CMF 3.0 Konfiguration

Zur Konfiguration der CommunityMirror-Software wird das Apache Commons Configuration-Framework verwendet.

(Mit der Neustrukturierung der CommunityMirrorFramework-Sourcen Anfang 2017 wurde von der Version 1.9 auf die Version 2.1 des Frameworks gewechselt.)

https://commons.apache.org/proper/commons-configuration/userguide/user_guide.html

Inhalt dieser Seite:

- Basics
- Konfigurationsdateien im CommunityMirror
    - Technischer Hintergrund
    - Technischer Hintergrund zu getResource()
    - Konfiguration über Eclipse-Projektgrenzen hinweg
- Templates laden

Basics

Über das Configuration-Framework werden aus verschiedenen Quellen Key-Value-Paare wie z.B. "mashup.endpoint" = "mashup/" geladen, welche dann in den Sourcen aus einem Configuration-Objekt abgerufen werden können.

Konkret werden die Konfigurationen aus folgenden Quellen geladen:

Wenn nur das CommunityMirrorFramework genutzt wird (Szenario 1):

- resources/configuration/*
- resources/mirror.properties (oder die beim Start der Anwendung übergebene Datei - in resources gesucht)

Wenn eine CommunityMirror-Anwendung genutzt wird (Szenario 2, 3 und 4):

- resources/configuration/*
- resources/mirror.properties (oder die beim Start der Anwendung übergebene Datei - in resources gesucht)
- Allerdings werden die Dateien zuerst im Anwendungs-Projekt und wenn dort nicht gefunden im CommunityMirrorFramework-Projekt gesucht.

Nachdem die Konfiguration auch auf die Umgebungsvariablen und System-Properties zurückgreift, können auch diese genutzt werden um die Konfigurationen zu setzen - Dabei überschreiben in den Umgebungsvariablen gesetzte Werte solche in den Konfigurationsdateien. In Eclipse können Umgebungsvariablen z.B. in der Run Configuration gesetzt werden (Environment).

Also zum Beispiel einfach im Reiter "Environment" in der Run Configuration definieren:

- mirror.fullscreen = true

Nachdem die Konfiguration auch auf die Umgebungsvariablen und System-Properties zurückgreift, können auch diese genutzt werden um die Konfigurationen zu setzen - Dabei überschreiben in den Umgebungsvariablen gesetzte Werte solche in den Konfigurationsdateien. In Eclipse können Umgebungsvariablen z.B. in der Run Configuration gesetzt werden (Environment).

Konfigurationsdateien im CommunityMirror

Konkret erfolgt die Konfiguration dabei folgendermaßen:

- org.sociotech.communitymirror.CommunityMirror:
    - In der methode start() wird die Methode loadConfiguration() aufgerufen
    - Dort wird als erstes die Umgebungsvariable „communitymirror.configuration.user.file“ auf den ersten Kommandozeilenparameter gesetzt. Ist kein Kommandozeilenparameter beim Aufruf angegeben, dann wird „mirror.properties“ als Defaultwert gesetzt.
    - Dann wird ein CombinedConfigurationBuileder-Objekt erstellt - für den Pfad „configuration/main_configuration.xml“ - und dann die Konfiguration geladen (siehe unten).
    - „main_configuration.xml“ verweist auf verschiedene andere Konfigurationsdateien (siehe unten).

Source-Code zum Laden der Konfiguration (aus CommunityMirror.loadConfiguration()):

```
// create a configuration builder

Parameters params = new Parameters();

CombinedConfigurationBuilder builder = new CombinedConfigurationBuilder()
```

```
                    .configure(params.fileBased().setURL(

                            org.sociotech.communitymirror.CommunityMirror.class.getClassLoader().getResource
(path)

                            ));

// load configuration

configuration = builder.getConfiguration();
```

