Bearbeitet von:

Peter Grunwald: [pgrunwald\_rostock@freenet.de](mailto:pgrunwald_rostock@freenet.de) Matr. Nr. 6610

Sebastian Brütt: [sebastian.bruett@outlook.com](mailto:sebastian.bruett@outlook.com) Matr. Nr. 6504

Felix Franke: [felixfrank91@web.de](mailto:felixfrank91@web.de) Matr. Nr. 6505

Hausarbeit: MWI2200 Softwareenigneering 2014o

2015

Inhaltsverzeichnis

[Installationsanleitung 1](#_Toc434872136)

[*Vorbedingung:* 1](#_Toc434872137)

[*DSL Projekt* 1](#_Toc434872138)

[*mydslIssueTracker* 1](#_Toc434872139)

[*DSL Customizen* 1](#_Toc434872140)

[Systemarchitektur 2](#_Toc434872141)

[Aufgabe und Strukturierung der DSL 3](#_Toc434872142)

[Glossar 3](#_Toc434872143)

[Issue 3](#_Toc434872144)

[ModelIssue: 3](#_Toc434872145)

[NameObject: 3](#_Toc434872146)

[Status 4](#_Toc434872147)

[Transition 4](#_Toc434872148)

[Workflow 4](#_Toc434872149)

[IssueScreen 4](#_Toc434872150)

[SummaryField 5](#_Toc434872151)

[StatusField 5](#_Toc434872152)

[IssueType 5](#_Toc434872153)

[Role 5](#_Toc434872154)

[Person 6](#_Toc434872155)

[Field 6](#_Toc434872156)

[PersonField 6](#_Toc434872157)

[MailField 6](#_Toc434872158)

[StringField 7](#_Toc434872159)

[ComboField 7](#_Toc434872160)

[DateField 7](#_Toc434872161)

[CheckField 8](#_Toc434872162)

[Weitere Funktionen innerhalb der DSL 8](#_Toc434872163)

[Mehrsprachigkeit 8](#_Toc434872164)

[Validation 8](#_Toc434872165)

[QuickFix 8](#_Toc434872166)

[Formatter 8](#_Toc434872167)

[Templates 8](#_Toc434872168)

[Tests 8](#_Toc434872169)

# Installationsanleitung

## Vorbedingung:

* aktuelle gradle Version installiert
* Gitbash installiert
* aktuelle MongoDB Version installiert und gestartet (https://www.mongodb.org/downloads#production)

## DSL Projekt

1. git clone git@github:alionfried/hausarbeit
2. im Eclipse das Projekt als „existing Project“ importieren
3. /de.nordakademie.ticket/src/de/nordakademie/ticket/Ticket.xtext -> Run As „Generate Xtext Artifacts“

## mydslIssueTracker

1. git clone git@github:alionfried/mydslIssueTracker
2. im mydslIssueTracker Project „gradle e“ ausführen
3. Runetime Eclipse aus DSL Project starten und alle vorhandenen Projekte löschen
4. im Eclipse das Projekt mydslIssueTracker als „existing Project“ importieren
5. /Ticket/src/main/java/de/nordakademie/issuetracker/server/MyDslServer.java Run as „Java Application“

## DSL Customizen

/Ticket/src/main/resources/dsl/spielwiese.mydsl ändern

# Systemarchitektur

View

MongoDB

Dynamisches HTML   
(aus DSL erzeug)

Statisches HTML/Javascript

Generiert HTML für Issuetype Masken

Post

Get

Controller

DSL

Erzeugt Dokumente für:

Personen (inkl. Rollen)

IssueTypen (inkl. möglicher Transitions)

Write Document

Read Document

Sparkjava

Model

# Aufgabe und Strukturierung der DSL

Ziel dieser Arbeit ist ein webbasiertes Ticketsystem, welches eine möglichst variable Ausgestaltung einer Ticketverwaltung ermöglicht und gleichzeitig dem Endanwender durch eine möglichst simple Oberflächengestaltung unterstützt.

Die Variabilität wird dadurch gewährleistet, dass die inhaltlichen Rahmenbedingungen des Ticketsystems vom Administrator mittels einer dafür vorgesehenen DSL festgelegt werden. Lediglich das Anlegen und Bearbeiten der Tickets wird in der Webanwendung durchgeführt. Dadurch hat der Administrator eine möglichst hohe Gestaltungsfreiheit ohne, dass die Endanwender dies spüren.

