

ELO Flows Development

Aufgaben

Inhaltsverzeichnis

Implementierungsaufgaben	3
Dokumentenzähler	3
ELOas Modul	10
Teilnehmeraufgabe (optional)	11

Implementierungsaufgaben

Dokumentenzähler

Wir möchten jetzt einen Zähler (fachliche Identifikation eines Dokumentes) für die im Repository abgelegten Dokumente implementieren. Der Zähler garantiert, dass die Dokumente eines bestimmten Typs (z. B. festgelegt durch den BS SOL_TYPE oder die Metadatenmaske) eine eindeutige fortlaufende Kennung bekommen. Als Beispiel kann man eine Vertrags- oder Rechnungsnummer nennen.

Anforderungen (allgemein)

- Auswahl eines Zählers (ein Zähler kann bereits vorhanden sein)
- Definition/Eingabe eines neuen Zählers (Prefix, Postfix - Anzahl der Stellen)
- Erstellung eines neuen Zählers
- Rückgabe des aktuellen Zählerstandes in weiteren Komponenten
- Bereitstellung eines Triggers (REST)

Eine mögliche Anordnung der grafischen Komponenten im Tab *Einstellungen*.

Einstellungen

Dienst X

Komponente ✓ Dienst ✓ Benutzer ✓ Einstellungen O Zusammenfassung

 Dokumentenzähler
academy.training/Counter/0.0.2

⚡ Zähler auswählen

Erstellt einen neuen Zähler bzw. ein vorhandener Zähler kann ausgewählt werden.

Information

Zählername* ⓘ

Bitte füllen Sie dieses Feld aus

Neuer Zähler Hier bitte das Präfix und Anzahl der Stellen für den Zähler eingeben

Präfix ⓘ

Postfix ⓘ 6

Auswahlliste in den Vorschlägen.

Einstellungen

Dienst



Komponente ✓ Dienst ✓ Benutzer ✓ Einstellungen ○ Zusammenfassung

Zählername* ⓘ

ab ▾

String

Daten Vorschläge

NeuerZaehler

NeuerZaehler [3]

Rechnungszaehler

Rechnungszaehler [3]

SOL_ACTION_ID

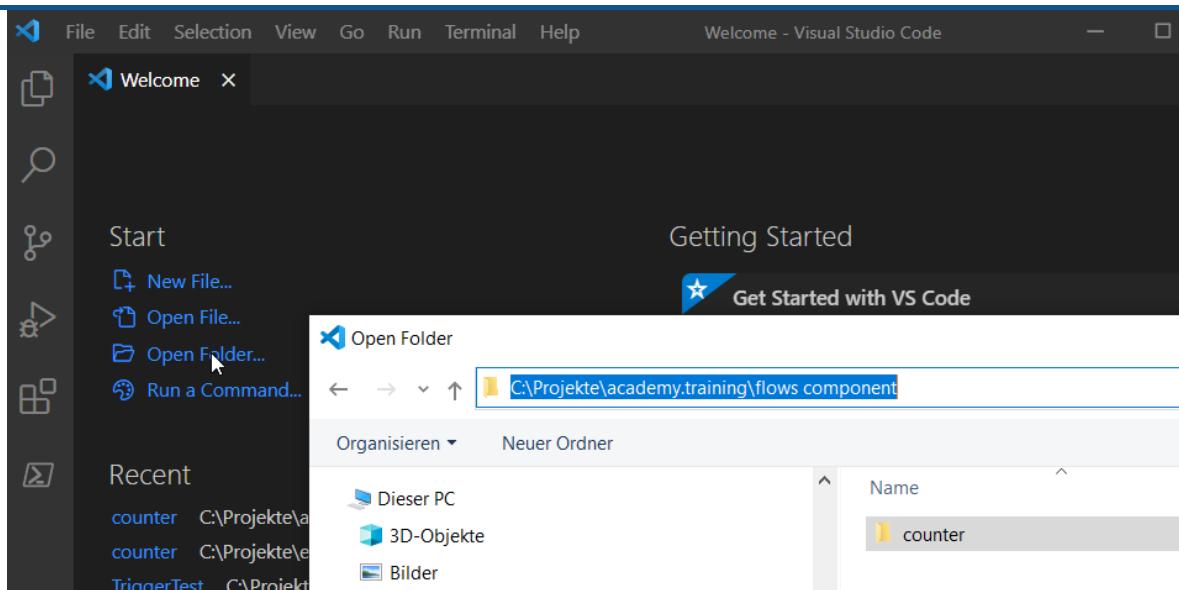
SOL_ACTION_ID [44]

