Personal Kanban with Jira

Persönliches Zeit- und Aufgabenmanagement

Michael Schlüter | Atlassian User Group Ruhrgebiet

Agenda

Ursprung und Grundlagen

Methodik

Umsetzung mit Jira

Ursprung und Grundlagen

Ursprung

Grundlagen

Literatur

Ursprung und Geschichte

Ursprung
Entwickelt 1947 von Taiichi Ohno bei Toyota
Kanban かんばん（看板）bedeutet Karte, Tafel oder Beleg
Kanban basiert auf dem „Pull-Prinzip“

Ziele
Optimierung des Materialflusses
Optimierung der Durchflussgeschwindigkeit

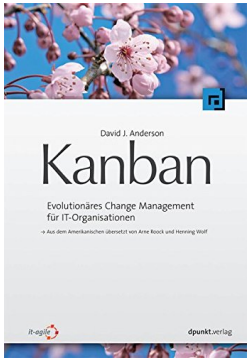
Grundlagen

Kapazität vs. Durchsatz
Nicht: Wie viele Aufgaben kann ich bearbeiten?
Sondern: Wie viel Zeit benötige ich, um eine Aufgabe zu erledigen?

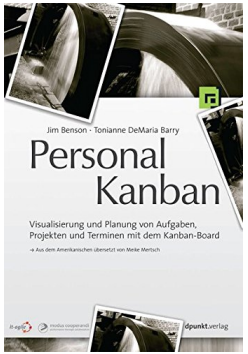
Analogie Autobahn
3-spurige Autobahn: 1000m * 11m = 11000m² Kapazität
Kapazitätsbedarf eines Autos 4,50m * 2,10m = 9,45m²
Optimale Ausnutzung der Kapazität = Stau
Durchsatz ~ 0

Literatur

Kanban in der Softwareentwicklung
Agile Methode
Vorgestellt 2007 von David Anderson
Schwerpunkt Software-Wartung / -Support
ISBN 978-3898647304



Personal Kanban
Jim Benson, Tonianne DeMaria Barry
Aufgaben-, Zeitmanagement für Einzelpersonen
oder kleine Teams
ISBN 978-3898648226



Methodik

Kanban

Arbeit visualisieren

Work in Progress begrenzen

Arbeit visualisieren

Motivation
Visualisierung entlastet das Gehirn
Erleichtert erkennen von Mustern und Zusammenhängen
Optischer Reiz beim Vollenden von Aufgaben

Auswertung
Was habe ich in der Woche/Monat eigentlich getan/geschafft?
Welche Aufgaben gehen schnell (subjektiv, objektiv)
Welche Aufgaben schiebe ich vor mir her?

Kanban

Arbeit visualisieren

Work in Progress begrenzen

Work in Progress begrenzen

Das menschliche Gehirn ist nicht Multitasking-fähig.

Das Gehirn merkt sich unterbrochene Aufgaben besser als abgeschlossene. (Zeigarnik-Effekt)
Unterbrochene Aufgaben beschäftigen das Gehirn im Unterbewusstsein
Jede laufende Aufgabe reduziert die verfügbare Arbeitsleistung

