Festlegungen für die Code Entwicklung

2016.06 2017.06
2017-08-21

by Kathleen Neumann, Thomas Scheffler, Jens Kupferschmidt

Table of contents

1 Vorbemerkungen

Von der Entwicklergruppe wird das Werkzeug Eclipse zur Arbeit am MyCoRe-Projekt empfohlen. Es enthält sowohl Funktionalitäten zur Integration von Subversion wie auch zur Qualitätssicherung des zu erstellenden JAVA-Codes.

2 Encoding

2.1 Allgemeines

Grundsätzlich geht MyCoRe davon aus, dass alle Dateien, die nicht sprachabhängig sind, mit **UTF-8** kodiert wurden. Dies gestattet eine gute Nutzung auch über die Grenzen des deutschsprachigen Raumes hinaus. Dies betrifft vor allem die Dateitypen:

- Java-Code - *.java
- XML-/XED-/XSD-/XSL-Dateien - *.xml

2.2 Konfiguration unter Eclipse

Die Einstellung erfolgt in Eclipse im jeweiligen Projekt unter:

1. Properties -> Info -> Text file encoding -> UTF-8

3 Java-Code Formatierung

3.1 Allgemeines

Um bei der Arbeit mit dem Subversion-System nur inhaltliche Änderungen zu erfassen und diese nicht mit Umformatierungen zu verwechseln, wird der gesamte Code einheitlich nach folgenden Regeln erstellt:

- Formatierung gemäß den Java-Konventionen
- Tabulatoren werden durch Leerzeichen ersetzt
- Einrückung mit vier Zeichen
- Zeilenumbruch und maximale Zeilenlänge 120
- Kommenare nicht umfomatieren

3.2 Konfiguration unter Eclipse

Die Einstellung erfolgt in Eclipse im jeweiligen Projekt unter:

1. Properties -> Java code style -> Formatter
2. dort Java Conventions als Vorlage wählen
3. Indentation -> Tab Policy -> Spaces only
4. Indentation -> Indentation size -> 4
5. Indentation -> Tab size -> 4
6. Line wrapping -> Maximum line width -> 120

7. Comments -> alle Markierungen entfernen
8. Die Vorlage wird dann als mycore gespeichert.

> **Download der Eclipse-Style-Definition**
>
> Eclipse-Nutzer können sich einfach die Code-Style-Definition herunterladen und importieren (unter: Properties -> Java code style -> Formatter).

4 XML-Code Formatierung

4.1 Allgemeines

Auch für die Gestaltung der XML-Dateien wurde eine einheitliche Formatierung festgelegt:

- Zeilenumbruch und maximale Zeilenlänge 120
- Für Einrückungen sind nur Leerzeichen zu verwenden.
- Pro Tab sind 2 Zeichen Einrückung vorgesehen.

4.2 Konfiguration unter Eclipse

Die Einstellung erfolgt in Eclipse im jeweiligen Projekt unter:

1. Window -> Preferences -> XML -> XML Files -> Editor
2. dort wählen
3. Line width -> 120
4. Indent using spaces
5. Indentation size -> 2
6. Format Comments abschalten

Weiterhin ist die Erkennung von *.xed-Dateien als XML im Editor einzuschalten

1. Window -> Preferences -> General -> Editors -> File Associations
2. dort wählen *.xed hinzufügen

> **Hinweis**
>
> Für die Bearbeitung der build.xml verwendet Eclipse einen eigenen Editor. Auch hier müssen die Einstellungen entsprechend geändert werden.

5 Compilieren und Code-Test

5.1 Allgemeines

Bevor ein Codeteil (Javaklasse, Stylesheet oder Konfigurationsdatei) committed wird, sollte diese **gründlich** getestet werden. Für Java-Klassen ist als erstes sicher zu stellen, dass sie den Compiler-Lauf erfolgreich bestehen. Um auch Konflikte mit anderen Java-Klassen von vorn herein auszuschließen, sollte vor jedem Commit einer Klasse des MyCoRe-Kerns der Aufruf

mvn [-o] clean install

erfolgen. So können Fehler in der Abhängigkeit schneller gefunden werden. Dabei steht -o für Offline. Dies kann genutzt werden, wenn man gerade im Zug oder an ähnlichen Orten entwickelt, wo man offline ist.

5.2 Nutzung der Entwicklungsumgebung Eclipse

Die Datei .project sollte entsprechend der unten angegebenen Darstellung angepasst werden. Anschließend sind sowohl für den MyCoRe-Kern wie auch für die Anwendung noch die Java-Ressourcen zu definieren.

1. project -> Properties -> Java Build Path
2. Tragen Sie alle Sources ein.
3. Tragen Sie alle Libraries ein. Bitte beachten Sie, dass für den Kern alle `*.jar` bzw. `*.zip` Dateien aus `mycore/lib` bzw. `mycore/components/*/sources` zu verwenden sind. Hinzu kommen noch ggf Systemweite Pakete (z. B. aus /usr/share/java). Für die Anwendung sollten nur Pakete aus application/build/lib, nicht aus dem MyCoRe-Baum, integriert werden.