In seiner Struktur besitzt die DSL einen grundsätzlich hierarchischen Aufbau. Dieser wird im Glossar beschrieben.

# Glossar

Im Folgenden werden alle Komponenten der DSL beschrieben. Dabei besitzen **fett** geschriebene Begriffe einen eigenen Eintrag im Glossar und werden entsprechend dort näher beschrieben. *Kursiv* geschriebene Begriffe stellen KeyWords der DSL dar.

## Issue

Ein Issue (bzw. Ticket) ist nicht Teil der DSL, da die Tickets direkt in der Anwendung von den Anwendern und nicht in der DSL vom Administrator angelegt werden. Die DSL reguliert lediglich, von was für einem Typ (siehe **IssueType**) ein Issue sein kann und wie dieses dementsprechend aufgebaut ist.

Da das Issue nicht Teil der DSL ist, kann es nicht über diese erzeugt werden und besitzt somit auch keine KeyWords.

## ModelIssue:

Das ModelIssue stellt einen DSL-internen Container dar, der alle anderen Komponenten beinhaltet. Er ist für den Administrator, der die DSL nutzt, nicht direkt erkennbar. Alle Komponenten, die er anlegt, sind Teil des ModelIssue. Dieser hierarchische Aufbau wird für die Generierung der restlichen Anwendung intern benötigt und stellt damit eher ein theoretisches Konstrukt dar.

Dementsprechend gibt es auch keine KeyWords zur Anlage des ModelIssue. Allerdings muss das ModelIssue und somit die gesamte Anwendung jede andere Komponente (=Rule) zumindest einmal enthalten. Dies wird durch eine entsprechende Validation sichergestellt.

## NameObject:

NameObject ist eine abstrakte Komponente, von der alle anderen Komponenten erben, welche eine ID (in der DSL = name) besitzen. Auch dies ist eher ein theoretisches Konstrukt, welches unter anderem dafür verwendet wird, sicherzustellen, dass nicht zwei gleichartige Komponenten mit identischer ID erzeugt werden können.

Wie das ModelIssue besitzt auch das NameObject keine KeyWords. Der Grund dafür ist, dass es nicht direkt erzeugt wird, sondern immer eines seiner erbenden Komponenten, welche dann ein NameObject sind.

Folgende Komponenten sind NameObjects:   
**Person, Role , Transition, IssueType, IssueScreen, Workflow, Status, Field**

## Status

Der Status besteht lediglich aus einer ID. Es können beliebig viele Status angelegt werden. Durch **Workflows** und die darin enthaltenen **Transitions** wird festgelegt, welche Status ein **Issue** annehmen kann.

Vom Aufbau innerhalb der DSL wird ein Status durch das KeyWord „*Status*“ und der darauf folgenden ID (=name) erzeugt. Es sind keine weiteren Angaben möglich oder notwendig.

## Transition

Die Transitions beschreiben den Wechsel von einem **Status** zu einem anderen **Status**. Sie bilden das Kernstück des Issue-**Workflows** sowie des Berechtigungskonzepts (siehe **Rollen)**. Eine Transition besteht somit aus einem Start- sowie einem End-**Status**. Zusätzlich hat er neben einer ID auch einen Titel, der in der späteren Anwendung als einzelner **Workflow**schritt angezeigt und soweit ein Anwender (siehe Person) dazu berechtigt ist, angestoßen werden kann.

Vom Aufbau innerhalb der DSL wird eine Transition durch das KeyWord „*Transition*“ und einer darauf folgenden ID (=name) erzeugt. Nach der ID folgt das KeyWord „*named*“ und dem später angezeigten Titel. Danach folgt eine geschweifte, öffnende Klammer („*{*“) worauf dann die verpflichtende Referenzierung auf einen Start- und einen End-**Status** erfolgen muss. Die beiden Status sind durch das KeyWord „to“ getrennt. Die beiden Status müssen voneinander verschieden sein, da ein Statuswechsel zum selben Staus in einem **Workflow** nicht sinnvoll ist.

Zuletzt wird die Definition der Transition mit einer geschweiften, schließenden Klammer („*}*“) beendet.