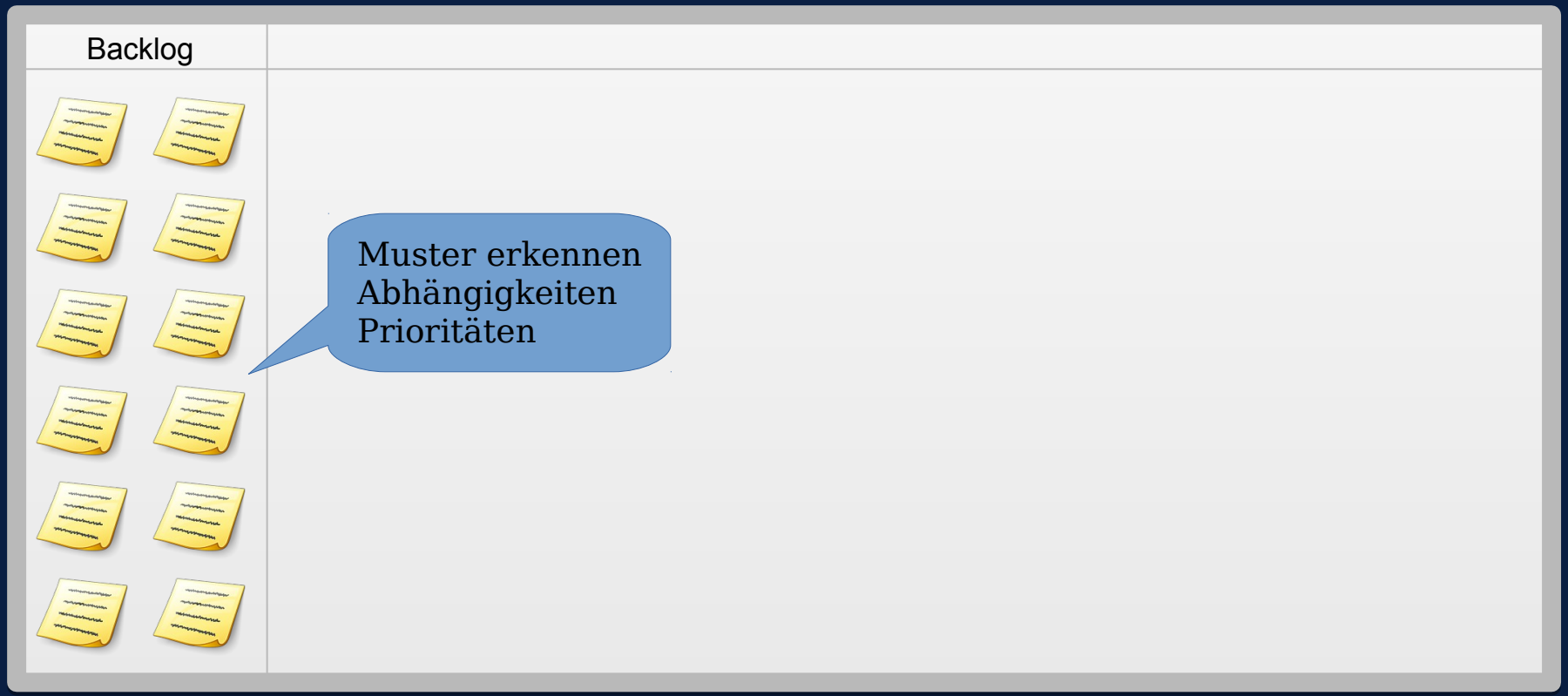
Kontextwechsel zwischen Aufgaben sind sehr zeitaufwändig (min. 20 Minuten)

Je mehr Aufgaben parallel bearbeitet werden, desto länger dauert die durchschnittliche Bearbeitung einer Aufgabe (Durchlaufzeit)

Je länger die Bearbeitung einer Aufgabe dauert, desto größer ist die Wahrscheinlichkeit, dass das Ergebnis unpassend oder irrelevant ist.

