Inhalt

[1. Eclipse 3](#_Toc451287967)

[1.1. Preferences setzen 3](#_Toc451287968)

[1.1.1. Alle in Eclipse einbinden 3](#_Toc451287969)

[1.1.2. Checkstyle einbinden: 3](#_Toc451287970)

[1.1.3. Code Styles angeben 3](#_Toc451287971)

[1.1.4. Neustart 3](#_Toc451287972)

[1.2. Klassen suchen 3](#_Toc451287973)

[1.3. svn 4](#_Toc451287974)

[1.3.1. Show Annotation 4](#_Toc451287975)

[1.3.2. Dateien vergleichen 4](#_Toc451287976)

[1.4. Java heap space-Error 4](#_Toc451287977)

[1.5. Compiler Fehler ausblenden 4](#_Toc451287978)

[1.6. Referenzen suchen 5](#_Toc451287979)

[1.7. Validierungen 5](#_Toc451287980)

[1.7.1. Allgemeine Validierungen 5](#_Toc451287981)

[1.7.2. JSP spezifisch 5](#_Toc451287982)

[1.8. Checkout as Maven project 5](#_Toc451287983)

[1.9. Save Actions 6](#_Toc451287984)

[1.10. Integrierter Tomcat 6](#_Toc451287985)

[1.10.1. Konfiguration 6](#_Toc451287986)

[2. Tomcat 7](#_Toc451287987)

[2.1. PermGen Space-Fehler 7](#_Toc451287988)

[2.2. Eclipse debugging 7](#_Toc451287989)

[3. Liferay 8](#_Toc451287990)

[3.1. Java-Pfad korrigieren 8](#_Toc451287991)

[3.2. Eclipse Debugging 8](#_Toc451287992)

[4. XML 9](#_Toc451287993)

[4.1. XML formatiert ausgeben 9](#_Toc451287994)

[4.2. XMLPad 9](#_Toc451287995)

[4.3. XMLStarlet 9](#_Toc451287996)

[4.4. Allgemein 10](#_Toc451287997)

[5. SoapUI Projekt anlegen 10](#_Toc451287998)

[6. Firefox Debugging 10](#_Toc451287999)

[7. ActiveMQ 10](#_Toc451288000)

[8. Maven 11](#_Toc451288001)

[8.1. Versionen 11](#_Toc451288002)

[9. HeidiSQL 12](#_Toc451288003)

[9.1. Auf Server tunneln 12](#_Toc451288004)

[9.2. Nach Debian 12](#_Toc451288005)

# Eclipse

## Preferences setzen

### Alle in Eclipse einbinden

1. File > Import > General > Preferences > eclipse\_preferences.epf auswählen
2. Import all

### Checkstyle einbinden:

1. Window > Preferences > Checkstyle > New
   1. External Configuration
   2. Name: “My Preferences”
   3. Location: Pfad zur checkstyle.xml auswählen
   4. Additional Properties
      1. Name: checkstyle.header.file
      2. Value: Pfad zur some\_copyright.txt
   5. Als Default setzen
2. In allen Projekte Checkstyle aktivieren
   1. Alle Projekte auswählen > Rechtsklick > Checkstyle > Activate

### Code Styles angeben

1. Window > Preferences > Java > Code Style
   1. Clean Up: cleanup.xml importieren
   2. Code Templates: codetemplates.xml importieren
      1. Comments > Types: @author auf eigene Mail-Adresse setzen
   3. Formatter: formatter.xml importieren
   4. Organize Imports: importorder.txt importieren

### Neustart

Nach dem Ändern der Einstellungen Eclipse neustarten.

## Klassen suchen

eclipse

1. Klassenansicht synchronisieren: Project Explorer > Link with Editor ()
2. Open Type (Ctrl+Shift+T bzw. ) > Klassenname eingeben

## svn

### Show Annotation

Perspective: SVN Repository Exploring

Rechtsklick auf eine Datei > Show Annotation

### Dateien vergleichen

Option 1: Rechtsklick auf Datei > Compare With > Latest from Repository

Option 2: Rechtsklick auf Projekt oder Datei > Team > Show History > Zwei Revisionen auswählen > Rechtsklick auf Auswahl > Compare

## Java heap space-Error

<https://wiki.eclipse.org/FAQ_How_do_I_increase_the_heap_size_available_to_Eclipse%3F>

Möglichkeit 1: Eclipse starten mit Parameter

eclipse -vmargs –Xmx1024M

Möglichkeit 2:

eclipse.ini anpassen (dort stehen auch die Standard-Parameter)

## Compiler Fehler ausblenden

Maven und eclipse bauen standardmäßig in das gleiche Verzeichnis und kommen dadurch durcheinander.

Möglichkeit 1: Unterschiedlich konfigurieren.