<?xml version="1.0" encoding="UTF-8"?> <projectDescription> <name>mycore</name> <comment>...</comment> <projects>...</projects> <buildSpec> <buildCommand> <name>org.eclipse.jdt.core.javabuilder</name> <arguments /> </buildCommand> </buildSpec> <natures> <nature>org.eclipse.jdt.core.javanature</nature> </natures> </projectDescription> **Codebeispiel 3.1:** .project Datei

5.3 Test mit JUnit

Eclipse integriert bereits ein Produkt namens JUnit (http://www.junit.org/index.htm). Mit ihm erhalten Sie ein Werkzeug, welches das Testen der von Ihnen erzeugten Java-Klassen ermöglicht. Alle Test sind im Verzeichnis `mycore/test` abzulegen. Um einen neuen Test zu generieren, gehen Sie wie folgt vor:

1. Navigieren Sie auf die Klasse, für die der Test erzeugt werden soll
2. rechte Maustaste -> New -> Junit Test Case
3. Setzen Sie den Source Folder auf `mycore/test`
4. Sinnvoll ist das automatische anlegen der Methoden `setUp()` und`tearDown()`
5. Next
6. Markieren Sie die Methoden, für die ein Test erfolgen soll.

Den Test selbst starten Sie, indem Sie in der Testklasse über die rechte Maustaste 'Run As' --> 'JUnit Test' aufrufen.

Testklassen, die auf Datenbankanfragen angewiesen sind, sollten die Klasse `org.mycore.common.MCRHibTestCase` erweitern. Alle anderen erweitern `org.mycore.common.MCRTestCase`.

6 Kommentare im Code

6.1 Allgemeines

Alle Kommentare sind in Englisch abzufassen. Dabei ist auf allgemein verständliche Sprachkonstrukte zu achten. Der Kommentar soll die kodierten Vorgänge gut beschreiben und für andere nachvollziehbar machen.

6.2 Kommentare im Java-Code

Jede Java-Quelltext-Datei hat das nachfolgende Aussehen. Das gestattet ein einheitliches Auftreten des Projektes. Da für alle Dateien mittels JavaDoc automatisch eine Dokumentation generiert wird, ist es Pflicht, die erstellte Klasse mit einer allgemeinen Beschreibung zum Zweck und Einsatz der Klasse zu versehen. Weiterhin sind alle public oder protected Methoden mit einer Beschreibung zu versehen. Hierzu gehört auch die Dokumentation der übergebenen Parameter, der Rückgabewerte und ggf. der geworfenen Exceptions.

/* * This file is part of *** M y C o R e *** * See http://www.mycore.de/ for details. * * This program is free software; you can use it, redistribute it * and / or modify it under the terms of the GNU General Public License * (GPL) as published by the Free Software Foundation; either version 2 * of the License or (at your option) any later version. * * This program is distributed in the hope that it will be useful, but * WITHOUT ANY WARRANTY; without even the implied warranty of * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the * GNU General Public License for more details. * * You should have received a copy of the GNU General Public License * along with this program, in a file called gpl.txt or license.txt. * If not, write to the Free Software Foundation Inc., * 59 Temple Place - Suite 330, Boston, MA 02111-1307 USA */ package org.mycore.datamodel.metadata; import java.io.File; /** * This class implements all methode for ... * * @author Jens Kupferschmidt * @version $Revision: 1.49 $ $Date$ */ final public class MCR... { ...

Mindestkommentar im Java Quellcode:

/** * This method .. * * @param a The paremeter a is the first value. * @return a if it is a positive number, else return 0. */ public final int abs(a) { ... }

6.3 Kommentare in den Stylesheets

Auch alle XSLT-Dateien sollen kurze Kommentare enthalten. Unbedingt erforderlich sind auf jeden Fall die Zeilen für die Versionskontrolle. Angestrebt wird folgendes Aussehen eines Stylesheets.

<?xml version="1.0" encoding="UTF-8"?> <!-- == --> <!-- $Revision: 1.2 $ $Date$ --> <!-- == -->

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> ...
</xsl:stylesheet>

6.4 Logging

Alle Logging-Informationen werden, sofern nicht eine Umsetzung mittels der Internationalisierung I18N erfolgt, in Englisch notiert. Für MyCoRe ist das **log4j**-Paket des Apache-Projektes zu verwenden. Es gib 4 definierte Log-Level mit nachfolgenden Bestimmungen. Es ist davon auszugehen, dass eine normale Anwendung allgemein auf den Level INFO gesetzt ist.

- >ERROR – Gibt Informationen zu nicht behebbaren Fehlern, z.B. Exceptions, zurück.
- WARN – Gibt Informationen zu Fehlern zurück, welche die Weiterarbeit der Anwendung nicht ausschließen. In der Regel wird mit Standardwerten weitergearbeitet.
- INFO – Gibt allgemeine Informationen für den normalen Anwender bzw. die Log-Datei aus. Diese Nachrichten haben nur informativen Charakter.
- DEBUG – Gibt zusätzliche Informationen, die gezielt durch Einschalten dieses Levels abgerufen werden, aus.