Umsetzung

Arbeit visualisieren



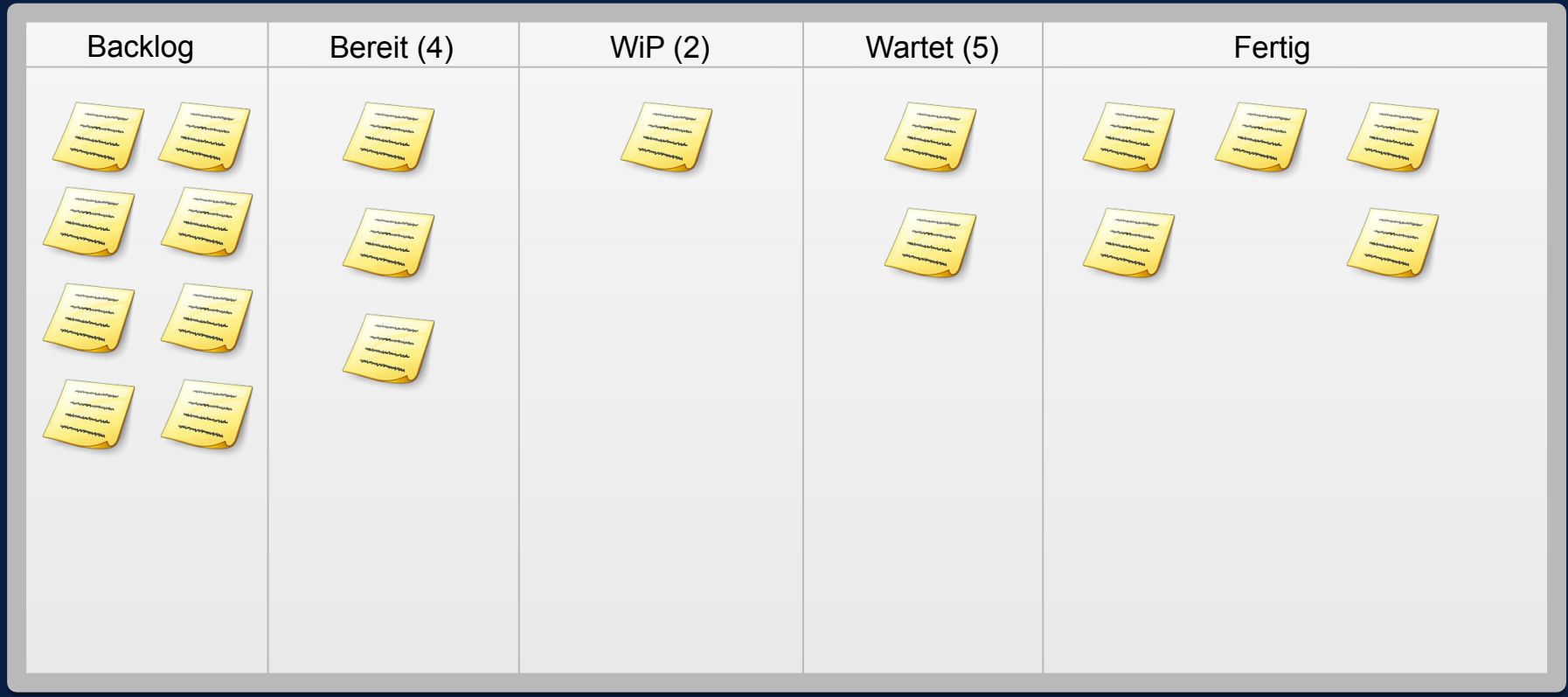
Work in Progress begrenzen

Backlog	WiP (2)	
		