Möglichkeit 2: Projekt > Clean…

**Wichtig: Generierte Sourcen (z.B. von Hibernate) müssen als Sourcefolder im generierenden Projekt eingebunden werden! Das geht auch über Maven:**

<plugin>

    <groupId>org.codehaus.mojo</groupId>

    <artifactId>build-helper-maven-plugin</artifactId>

    <executions>

        <execution>

            <id>add-source</id>

            <phase>generate-sources</phase>

            <goals>

                <goal>add-source</goal>

            </goals>

            <configuration>

                <sources>

                    <source>${project.build.directory}/generated-sources/java/</source>

                </sources>

            </configuration>

        </execution>

    </executions>

</plugin>

## Referenzen suchen

Cursor auf Funktion > Rechtsklick > References > Workspace

oder: Strg+Shift+G

Zweite Methode:

Strg+G

## Validierungen

### Allgemeine Validierungen

* Window > Preferences > Validation

z.B. JPA Validator, JSP Content Validator, JSP Syntax Validator: Manual ein, Build aus

* Manuell starten: Projekt > Rechtsklick > Validate

### JSP spezifisch

Window > Preferences > Web > JSP Files > Validation

## Checkout as Maven project

1. Folgende Plugins müssen in Eclipse installiert sein:

* Subclipse (Standard-Plugin im Entwickler-Team)
* m2e (vorinstalliert in Eclipse J2EE)

2. Folgende Plugins in Eclipse nachinstallieren:

* m2e-subclipse
* buildhelper

Man findet diese Plugins nur im Marketplace, wenn man über die Maven-Preferences navigiert. Also:

Preferences->Maven->Discovery->Open Catalog...

und in dem folgenden Dialog die o.a. Plugins aktivieren und installieren.

3. Anschließend die Projekte aus Subversion im SVN-Repository Explorer markieren und über den Menüeintrag "Check out as Maven Project..." im Kontext-Menü auschecken.

4. Nachdem der Checkout und der anschließende Eclipse-Build vom Rechner abgearbeitet wurden, einmal im Navigator / Package Explorer alle Projekte markieren und mit Kontext-Menü->Maven->Update Project... die Eclipse-Projekte neu generieren lassen.

Nun sollten alle Projekte fehlerfrei von eclipse gebaut werden.

## Save Actions

Window > Preferences > Java > Editor > Save Actions

## Integrierter Tomcat

Window > Show View > Servers > Rechtsklick > New > Server

* Einstellungen vornehmen, „Next“ (kann ein wenig dauern).
* Projekte hinzufügen, „Finish“

Einstellungen aus context.xml und server.xml werden automatisch übernommen. Einstellungen aus \*.properties nicht.

### Konfiguration

Rechtsklick auf Server > Properties

Doppelklick auf Server

* Timeouts > Start hochstellen
* Server Locations: besser workspace metadata (sonst werden Einstellungen der Tomcat-Installation überschrieben)

# Tomcat

## PermGen Space-Fehler

Im Verzeichnis tomcat/bin eine Datei setenv.bat anlegen mit folgendem Inhalt (Standard ist 64m):

set CATALINA\_OPTS="-XX:MaxPermSize=128m"

## Eclipse debugging

* + - 1. **eclipse settings**

Run > Debug Configurations… > Remote Java Application

Name: tomcat local

Project: <Name vom Projekt>

Connection Type: Standard (Socket Attach)

Host: localhost

Port: 8000

* + - 1. **Start**

startup.bat ist nicht korrekt konfiguriert und übergibt den Parameter jpda nicht richtig. Daher muss die letzte Zeile wie folgt aussehen:

call "%EXECUTABLE%" jpda start %CMD\_LINE\_ARGS%

*(Ansonsten könnte man das Start-Skript so aufrufen:*

startup.bat jdpa start*)*

# Liferay

## Java-Pfad korrigieren

Standardmäßig wird Java aus %CATALINA\_HOME%/jre1.6.0\_20/win verwendet. Um das Standard-Java (meist 1.7) zu verwenden, muss in %CATALINA\_HOME%/bin/setenv.bat das Setzen von JAVA\_HOME / JRE\_HOME per REM ausgeklammert werden.

## Eclipse Debugging

* + - 1. **eclipse settings**

Run > Debug Configurations… > Remote Java Application

Name: liferay local

Project: <Name vom Projekt>

Connection Type: Standard (Socket Attach)

Host: localhost

Port: 8081

* + - 1. **setenv.bat**

set JPDA\_ADDRESS=8081

* + - 1. **Start**

startup.bat ist nicht korrekt konfiguriert und übergibt den Parameter jpda

nicht richtig. Daher muss die letzte Zeile wie folgt aussehen:

call "%EXECUTABLE%" jpda start %CMD\_LINE\_ARGS%

*(Ansonsten könnte man das Start-Skript so aufrufen:*

startup.bat jdpa start*)*

# XML

## XML formatiert ausgeben

Auf der Konsole:

ggf. sudo apt-get install xmlstarlet

xmlstarlet fo < filename

Mit Notepad++:

ggf. Plugin “XML Tools” installieren

Datei öffnen

Erweiterungen > XML Tools > Pretty print (XML only - with line breaks)

## XMLPad

1. Anzahl aller Nodes in aktueller Datei, die einen ABC-Barcode besitzen:

count(/StatusAuftraege/StatusAuftrag/Barcode[substring(text(),1,3)="ABC"])