## Workflow

Der Workflow definiert, welche **Status** ein **Issue**, der diesem Workflow folgt, in welcher Reihenfolge durchlaufen kann. Dies wird durch beliebig viele **Transitions**, auf die ein Workflow referenziert, realisiert. Ein Workflow muss zumindest eine **Transition** beinhalten.

Vom Aufbau innerhalb der DSL wird ein Workflow durch das KeyWord „*Workflow*“ und einer darauf folgenden ID (=name) erzeugt. Nach der ID folgt eine geschweifte, öffnende Klammer („*{*“) worauf dann auf beliebig viele, aber zumindest eine, **Transition** referenziert werden kann.

Zuletzt wird die Definition des Workflow mit einer geschweiften, schließenden Klammer („*}*“) beendet.

## IssueScreen

Der IssueScreen ist die einzige Komponente, welche lediglich einmalig erzeugt werden kann und muss. Inhaltlich legt er fest, welche Felder jedes später in der Anwendung erzeugte Ticket unabhängig davon, um welche Art von Ticket es sich dabei handelt (siehe **IssueType**), auf jeden Fall besitzt.

Festgelegt ist, dass es zumindest eine Kurzbeschreibung (siehe **SummaryField**) und einen Anfangsstatus (siehe **StatusField**) besitzt. Alle weiteren Felder sind beliebig vom Administrator hinzufügbar.

Vom Aufbau innerhalb der DSL wird ein IssueScreen durch das KeyWord „*IssueScreen*“ und einer darauf folgenden ID (=name) erzeugt. Nach der ID folgt eine geschweifte, öffnende Klammer („*{*“) worauf dann die verpflichtende Definition eines **SummaryFields** sowie eines **StatusField** erfolgen muss. Hinter dem **StatusField** können beliebig viele andere Fields referenziert werden, indem deren ID genannt wird.

Zuletzt wird die Definition des IssueScreen mit einer geschweiften, schließenden Klammer („*}*“) beendet.

### SummaryField

Ein SummaryField stellt ein Kurzbeschreibungsfeld eines jeden **Issue** dar und ist deshalb fester Teil des **IssueScreen**.

Vom Aufbau innerhalb der DSL wird ein SummaryField durch das KeyWord „*SummaryField*“ und einer darauf folgenden ID (=name) erzeugt. Darauf folgt eine geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert als String angegeben wird.

Zuletzt wird die Definition des SummaryField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

### StatusField

Ein StatusField stellt ein StatusFeld eines jeden **Issue** dar und ist deshalb fester Teil des **IssueScreen**. Hier wird festgehalten, in welchem **Status** sich ein **Issue** gerade befindet. Der Default**status** definiert, welchen Status ein **Issue** nach dem Anlegen besitzt.

Vom Aufbau innerhalb der DSL wird ein StatusField durch das KeyWord „*StatusField*“ und einer darauf folgenden ID (=name) erzeugt. Darauf folgt eine geschweiften, öffnenden Klammer („*{*“) worauf dann ein Default**status** definiert werden muss, indem hinter einem KeyWord „*default*“ auf einen **Status** referenziert wird.

Zuletzt wird die Definition des StatusField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

## IssueType

Ein IssueType definiert, von was für einem Typ ein **Issue** sein kann. Das bedeutet, er legt fest, welchem **Workflow** der **Issue** folgt und welche Felder (siehe **Fields**) der **Issue** neben den bereits im **IssueScreen** festgelegten Feldern zusätzlich besitzt.

Vom Aufbau innerhalb der DSL wird ein IssueType durch das KeyWord „*IssueType*“ und einer darauf folgenden ID (=name) erzeugt. Nach der ID folgt das KeyWord „*follows*“ und der Referenzierung auf einen **Workflow**, dem jeder **Issue** dieses IssueTypes folgt. Danach steht eine geschweifte, öffnende Klammer („*{*“) worauf dann die freiwillige Referenzierung auf beliebig viele Felder (siehe **Fields**) folgt.

Zuletzt wird die Definition des IssueType mit einer geschweiften, schließenden Klammer („*}*“) beendet.