Aufgaben fertigstellen

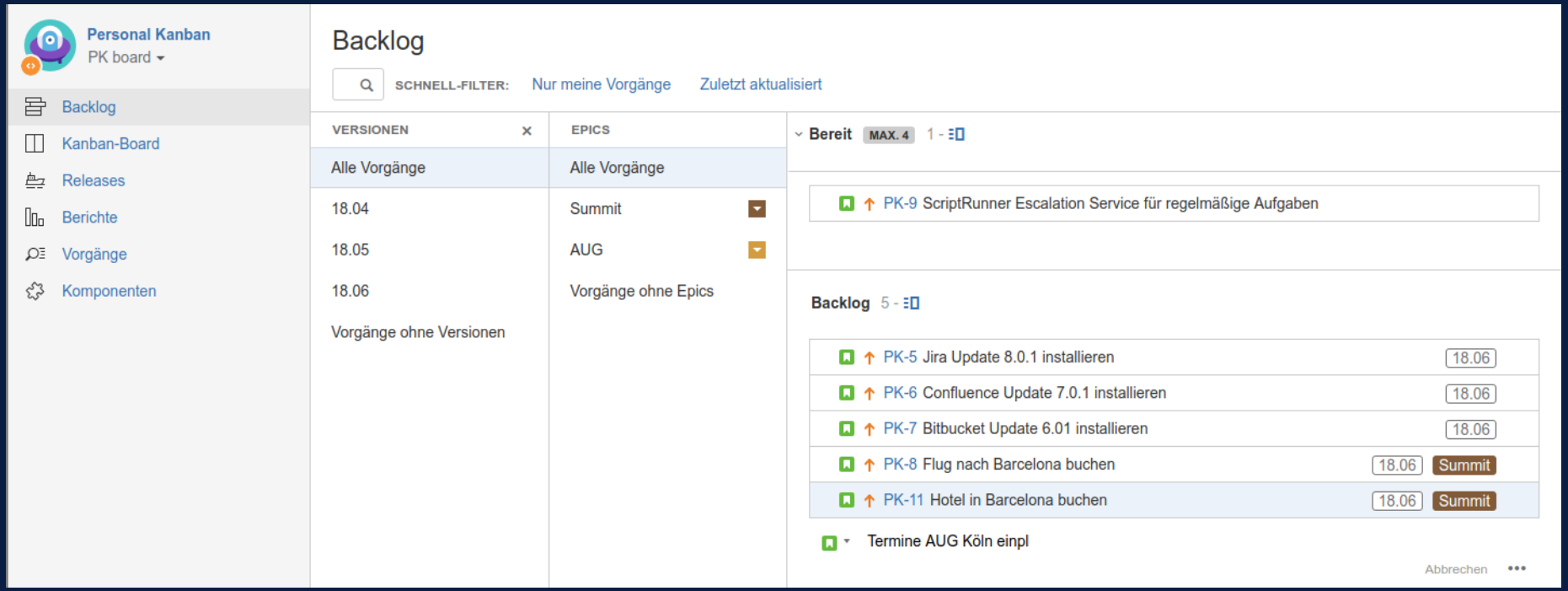
Backlog	WiP (2)	Fertig	
			

Customizing

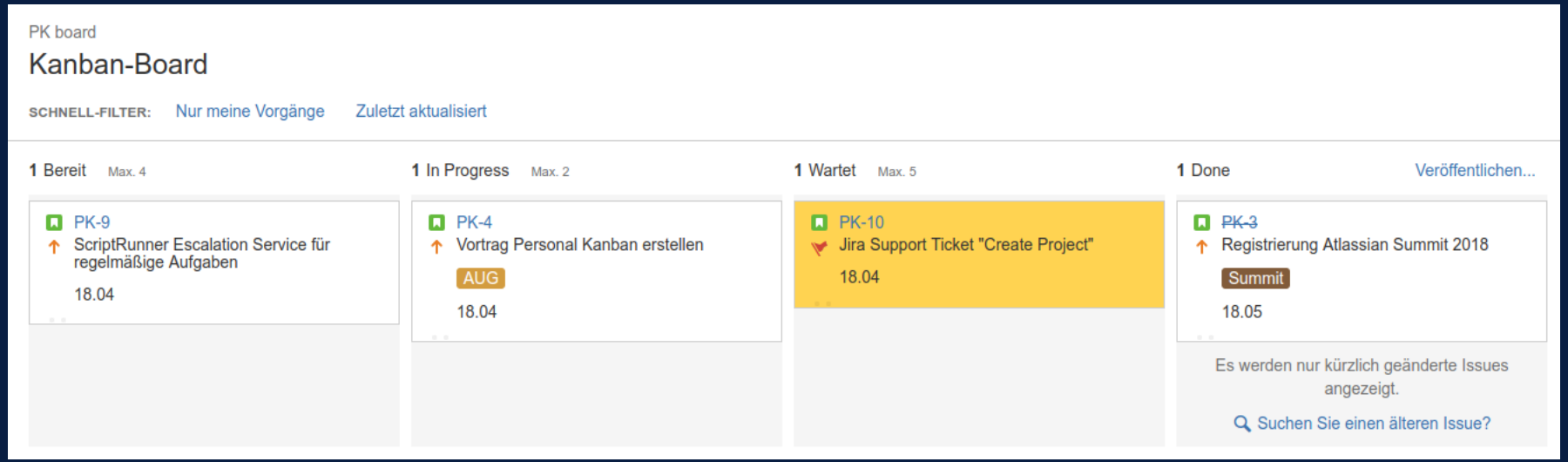


Umsetzung in Jira

Personal Kanban - Backlog



Personal Kanban - Board



Jira Konfiguration

Konfiguration

- **Issue Security**
- Versionierung
- Murmeltiertag
- Escalationservice

Issue Security

Issue Security Schema

Alle Benutzer teilen sich das gleiche Personal Kanban Projekt
Jeder Benutzer sieht nur seine eigenen Aufgaben