1. Anzahl aller Nodes in aktueller Datei mit PositionsNummer 001, die einen ABC-Barcode besitzen:

count(/StatusAuftraege/StatusAuftrag[substring(Barcode/text(),1,3)="ABC"]/StatusPosition[@positionNummer="001"])

1. Alle Nodes in aktueller Datei, die einen ABC-Barcode besitzen:

/StatusAuftraege/StatusAuftrag[substring(Barcode/text(),1,3)="ABC"]

1. Alle Barcodes von Aufträgen, die den Statuscode 410 besitzen:

/StatusAuftraege/StatusAuftrag[StatusPosition/Statuswechsel/Statuscode/text()=410]/Barcode

## XMLStarlet

*Bemerkung: wenn auch die Responses für Namespace …142 berücksichtigt werden sollen, muss die Abfrage dementsprechend angepasst werden.*

1. Anzahl aller Nodes in einer Datei, die einen ABC-Barcode besitzen:

xmlstarlet sel -N x="http://www.arcor.de/status\_149" -t -v "count(/x:StatusAuftraege/x:StatusAuftrag/x:Barcode[substring(text(),1,3)='ABC'])" STATUS.xml

1. Anzahl aller Nodes in einer Datei mit PositionsNummer 001, die einen ABC-Barcode besitzen:

xmlstarlet sel -N x="http://www.arcor.de/status\_149" -t -v "count(/x:StatusAuftraege/x:StatusAuftrag[substring(x:Barcode/text(),1,3)='ABC']/x:StatusPosition[@positionNummer='001'])" STATUS.xml

1. Alle Barcodes von Aufträgen, die den Statuscode 410 besitzen:

xmlstarlet sel -N x="http://www.arcor.de/status\_149" -t -m "/x:StatusAuftraege/x:StatusAuftrag[x:StatusPosition/x:Statuswechsel/x:Statuscode/text()=410]/x:Barcode" -v "." -n STATUS\*.xml

## Allgemein

Anzahl aller Dateien, die den Namespace http://www.arcor.de/status\_142 besitzen:

grep http://www.arcor.de/status\_142 STATUS \*.xml -c|grep -v :0|wc –l

# SoapUI Projekt anlegen

* Project Name angeben
* Initial WSDL angeben
* Create Requests: ok!

# Firefox Debugging

Strg+Shift+I: Entwickler-Werkzeuge öffnen (Firebug) > Konsole

Alle Feldnamen anzeigen per JavaScript. In Konsole eingeben:

* Einfügen erlauben
* $.each($(':text'), function(){$(this).val($(this).prop('name'))});
* $.each($(':checkbox'), function(){$(this).addClass('imo\_force\_inline');$(this).after('<span>'+$(this).prop('name')+'</span>')});

# ActiveMQ

Starten: ACTIVEMQ\_HOME/bin/activemq start

Stoppen: ACTIVEMQ\_HOME/bin/activemq stop

Konfiguration: ACTIVEMQ\_HOME/conf/activemq.xml

Speicherort: <persistenceAdapter>

Max. Größe: <storeUsage>

Logs: dataDirectory

Jede nicht abgeholte Nachricht (Admin-Oberfläche > Queues > Number Of Pending Messages) wird persistiert. Dafür wird das gesamte zugehörige journalFile der kahaDB gespeichert. Es gilt als Standardeinstellung:

journalMaxFileLength="32mb"

# Maven

## Versionen

mvn versions:set -DnewVersion=a.b.c

Falls abgeleitete Projekte die gleiche Version haben sollen (= erben sollen), darf das entsprechende Versions-Tag nicht überschrieben werden (= nicht vorkommen).

# HeidiSQL

## Auf Server tunneln

|  |  |
| --- | --- |
| **Reiter „Einstellungen“** | |
| Verbindungstyp | MySQL (SSH tunnel |
| Hostname/IP | localhost (da der Tunnel lokal über plink geht) |
| Benutzername | Benutzername des MySQL-Users |
| Passwort | Passwort des MySQL-Users |
| Port | 3306 (Port der MySQL-DB auf dem Ziel) |
| **Reiter „SSH Tunnel“** | |
| plink.exe Pfad | Pfad zur lokalen plink.exe |
| SSH Host | IP des Servers, auf dem die MySQL-DB liegt |
| SSH Port | SSH Port des Servers, auf dem die MySQL-DB liegt (meist 22) |
| Lokaler Port | Irgendeiner, der frei ist (z.B. 3307) |

## Nach Debian

Beim Versuch sich von außen auf einen MySQL-Server zu verbinden begegnet man zuweilen folgender Fehlermeldung:

#2013 - Lost connection to MySQL server at 'reading initial communication packet', system error: 61

Nach der Aktivierung des Großhirns und einem Studium der Konfigurationsdatei ergibt sich die Lösung des Problems durch folgende Änderung in

/etc/mysql/my.cnf

Im Abschnitt [mysqld] darf die bind-address Einstellung nicht localhost bzw. 127.0.0.1 sein.

bind-address = 0.0.0.0

Mit 0.0.0.0 bindet sich der MySQL-Server an jede verfügbare IP des Systems. Um dies restriktiver zu halten, kann auch die IP des jeweiligen Interface verwendet werden.