Umsetzung der Anforderungen

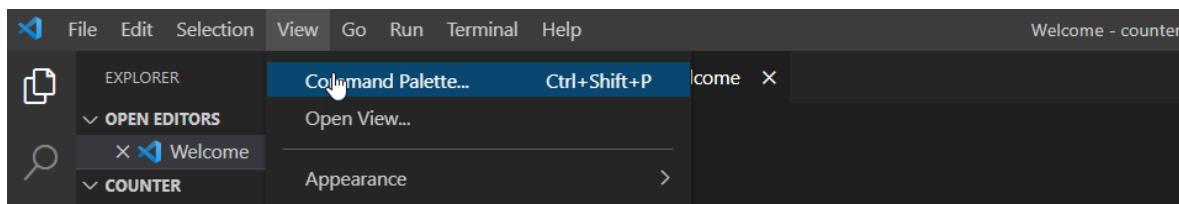
An dieser Stelle wird zum einen der Entwicklungsprozess zum anderen die Implementierung der Lösung skizziert. Die Anforderungen werden wir mithilfe des uns zur Verfügung stehenden Flows-Framework im Visual Studio Code durchführen.

Framework starten

1. Als ersten Schritt starten wir VS Code.
2. Öffnen Sie den für die Umsetzung vorbereiteten Projektordner (im Dateisystem).



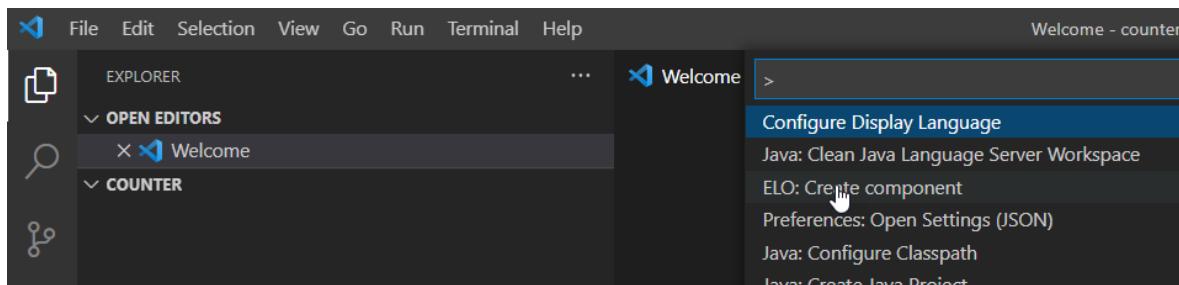
3. Weiterhin erstellen wir die initiale Projektstruktur wie in den nachfolgenden Abbildungen zu sehen ist.



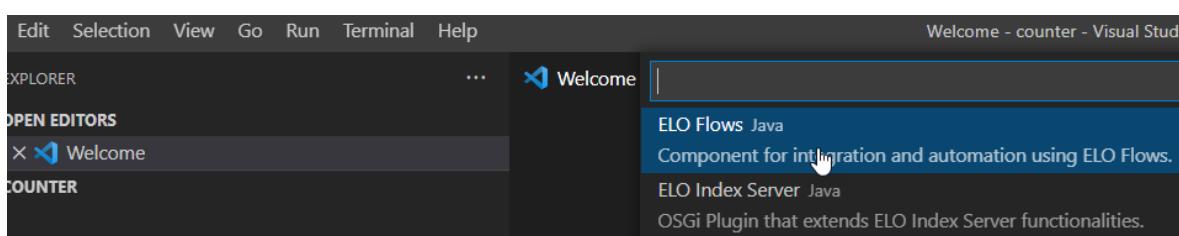
4. Wählen Sie die VS Code Befehlspalette aus. Benutzen Sie hierfür den Tasturbefehl STRG + SHIFT + P.

Alternative1: Drücken Sie F1.

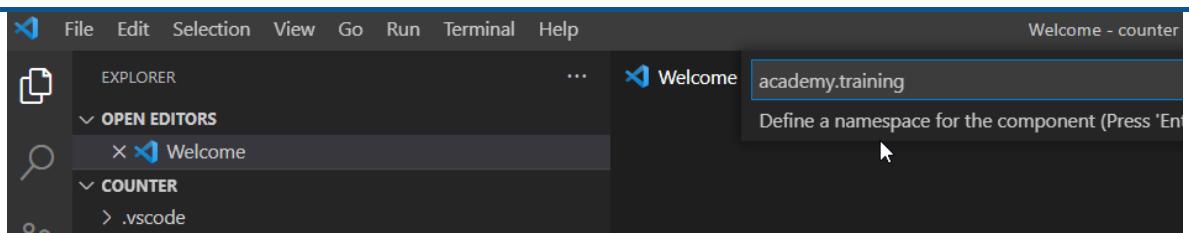
Alternative2: *View > Command Palette....*