Security Level	Description	Users / Groups / Project Roles
Privat DEFAULT	All issued are visible to the reporter and assignee only.	• Current assignee • Reporter

Default-Assignee

Workflow Post-Function setzt Autor als Bearbeiter

Konfiguration

Issue Security

Versionierung

Murmeltiertag

Escalationservice

Versionierung

Versionsschema

Monatliche Version nach dem Schema *Jahr.Monat* (z.B: 18.04)
Lösungsversion wird automatisch beim Abschluss des Vorgangs gesetzt

Monatlicher Service
Automatisches Erzeugen neuer Versionen
Automatisches „Release“ der aktuellen Version

Service: Lösungsversion erstellen

```
/** Diese Methode erstellt eine neuen Version in dem Projekt
 * @param month Monat innerhalb des Jahres in Java Notation (Calendar.JANUARY = 0 Calendar.December = 11)
 * @param year  Jahreszahl (vierstellig)
 * @see java.util.Calendar
 */
void createVersion(int month, int year){
    VersionService versionService
    versionService = ComponentAccessor.getComponent(VersionService)

    // Prepare start and release date of the version based on the given moth an year
    Calendar cal = Calendar.getInstance();
    cal.set(Calendar.YEAR, year)
    cal.set(Calendar.MONTH, month)
    cal.set(Calendar.DAY_OF_MONTH, 1)
    Date startDate = cal.getTime();
    cal.set(Calendar.DAY_OF_MONTH, cal.getActualMaximum(Calendar.DAY_OF_MONTH))
    Date releaseDate = cal.getTime();

    def versionName = sprintf('%02d.%02d', year-2000, month+1)

    VersionBuilder versionBuilder = versionService.newVersionBuilder()
    versionBuilder.projectId(m_project.getId())
    versionBuilder.name(versionName)
    versionBuilder.description("Monat $versionName")
    versionBuilder.startDate(startDate)
    versionBuilder.releaseDate(releaseDate)

    VersionService.VersionBuilderValidationResult validationResult = versionService.validateCreate(m_user, versionBuilder)
    if (validationResult.isValid()) {
        // Fehler werden ignoriert, es ist wahrscheinlich, dass Versionen schon existieren
        versionService.create(m_user, validationResult)
    } else {
        m_log.info "Erzeugen einer neuen Version fehlgeschlagen"
        m_log.error("Validation Result: " +validationResult.getErrorCollection().getErrorMessages())
        m_log.error(validationResult.getErrorCollection().getErrors())
        m_log.error("User: " + m_user.getName() + " " + m_user.getKey() + " " + m_user.toString())
    }
}
```

Service: Release aktuelle Version

```
/*
 * Verschiebe Vorgänge aus fälligen Version in die nächst nicht fällige Version
 */
void moveIssuesFromDueVersions(){
    JiraServiceContextImpl jiraServiceContextImpl = new JiraServiceContextImpl(m_user);

    m_log.info "Verschiebe Vorgänge von abgelaufenen Versionen"

    VersionService versionService
    versionService = ComponentAccessor.getComponent(VersionService)
    Collection<Version> dueVersions       = this.getDueVersions()
    Collection<Version> nonDueVersions       = this.getNonDueVersions()
    Version nextUnreleasedVersion = nonDueVersions[0]

    dueVersions.each {
        // Verschiebe alle offenen Issues zur aktuellen Version
        versionService.moveUnreleasedToNewVersion(m_user, it, nextUnreleasedVersion)
        VersionService.ReleaseVersionValidationResult validationResult = versionService.validateReleaseVersion(m_user, it, it.getReleaseDate());
        if (validationResult.isValid()) {
            versionService.releaseVersion(validationResult)
        }

    }

}
```

Konfiguration

Issue Security

Versionierung

Murmeltiertag

Escalationservice

Escalation service

Can periodically modify issues based on a JQL query, for instance: to change the state of issues if they have been inactive for 2 weeks.

Description: Reopen Monthly

Description - this field is used for your reference only.

JQL Query: project in ("Personal Kanban") and status = done and labels=monthly

JQL query to select issues with. For example "issuetype = Task and updated < -7d"

As User: Escalation Service

The user that the service will run as

Interval/CRON Expression: 0 0 1 1 * ?

