logischen und konkreten Testfällen **aus Aufgabe 3.2** („03-Testcases.xlsx“). Falls sich keine weiteren Testfälle spezifizieren lassen, begründen Sie dies in einer Textdatei.

Ergebnis:

Speichern Sie bitte eine .zip-Datei bis **Montag 9.11.2015 um 10.00 Uhr** in Moodle bestehend aus:
- 1x PNG/JPEG/PDF Datei mit Kontrollflussgraph
- Testfallspezifikation **04-Testcases.xlsx** mit neuen spezifizierten und protokollieren Testfällen

Verwendung des GitHub Issue Trackers
Für die folgende Aufgabe verwenden sie den **GitHub Issue Tracker** der Open Source Projekte **JUnit** [1]und **Radiant**[2]. Für die Einführung in die Verwendung des GitHub Issue Trackers finden Sie im Moodle Kurs unter Tutorials das Tutorial 04-GitHub_ITS_Verwendung.pdf. Bearbeiten Sie dieses Tutorial bevor Sie mit der Bearbeitung der nachfolgenden Aufgabe 4.4 beginnen.

[1] https://github.com/junit-team/junit/issues/
[2] https://github.com/radiant/radiant/issues/

Aufgabe 4.4: Issue-Tracker-Analyse

Präsenz: Ja	**Punkte:** 12	**Team: Ja(2)**	**Projekt:** Nein	**Testat**

In dieser Aufgabe betrachten Sie Softwareentwicklungsprojektdaten aus Open Source Projekten. Dazu verwenden Sie den GitHub-Issue-Tracker. Ziel dieser Aufgabe ist es, dass Sie einerseits die

Verwendung der Issue-Tracker-Systeme in den zwei Open Source Projekten **JUnit**[1] und **Radiant**[2] bewerten. Andererseits sollen Sie insbesondere anforderungsrelevante Information identifizieren. Beantworten Sie dafür die nachfolgenden Fragen und gehen Sie dabei wie beschrieben vor:

1. **Verständnis für Projektdaten entwickeln**
   1.1. Bestimmen Sie für beide Projekte jeweils den 1. Issue, der erstellt wurde und geben Sie dann den Zeitraum (Anzahl der Monate)an, seit dem die beiden Projekte aktiv sind. Bestimmen Sie auch die Gesamtzahl der Issues in beiden Projekten.
   1.2. Bestimmen Sie jeweils die Anzahl der offenen und geschlossenen Issues und geben Sie für beide Projekte jeweils die absoluten und prozentualen (im Vergleich zur Gesamtzahl) Werte an.
2. **Versionen bewerten**
   2.1. Geben Sie jeweils die Anzahl der offenen und geschlossenen Issues für die ältesten beiden Versionen (Milestones) der beiden Projekte an (absolut und prozentual).
   2.2. Bewerten sie den Grad der Fertigstellung der Versionen bis zum 31.10.2015 aufgrund des Anteils geschlossener Issues in den Versionen. Verwenden Sie dazu ihre Ergebnisse aus der vorherigen Frage. Vergleichen Sie den Grad der Fertigstellung der beiden Versionen miteinander und beschreiben Sie diesen jeweils für beide Projekte.
3. **Anforderungen identifizieren**
   Identifizieren Sie Issues, in welchen keine Fehler beschrieben sind, sondern Anforderungen in Form von neuer Funktionalität (Feature).
   3.1. Bestimmen Sie jeweils alle Issues der beiden Projekte, die im Februar 2012 geschlossen wurden (Anzahl, IDs, Titel der Issues, Labels).
   3.2. Geben Sie für alle Issues, welche Sie in der vorherigen Fragestellung identifiziert haben an, ob es sich bei dem Issue um eine Feature-Beschreibung handelt oder nicht. Begründen Sie Ihre Einschätzung. Woran machen Sie fest, dass ein neues Feature beschrieben wird?
4. **Gesamtbewertung**
   4.1. Wie gut konnten Sie die vorherigen Fragen jeweils beantworten?
   4.2. In welchem der beiden Projekte sind die Issue-Daten einfacher zu verstehen und warum?
   4.3. Wie sollten Feature-Issue beschrieben werden, damit sie gut verständlich sind? Machen Sie einen Vorschlag.

Ergebnis:

Speichern Sie bitte Ihr Ergebnis als PDF bis **Montag 9.11.2015 um 10.00 Uhr** in Moodle.

[1] http://junit.org/ (JUnit Projektwebseite)
[2] http://radiantcms.org/ (Radiant Projektwebseite)