# Inhaltsverzeichnis

[1. Inhaltsverzeichnis 1](#_Toc425186661)

[2. Tomcat 3](#_Toc425186662)

[2.1. Tomcat – Db Konfiguration 3](#_Toc425186663)

[3. Prozesse 4](#_Toc425186664)

[3.1. Branch erzeugen 4](#_Toc425186665)

[3.1.1. Info-Mail 4](#_Toc425186666)

[3.1.2. Version 4](#_Toc425186667)

[3.2. Release Information pflegen 5](#_Toc425186668)

[3.3. Releaseletter 5](#_Toc425186669)

[4. Entwicklungsumgebung 6](#_Toc425186670)

[4.1. Installation 6](#_Toc425186671)

[4.1.1. Java 6](#_Toc425186672)

[4.1.2. Eclipse IDE for Java EE Developers 6](#_Toc425186673)

[4.1.3. MySQL 6](#_Toc425186674)

[4.2. Build 6](#_Toc425186675)

[4.2.1. svn 6](#_Toc425186676)

[4.2.2. MySQL 7](#_Toc425186677)

[4.2.3. eclipse 7](#_Toc425186678)

[4.3. Deploy 9](#_Toc425186679)

[4.3.1. Frontend Cleanup 9](#_Toc425186680)

[4.3.2. Backend Cleanup 9](#_Toc425186681)

[4.3.3. Frontend: Copy auf Liferay 9](#_Toc425186682)

[4.3.4. Backend: Copy auf Tomcat 9](#_Toc425186683)

[4.4. Run 9](#_Toc425186684)

[4.4.1. Tomcat 9](#_Toc425186685)

[4.4.2. Liferay 9](#_Toc425186686)

[5. PräProduktion 10](#_Toc425186687)

[5.1. Build 10](#_Toc425186688)

[5.1.1. Jenkins-Build 10](#_Toc425186689)

[5.1.2. Versionsnummern 10](#_Toc425186690)

[5.1.3. Datenbank 10](#_Toc425186691)

[5.1.4. Flyway Probleme beim Jenkins deployment 10](#_Toc425186692)

[5.2. Partielles Build (Hotfix) 10](#_Toc425186693)

[6. Produktion 11](#_Toc425186694)

[6.1. Checkliste 11](#_Toc425186695)

[6.2. Build 11](#_Toc425186696)

[6.2.1. Datenbank 11](#_Toc425186697)

[7. Merge 12](#_Toc425186698)

[7.1. Neuen Branch auschecken 12](#_Toc425186699)

[8. Betrieb 13](#_Toc425186700)

[8.1. ActiveMQ neustarten 13](#_Toc425186701)

[9. Skripte 14](#_Toc425186702)

[9.1. Code Count 14](#_Toc425186703)

[9.1.1. Anzahl Java Klassen 14](#_Toc425186704)

[9.1.2. Anzahl Java Codezeilen 14](#_Toc425186705)

[9.1.3. Anzahl XML Files 14](#_Toc425186706)

[9.1.4. Anzahl XML Zeilen 14](#_Toc425186707)

# Tomcat

## Tomcat – Db Konfiguration

/opt/apache-tomcat/conf/context.xml

# Prozesse

## Branch erzeugen

Nach Teststellung auf PräProd.

### Info-Mail

Anschließend Info-Mail an [agn-dev@ecotel.de](mailto:agn-dev@ecotel.de):

Hi team,

We forked a separate SVN branch for **Release x.y**: <branch link>

Till we plan to deploy to production from this branch on <date> and the test team has to test on pre-production, **please do not commit any code changes to this branch without getting an approval from <person>**.

Features for Release x.y+1, planned production <date>, have to be committed ONLY to the trunk.

Current branch overview:

* **SVN Branch Release x.y-1  (PRODUCTION!!)**
  + <branch link>
  + This is ONLY for high prioritized hotfixes for production !!
  + SVN commit approval: <person>
* **SVN Branch Release x.y**
  + <branch link>
  + This is only for fixes and features which have to deployed to production on <date>
  + SVN commit approval: <person>
* **SVN trunk Release x.y+1**
  + <branch link>
  + release x.y+1, commit ALL changes which are committed in the other branches into this branch as well
  + No approval necessary

Thank you for the compliance of these rules.

Regards,

<Signature>

### Version

Danach Update der Version im pom auf nächste Release-Nummer.