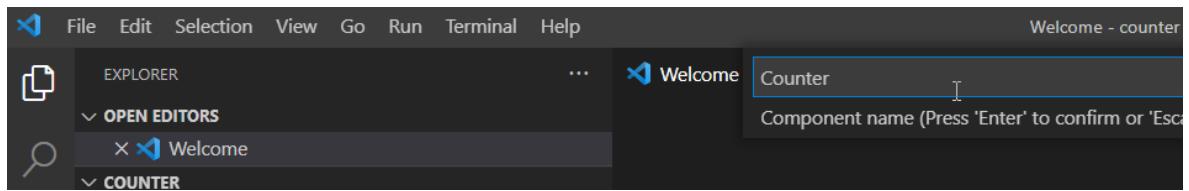
5. Wählen Sie *ELO: Create Component* aus.



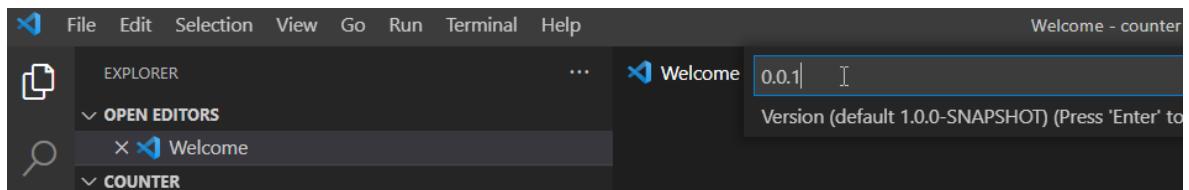
6. Wählen Sie *ELO Flows:Java* aus.



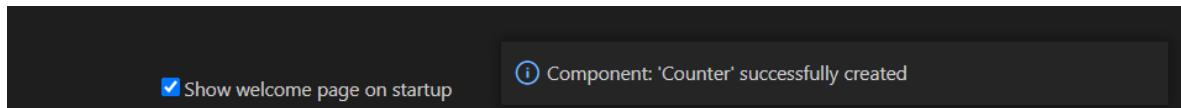
7. Geben Sie den Paketnamen ein und bestätigen Sie die Eingabe mit ENTER.



8. Geben Sie den Komponentennamen ein und bestätigen Sie die Eingabe mit ENTER.



9. Geben Sie die Versionsnummer ein und bestätigen Sie die Eingabe mit ENTER.



Wenn das Framework gestartet und das initiale Projekt angelegt wurde, dann erhalten Sie eine Bestätigungsmeldung (rechts unten im VSC).

```

3 import de.elo.ix.clic...
4
5 import com.elo.flows...
6 import com.elo.flows...
7 import com.elo.flows...
8 import com.elo.flows...
9
10 @Component(version = ...
11 public class HelloWorld...
12
13     @Connection
14     IXConnection ixCo...
15
16     @Service(displayName...
17     public HelloOutput...
18         HelloOutput re...
19         result.setText...
20         return result;
21     }
22
23     @Service(displayName...
24     @ConnectionRequir...
25     public HelloOutput...
26         HelloOutput re...
27         result.setText...
28         return result;
29     }

```

10. Wählen Sie jetzt in der angelegten Projektstruktur eine beliebige Java-Klasse aus.

```

"com.elo.flows.helloworld.servicewithconnection")

serviceWithConnection() {
= new HelloOutput();
Version: " + ixConnect.getImplVersion());

```

The workspace contains Java projects. Would you like to import them?

Source: Language Support for Java(TM) by Red ... Yes Always Later

11. Bestätigen Sie den Vorgang wie unten abgebildet.

```

a0e10323 Building [Done]
370a210c Building [Done]
7ed9431a Register Watchers [Done]
91780638 Send Classpath Notifications [Done]
d1ca7077 Send Classpath Notifications [Done]
8614de22 Building [Done]

```

ServiceReady

12. Warten Sie kurz, bis die Initialisierung des Java-Projektes erfolgreich abgeschlossen ist.

Information

Bei der ersten Initialisierung kann dies durchaus einige Sekunden in Anspruch nehmen.

Projektstruktur anpassen

ELO hat in einer neu angelegten Flows-Komponente eine Beispielkomponente implementiert. Sie können diese für neue Anforderungen anpassen oder durch vorheriges Löschen komplett neu aufsetzen. Im folgenden Beispiel gehen wir einen Mittelweg.