## Role

Eine Role ist ein Berechtigungscontainer von **Transitions**. Durch Rollen wird definiert, welche Workflowaktionen eine **Person** durchführen darf. Workflowaktionen bestehen entweder aus dem Anstoßen einer **Transition**, um den Status eines bestehenden **Issue** zu verändern oder dem Anlegen eines neuen **Issue**. Letzteres wird nicht über **Transitions** sondern über das Setzen eines bool’schen Wertes innerhalb der Rolle erreicht.

Eine Rolle muss zumindest eine Workflowaktion beinhalten.

Vom Aufbau innerhalb der DSL wird eine Role durch das KeyWord „*Role*“ und einer darauf folgenden ID (=name) erzeugt. Nach der ID folgt eine geschweifte, öffnende Klammer („*{*“) worauf dann die entweder die Definition des bool’schen Wertes, der das Erzeugen eines neuen **Issue** erlaubt, und/oder die Referenzierung auf beliebig viele **Transitions** folgt. Das Setzen des bool’schen Wertes erfolgt durch die KeyWords „*create Issue*“. Auch wenn sowohl dies als auch das Referenzieren auf **Transitions** optional ist, muss zumindest eine der beiden Möglichkeiten genutzt werden.

Zuletzt wird die Definition der Role mit einer geschweiften, schließenden Klammer („*}*“) beendet.

## Person

Eine Person definiert einen Endanwender der Webanwendung. Eine Person besitzt neben einer eindeutigen ID auch einen angezeigten Namen.

Einer Person können beliebig viele Rollen (siehe **Roles**) zugeordnet sein, welche sie zu bestimmten Aktionen berechtigt.

Vom Aufbau innerhalb der DSL wird eine Person durch das KeyWord „*Person*“ und einer darauf folgenden ID (=name) erzeugt. Nach der ID folgt das KeyWord „*called*“ sowie dem angezeigten Namen als String. Dahinter steht eine geschweifte, öffnende Klammer („*{*“) worauf dann auf beliebig viele Rollen referenziert werden kann.

Zuletzt wird die Definition der Person mit einer geschweiften, schließenden Klammer („*}*“) beendet.

## Field

Ein Field ist eine abstrakte Oberklasse, von der verschiedene Eingabefelder, die ein Issue besitzen kann, erben. Alle Fields besitzen eine eindeutige ID, eine Bezeichnung, sowie einen optionalen Defaultwert. Welchen Typ dieser Defaultwert besitzt, hängt von der Art des Feldes ab (siehe unten).

Da ein Field eine abstrakte Komponente ist, kann diese nicht direkt angelegt werden, sondern wird durch die Definition eines konkreten Feldes erzeugt (dieses konkrete Feld ist dann auch ein Field).

**StatusField** und **SummaryField** erben NICHT von Field, da diese aufgrund ihrer besonderen Stellung im **IssueScreen** keiner Beschreibung bedürfen und zudem nur einmalig definiert werden dürfen (es kann keine zwei Statusfelder geben).

### PersonField

Ein PersonField referenziert als Wert immer auf eine **Person**.

Vom Aufbau innerhalb der DSL wird ein PersonField durch das KeyWord „*PersonField*“ und einer darauf folgenden ID (=name) erzeugt. Wie bei allen Fields folgt danach das KeyWord „as“ und die Bezeichnung des Feldes und nachfolgend einer geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert angegeben wird.

Der Defaultwert eines PersonField besteht immer aus einer Referenz auf eine **Person.**

Zuletzt wird die Definition des PersonField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

### MailField

Ein MailField besitzt als Wert immer auf eine String in Form einer Mailadresse.

Vom Aufbau innerhalb der DSL wird ein MailField durch das KeyWord „*MailField*“ und einer darauf folgenden ID (=name) erzeugt. Wie bei allen Fields folgt danach das KeyWord „as“ und die Bezeichnung des Feldes und nachfolgend einer geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert angegeben wird.

Der Defaultwert eines MailField besteht immer aus ein String in Form einer Mailadresse**.**

Zuletzt wird die Definition des MailField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

### StringField

Ein StringField besitzt als Wert immer einen String.

Vom Aufbau innerhalb der DSL wird ein StringField durch das KeyWord „*StringField*“ und einer darauf folgenden ID (=name) erzeugt. Wie bei allen Fields folgt danach das KeyWord „as“ und die Bezeichnung des Feldes und nachfolgend einer geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert angegeben wird.