## Release Information pflegen

parentpom > release\_information > Rx\_y\_z.txt enthält:

* svn branch
* Planned Production Date
* Applications to be deployed
* Database changes to be applied
* Config settings to be applied
* …

## Releaseletter

Deployment Manager pflegt Scan der Checkliste und Mail zur Veröffentlichung:

1. Speichern unter P:\02\31\08\AGN
2. Update im Wiki

# Entwicklungsumgebung

## Installation

### Java

* Path Variable setzen

### Eclipse IDE for Java EE Developers

* Subclipse
* m2e (ggf. schon in obiger Version enthalten)
* Auschecken: svn View > AGN > trunk > 56 Projekte (in eigenen Workspace \Workspaces\AGN\trunk)
* Settings
  + General > Workspace > Text file encoding > Other: UTF-8
  + Java > Installed JREs > akt. JDK hinzufügen, altes entfernen
  + Settings aus svn hinzufügen: svn\boss\_ng\runtime\eclipse\ecotel\_preferences\eclipse\_preferences.epf
  + Checkstyle Prefs hinzufügen: svn\boss\_ng\runtime\eclipse\ecotel\_preferences\checkstyle.xml
  + Maven Pfad einstellen: Maven > Installations > Add

### MySQL

* Neue Dbs einrichten (jeweils Collation: utf8\_general\_ci)
* HeidiSQL > Werkzeuge > Benutzerverwaltung > Neu
  + Name/Passwort angeben
  + Zugriff auf: alles

#### Pfade zu den Dbs

1. C:\Program Files\MySQL\MySQL Server 5.6\data
   * wird genutzt von bin\mysqld.exe
2. C:\ProgramData\MySQL\MySQL Server 5.6\data
   * wird genutzt vom Dienst

## Build

### svn

#### Vorschau (Optional)

Anzeige aller Changes, ohne tatsächlichem Commit/Update

1. Umgebung “Java EE”
2. Alle Projekte markieren > Rechtsklick > Team > Synchronize with Repository

|  |  |
| --- | --- |
|  | Compile-Fehler; funktioniert nicht 🡪 ignorieren |
|  | lokale Änderungen |
|  | Konflikte |
|  | Update im Repository |
|  | Neu im Repository |
|  | Im Repository gelöscht |

#### Update/Commit

Falls Vorschau gewählt wurde: Umgebung “Team Synchronizing”

1. Anzeige aller Updates/Commits
2. Alles auswählen (Strg+A) > Rechtsklick > Update

Falls keine Vorschau gewählt wurde: Umgebung „Java EE“

1. Alles auswählen (Strg+A) > Rechtsklick > Team > Update to HEAD

#### Switch to another Branch/Tag/Revision

Im Untermenü von Team zu finden.

Projekte mit externen Ressourcen werden nicht sauber geswitcht (betrifft derzeit 8 Projekte). Lösung für die jeweiligen Projekte:

* Team > Disconnect… (Do not delete…)
* Team > Share Project… (SVN > Finish)

### MySQL

* Starten durch **C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld.exe**
* Wird bereits beim Build benötigt, da Persistenz-Schicht Objekte aus Datenbank generiert.

### eclipse

1. Falls com.google.code.maven-svn-revision-number-plugin:svn-revision-number-maven-plugin:1.13 mit der Fehlermeldungsvn: E155021: This client is too old to work with the working copy quittiert, muss im Plugin vor dem Excecution-Element die folgende Dependency hinzugefügt warden:

<dependencies>

<dependency>

<groupId>org.tmatesoft.svnkit</groupId>

<artifactId>svnkit</artifactId>

<version>1.8.3-1</version>

</dependency>

</dependencies>

1. Wenn ein Typ nicht gefunden wird, kann helfen
   * Die generierten Sourcen in im Projekt einzubinden, z.B.
     + Projekt agn.aef.persistence > Rechtsklick > New > Source Folder > Foldername Browse… > target\generated-sources\**hibernate3**
     + **Wichtig: nicht generated-sources, sondern hibernate3 einbinden**
   * die zugeh. Persistence- bzw. Schema-Schicht neu bauen, dann dort F5
   * Project > Clean…
   * Rechtsklick auf Projekt > Maven > Update Project (Alt+F5)
   * foundation bauen, dann core bauen
   * *Bemerkung: durch die Save Actions kann es passieren, dass Imports verloren gehen (wenn gespeichert wird, ohne dass Persistence- und Schema-Objekte generiert und sichtbar waren). Die müssen dann neu eingebunden werden!*

#### Ausführen

* Rechtsklick auf pom.xml > Run as > Maven build (Alt+Shift+X, M)
* Auf der Entwicklungsumgebung (devenv) greift flyway.
  + Bei Problemen mit flyway
    - lokal die jeweilige DB löschen, neu erstellen
    - Build target „flyway:repair“
    - **LOKAL!!!** In der Tabelle schema\_version den zugehörigen (meist letzten) Eintrag löschen

#### Build Parameter

* Beim ersten Build: Goals = „clean generate-sources install”
* Falls Unit-Tests fehlschlagen: in Run Configurations “Skip Tests”
* Pro Build: Profiles

1. local *oder leer* (okale Umgebung)
2. testenv (= PräProd)
3. prodenv

Die Build-Paramenter müssen für jedes pom.xml und für jeden Branch neu eingegeben werden. Trick, um das zu verallgemeinern: Nutzung vom relativen Pfad innerhalb Workspace zu den Metadaten – siehe Kapitel 7.1.