1. Löschen Sie den Inhalt der Dateien im Ordner *localisation*:

Achtung

Löschen Sie keine Dateien. Löschen Sie ausschließlich die Inhalte der Dateien.

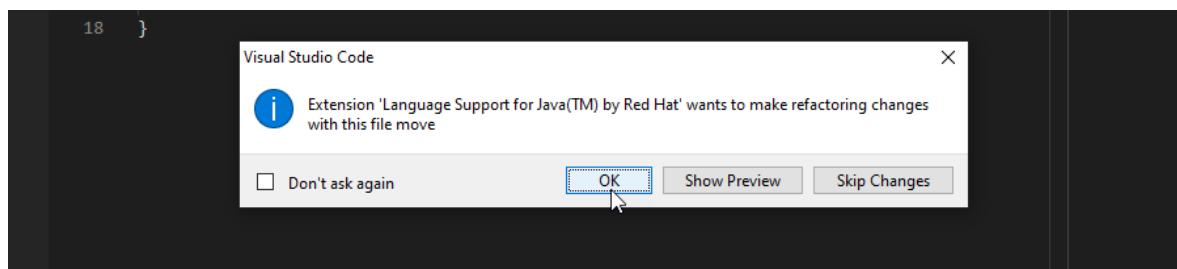
translations.de.properties und *translations.properties*.

2. Löschen Sie die Java-Klassen im Ordner `java\academy\training`:

HelloInput.java und HelloOutput.java

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows the project structure:
 - OPEN EDITORS**: Welcome, translations.properties, CounterComponent.java, translations_de.properties.
 - COUNTER**: .vscode, flows, .gradle, Counter (which contains bin), src\main (which contains java\academy\training, CounterComponent.java).
- WELCOME**: The active tab, showing the title "Welcome".
- CounterComponent.java**: An open editor window showing Java code for a component named CounterComponent. The code includes imports for package academy.training, IXConnection, Component, and Connection, along with the class definition and its constructor.



3. Passen Sie die Java-Klasse HelloWorldComponent.java an:

Nutzen Sie hierzu die vom VS Code angebotene Refactoring-Möglichkeiten

```
translations_de.properties X
flows > Counter > src > main > resources > localisation > translations_de.properties
1 Counter.display.name=Dokumentenzähler
```

```
translations.properties X
flows > Counter > src > main > resources > localisation > translations.properties
1 Counter.display.name=Document Counter
```

4. Tragen Sie folgendes in den Lokalisierungsdateien im Ordner *localisation* ein:

`translations_de.properties` und `translations.properties`

Schlüssel: `Counter.display.name`

Werte: `Dokumentenzähler` und `Document Counter`.

```
6 import com.elo.flows.api.components.annotations.Connection;
7
8 @Component(version = "0.0.1", namespace = "academy.training",
9 name = "Counter", displayName = "Counter.display.name")
10 public class CounterComponent {
11
12     @Connection
13     IXConnection ixConnect;
14 }
15 }
```

5. Erweitern Sie die Projektstruktur um weitere Ebenen.

Eine Beispielumsetzung dieser Anforderung können Sie im [Anhang](#) finden.

ELOas Modul

In dieser Aufgabe werden wir die Funktionalität aus dem ELOas Modul (teilweise) nachimplementieren.

Wir wollen eine Suche (Indexsuche) auslösen, die als Suchparameter die Metadatenmaske und Gruppenfelder definiert.

Die gefundenen Objekte werden in eine angegebene Repository-Struktur referenziert.

Anforderungen (allgemein)

- Angabe einer Metadatenmaske
- Auswahl einer Metadatenmaske aus der Vorschlagsliste
- Angabe von mehreren Feldern, d.h. Name und der Suchwert
- Angabe des Ablagepfades für die Referenzen der gefundenen Objekte
- Bereitstellung eines Triggers (REST)

Die Konfigurationsoberfläche im Tab *Einstellungen*.

The screenshot shows the configuration interface for a search mask named "Vertrag". The interface is divided into several sections:

- Maske:** A dropdown menu showing "Vertrag" with a green checkmark.
- Feld:** A section for defining fields, currently set to "Positionen" with an "Array" button.
- Eingabemethode***: Set to "Positionen".
- Positionen**: A list of search parameters:
 - Name: COMPANY
 - Wert: United Ltd.Each parameter has a green checkmark and a delete icon.
- Referenzpfad**: Set to "ARCPATH:/Contract/Indexsearch".

Teilnehmeraufgabe (optional)

Diese Aufgabe wird nur bearbeitet, wenn von den Teilnehmern Vorschläge zur Umsetzung ganz bestimmter Anforderung gemacht werden.