Datei "main_configuration.xml“ zum Laden der verschiedenen Konfigurationsdateien:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<!-- main_configuration.xml -->
<configuration>
 <!-- allow system variables -->
 <system />
 <!-- allow environment variables -->
 <env />
 <!-- include other configuration files -->
 <additional>
 <!-- include mashup configuration -->
 <xml fileName="mashup_configuration.xml" config-name="mashup" throwExceptionOnMissing="true"/>
 <xml fileName="mirror_configuration.xml" config-name="mirror" throwExceptionOnMissing="true"/>
 <xml fileName="view_configuration.xml" config-name="view" throwExceptionOnMissing="true"/>
 <xml fileName="theme_configuration.xml" config-name="theme" throwExceptionOnMissing="true"/>
 <xml fileName="interaction_configuration.xml" config-name="interaction" throwExceptionOnMissing="true"/>
 <xml fileName="logger_configuration.xml" config-name="logger" throwExceptionOnMissing="true"/>
 <!-- override with user theme parameters -->
 <!-- <properties fileName="../${theme.folder}${theme.name}/theme.properties" config-name="user-theme"
throwExceptionOnMissing="false" /> -->
 <!-- load default theme -->
 <!-- <properties fileName="../${theme.folder}default/theme.properties" config-name="default-theme"
throwExceptionOnMissing="false" /> -->
 </additional>
 <override>
 <!-- override with forced properties, this has priority over user specific settings -->
 <xml fileName="force_configuration.xml" config-name="force" throwExceptionOnMissing="true"/>
 <!-- override with user specific properties -->
 <properties fileName="../${communitymirror.configuration.user.file}" config-name="user" throwExceptionOnMissing="
false" />
 </override>
</configuration>
```

Technischer Hintergrund

In loadConfiguration() wird der Configuration-Bibliothek die URL der Datei „main_configuration.xml“ übergeben, welche über org.sociotech.communitymirror.CommunityMirror.class.getClassLoader().getResource(„configuration/main_configuration.xml“) ermittelt wird.

Die Verwendung von getResource() ist dabei übliche Praxis und führt dazu, dass die Datei relativ zur Wurzel der verschiedenen Einträge im CLASSPATH gesucht wird.

Damit das in Eclipse funktioniert ist folgendes zu beachten:

- Die Dateien und Verzeichnisse im resources-Ordner im Eclipse-Projekt müssen im CLASSPATH auftauchen
- Die einfachste und flexibelste Möglichkeit dazu ist es die Inhalte des resources-Ordners von Eclipse jeweils in das bin-Verzeichnis kopieren zu lassen (welches Standardmäßig im CLASSPATH enthalten ist und auch beim Exportieren in JAR-Files übernommen wird)
- Dies geschieht am einfachsten über die Projekt-Properties in Eclipse: Also beim Projekt (Rechter Mausklick): Properties -> Java Build Path -> Source - und dort Add Folder und dabei das Verzeichnis resources im Projekt auswählen.
- Alternativ könnte in Eclipse auch in der Run Configuration der CLASSPATH ergänzt werden: Run -> Run Configurations -> Classpath -> Select User Entries -> Advanced -> Add Folders. Dann werden die Inhalte allerdings beim Exportieren in JAR-Files nicht übernommen.

Damit das in einem aus Eclipse exportierten ausführbaren JAR-File funktioniert ist folgendes zu beachten:

TBD

Technischer Hintergrund zu getResource()

Wir verwenden zum Initialisieren des Konfigurations-Loaders org.sociotech.communitymirror.CommunityMirror.class.getClassLoader().getResource („configuration/main_configuration.xml“)

Der ClassLoader ist dabei normalerweise

- sun.misc.Launcher$AppClassLoader (bei Ausführung in Eclipse)
- java.net.URLClassLoader (bei Ausführung des aus Eclipse exportierten ausführbaren JAR)

Die Methode getURLs() des ClassLoaders liefert (u.a.)

- - file:/Users/kochm/Development/CMFv2-Test/CommunityMirrorFramework/bin/ (bei der Ausführung in Eclipse)
- - rsrc:./ (bei Ausführung des ausführbaren JARs)

Alternativen: Es könnte alternativ auch getResource() direkt auf dem Class-Objekte aufgerufen werden - Dann ist der Pfad zur Ressource aber explizit absolut zu machen - also „/configuration/main_configuration.xml“ (mit führendem „/„).

Konfiguration über Eclipse-Projektgrenzen hinweg

Die für die Ermittlung der Konfigurationsdateien benutze Methode eignet sich auch dann, wenn die Konfiguration über zwei Eclipse-Projekte verteilt wird (siehe ….):

In dem Fall ist in Eclipse das Kind-Projekt mit dem Eltern-Projekt zu verlinken (TBD Wie). Diese führt dazu, dass bei der Ausführung des Kind-Projektes, das bin-Verzeichnis des Eltern-Projektes in den CLASSPATH aufgenommen wird.

TBD

Templates laden

Neben den Konfigurationsdateien werden im CommunityMirror Template-Dateien geladen.

Nachfolgend der Ablauf für die FXML-Dateien zur Steuerung des Layouts im Kleinen im FlowView (siehe auch die Dokumentation des FlowViews):

- In den konkreten Klassen unter org.sociotech.communitymirror.view.flow.visualitems (z.B. person/DetailPersonItem) ist die Methode render() implementiert
- In dieser Methode wird mit folgendem Sourcecode eine Template-Datei geladen

```
FXMLLoader loader = new FXMLLoader();
URL location;
String fxmlPath = "fxml/inDetail/person.fxml";
if (this.themeResources != null) {
  location = this.themeResources.getResourceURL(fxmlPath);
} else {
  location = Resources.getResource("flow/" + fxmlPath);
  logger.warn(this.getClass().getName() + " using deprecated FXML path.");
}

loader.setLocation(location);
AnchorPane personPane = (AnchorPane) loader.load(location.openStream());
```

Dabei ist themeResources vom Typ org.sociotech.communitymirror.configuration.ThemeResourceLoader

Das ist ein ResourceLoader, welcher mit folgenden Konfigurations-Properties initialisiert wird:

- theme.resourcefolder
- theme.defaultresourcefolder

Zum Laden von Ressourcen wird dort auf die Methode com.google.common.io.Resources.getResource() zurückgegriffen.

Die getResource()-Methode dieser Klasse greift dabei wieder auf die getResource()-Methode des ClassLoaders zurück:

- ClassLoader loader = MoreObjects.firstNonNull(Thread.currentThread().getContextClassLoader(), Resources.class.getClassLoader());
- URL url = loader.getResource(resourceName);