## Deploy

### Frontend Cleanup

liferay/tomcat/temp/\*

### Backend Cleanup

tomcat/work/Catalina/localhost/agn.\*

tomcat/webapps/project.\* *(Verzeichnisse und Dateien)*

### Frontend: Copy auf Liferay

Quellverzeichnis: Workspace/delivery/frontend

Zielverzeichnis: liferay/deploy

Pakete: \*.war

### Backend: Copy auf Tomcat

Quellverzeichnis: Workspace/delivery/frontend

Zielverzeichnis: tomcat/webapps/

Pakete: \*.war

## Run

### Tomcat

* tomcat\bin\startup.bat
* Anpassungen: s. General-Wiki.docx, Abschnitt Tomcat
* läuft lokal auf Port 8180 🡪 anzupassen in tomcat\conf\server.xml
  + <Connector port=“8180“ …/>

### Liferay

* liferay\tomcat\bin\startup.bat
* Anpassungen: s. General-Wiki.docx, Abschnitt Liferay
* läuft lokal auf Port 8080, d.h. Aufruf per <http://localhost:8080/>

# PräProduktion

## Build

### Jenkins-Build

trigger.manual.<project>.pipeline.build.preproduction > 

### Versionsnummern

Auf Server einloggen:

* sudo su <user>

bzw.

* sudo su <user> -

### Datenbank

Ggf. DB-Skripte ausführen.

* Siehe Release-Notes: parentpom > release\_information > Rx\_y\_z.txt
* Skripte befinden sich unter: <project>.persistence > src/main/resources > db
* Blatt “Datenbankstände” (an der Wand) aktualisieren

### Flyway Probleme beim Jenkins deployment

Wenn auf dem Jenkins ein Branch gebaut wird, der einen älteren Stand bezüglich der <project>.persistence besitzt als ein früherer Build, kommt es zu Problemen.

**Ursache**: das flyway:clean funktioniert für die aef\_db nicht sauber.

**Lösung**: Manuell auf Jenkins MySQL-DB

DROP DATABASE aef\_db;

CREATE DATABASE aef\_db

DEFAULT CHARACTER SET=utf8

DEFAULT COLLATE=utf8\_general\_ci;

Danach den Build-Job neu starten.

## Partielles Build (Hotfix)

Wenn nur eine oder wenige Applikationen zur Verfügung gestellt werden sollen, wird nur ein Teil gebaut.

1. Merge branch 🡪 trunk (oder umgekehrt)
2. Version pflegen im branch (s. 3.1.2 „Version“)
3. Jenkins-Build: <project>.pipeline.build.all auf testenv und branch
4. Im Falle eines Hotdeploy (falls das möglich ist): nur die betroffene war-Files ins zugehörige Verzeichnis kopieren

# Produktion

## Checkliste

Deployment Checkliste

## Build

* Jenkins-Dir: /data1/.jenkins/workspace/<build job>/delivery/
* Blatt “Datenbankstände” pflegen

### Datenbank

Ggf. DB-Skripte ausführen.

* Siehe Release-Notes: parentpom > release\_information > Rx\_y\_z.txt
* Skripte befinden sich unter: <project>.persistence > src/main/resources > db

# Merge

1. Zweig, in den gemergt werden soll, auschecken.
2. Lokaler Merge:

Rechtsklick auf den ausgecheckten Zweig > TortoiseSVN > Zusammenführen

* 1. Einen Revisionsbereich
  2. Zusammenführen aus URL: anderen Zweig/trunk wählen
  3. Revisionsbereich > Zeige Log: Revision(en) wählen

1. Commit des lokalen Mergs:

Rechtsklick > Commit

* 1. Kommentar: Letzte Meldungen > Tortoise schläge einen Text vor (ggf. überarbeiten)

## Neuen Branch auschecken

* + 1. Neues Verzeichnis anlegen
    2. .metadata in das neue Verzeichnis kopieren
    3. Projekte in der View „Project Explorer“ löschen (allerdings nicht auf der Disk)
    4. Neu auschecken

# Betrieb

Betrifft Produktion und PräProduktion.

## ActiveMQ neustarten

1. ActiveMQ stoppen (als **root**)
   1. sudo su
   2. amqstop
   3. ps aux | grep active
      1. Falls activemq noch läuft: kill -9 <id>
2. Tomcat stoppen (als **agn**)
   1. su agn
   2. tomstop
   3. ps aux | grep tom
      1. Falls tomcat noch läuft: kill -9 <id>
3. ActiveMQ starten (als **root**)
   1. ^d
   2. amqstart
   3. ps aux | grep active
4. Tomcat starten (als **agn**)
   1. su agn
   2. tomstart
   3. tomtail
      1. Warten auf Eintrag: “INFO: Server startup in 166476 ms“

# Skripte

## Code Count

1. Alles auschecken
2. MobaXterm starten (Alternative: Cygwin).
3. cd /drives/<Laufwerk>/<Pfad>
4. Skripte ausführen.

### Anzahl Java Klassen

$ find . -name \*.java | wc -l

2366

### Anzahl Java Codezeilen

$ find . -name \*.java -exec cat {} \; | wc -l

381725

### Anzahl XML Files

$ find . -name \*.xml | wc -l

1025

### Anzahl XML Zeilen

$ find . -name \*.xml -exec cat {} \; | wc -l

278824

Alternativ: Tool cloc (count lines of code)