How often the service should run, in minutes. If not a number, CRON expression is expected

Action: Backlog (11)

Optionally set an action...

Additional issue actions

SCRIPT | FILE

```
1  // Define empty Collection of Fix Versions
2  Collection fixVersionscollection   = new HashSet();
3  // Set the empty Collection to the Fix Version field on the cloned issue
4  issue.fixVersions = fixVersionscollection
```

Expand examples

Enter any customisations to the target issue, e.g. hard-coding specific field values.

Transition Options

- [x] Skip Permissions
- [x] Skip Validators
- [x] Skip Conditions

Allow the transition to skip certain checks

Konfiguration

Issue Security

Versionierung

Murmeltiertag

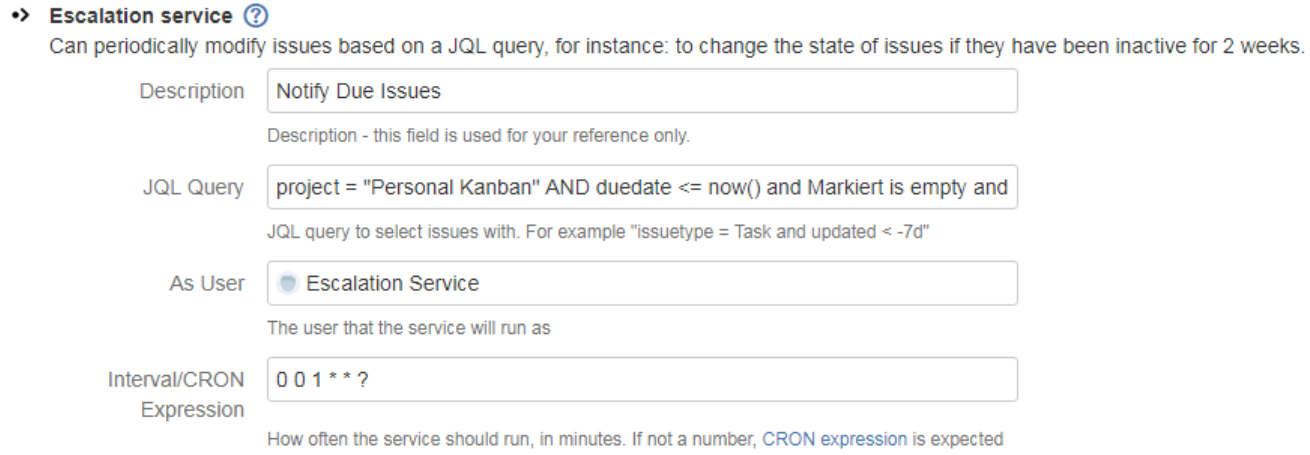
Escalationservice

Escalationservice

Fälligkeitsdatum erreicht

E-Mail-Benachrichtigung

Markierung der Karten auf dem Kanban Board



Service: Abgelaufene Vorgänge

```
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.event.issue.IssueEventBundleFactory
import com.atlassian.jira.event.issue.IssueEvent

def customFieldManager = ComponentAccessor.getCustomFieldManager()
def customField = customFieldManager.getCustomFieldObjectByName("Markiert") // name of CF

def optionsManager = ComponentAccessor.getOptionsManager()
def issueEventManager = ComponentAccessor.getIssueEventManager()
def issueEventFactory =  ComponentAccessor.getComponent(IssueEventBundleFactory.class)

def fieldConfig = customField.getRelevantConfig(issue)
def option = optionsManager.getOptions(fieldConfig).find { it.value == "Hindernis" } // value of option

issueInputParameters.addCustomFieldValue(customField.id, option.optionId as String)
issueInputParameters.setSkipScreenCheck(true)
issueInputParameters.setComment('Fälligkeitsdatum erreicht!')

def eventBundle = issueEventFactory.wrapInBundle(new IssueEvent (issue, null, currentUser, 10001l, true))
issueEventManager.dispatchEvent(eventBundle)
```

Thank you

Personal Kanban with Jira

Persönliches Zeit- und Aufgabenmanagement

Michael Schlüter | Atlassian User Group Ruhrgebiet | michael.schlueter@posteo.de