Der Defaultwert eines StringField besteht immer aus einem String**.**

Zuletzt wird die Definition des StringField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

### ComboField

Ein ComboField beinhaltet als Werte eine Liste von Strings, von denen einer (im Standard immer der erste) ausgewählt werden kann.

Vom Aufbau innerhalb der DSL wird ein ComboField durch das KeyWord „*ComboField*“ und einer darauf folgenden ID (=name) erzeugt. Wie bei allen Fields folgt danach das KeyWord „as“ und die Bezeichnung des Feldes und nachfolgend einer geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert angegeben wird.

Der Defaultwert eines ComboField besteht immer aus einer Liste von Stringeinträgen**.**

Zuletzt wird die Definition des ComboField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

### DateField

Ein DateField besitzt als Wert immer auf ein Datum (siehe **Date**).

Vom Aufbau innerhalb der DSL wird ein DateField durch das KeyWord „*DateField*“ und einer darauf folgenden ID (=name) erzeugt. Wie bei allen Fields folgt danach das KeyWord „as“ und die Bezeichnung des Feldes und nachfolgend einer geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert angegeben wird.

Der Defaultwert eines DateField besteht immer aus einem Datum (siehe **Date**).

Zuletzt wird die Definition des DateField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

#### Date

Ein Date beschreibt ein Datum und besteht aus einem Tag, einem Monat und einem Jahr. Es ist immer Teil eines **DateField**. Diese bestehen jeweils aus einer Zahl, sind laut DSL durch einen Punkt „.“ getrennt und werden entsprechend validiert.

### CheckField

Ein CheckField besitzt als Wert immer einen bool’schen Wert.

Vom Aufbau innerhalb der DSL wird ein CheckField durch das KeyWord „*CheckField*“ und einer darauf folgenden ID (=name) erzeugt. Wie bei allen Fields folgt danach das KeyWord „as“ und die Bezeichnung des Feldes und nachfolgend einer geschweiften, öffnenden Klammer („*{*“) worauf dann ein Defaultwert definiert werden kann (optional), indem hinter einem KeyWord „*default*“ der Defaultwert angegeben wird.

Der Defaultwert eines CheckField besteht immer aus einem bool’schen Wert (entweder „*FALSE*“ oder „*TRUE*“).

Zuletzt wird die Definition des CheckField mit einer geschweiften, schließenden Klammer („*}*“) beendet.

# Weitere Funktionen innerhalb der DSL

## Mehrsprachigkeit

Die Sprache der Validation- und QuickFix-Meldungen lassen sich durch Ein- und Auskommentieren der implementierten Interfaces in den beiden entsprechenden Klassen zwischen deutsch und englisch variieren.

## Validation

Um die Qualität der vom Administrator geschriebenen Konfiguration zu verbessern, wurden verschiedenste Validationen auf dessen Eingaben durchgeführt. Diese sind in der Klasse „TicketValidator.xtend“ definiert.

## QuickFix

Zu fast jeder kundeneigenen und zu einigen Standard-Validation existiert zumindest ein QuickFix, um die Bedienbarkeit der DSL für den Administrator zu verbessern. Dies ist in der Klasse „TicketQuickfixProvider.xtend“ definiert.

## Formatter

Um die Funktionalität eines eigenen Formatters zu nutzen, wurden für folgende Komponenten und (falls vorhanden) ihre Unterkomponenten die Standardformatierung angepasst: Person, Date sowie IssueScreen.

Die Formatierung des IssueScreen wurde mit Hilfe eines FormatterTests validiert.

## Templates

Um die Bedienbarkeit der DSL für den Administrator zu verbessern, wurde für jede Komponente zumindest ein Template zu dessen Erzeugung definiert. Diese beginnen immer mit „new<Komponentenname>“ und enden teilweise mit einem zusätzlichen Suffix.

## Tests

Um die Qualität der Validation zu gewährleisten, wurden zu jeder Validation mehrere Tests in der Klasse „ValidationTest.xtend“ implementiert. Zusätzlich existiert ein Formatter-Test in der Klasse „FormatterTest.xtend“ und einige Syntax-Tests für ComboFields in der Klasse „SyntaxTest.xtend“.