Developer Guide

RELEASE 2.0

22. März 2007

Frank Lützenkirchen (Essen/Duisburg)

Jens Kupferschmidt (Leipzig)

Andreas Trappe, Thomas Scheffler (Jena)

Kathleen Krebs (Hamburg)

Vorwort

Dieses Dokument ist für die aktiven und interessierten Entwickler gedacht. Es dokumentiert alle im **Architecture Board** getroffenen Vereinbarungen. Die hier beschriebenen Festlegungen sind für alle am Entwicklungsprozess Beteiligten verbindlich. Sie sollen vor allem die einheitliche Gestaltung des Projektes garantieren.

Inhaltsverzeichnis

1 Gemeinsame Entwicklungsplattformen

1.1 SourceForge

Für die Präsentation und den Download von Releases und Distributionen wurde ein Projekt auf den Server von SourceForge[1] eingerichtet. Hier können auch auftretende Bugs dokumentiert werden.

1.2 CVS-Zugang

Sowohl die aktuellen Quellen des MyCoRe-Kerns wie auch einiger beispielhafter Anwendungen werden auf einem CVS-Server in Essen verwaltet. Alle Quellen stehen für einen anonymen Zugang mittels CVS-Client zur Verfügung[2]. Darüber hinaus ist der Zugang zum CVS-Server des MyCoRe-Projekts für interessierte Entwickler mit Schreibberechtigung möglich. Dieser erfolgt nach Freischaltung eines persönlichen Accounts über `ssh`.

Nach dem Freischalten sind folgende Umgebungsvariablen zu setzen:

```
CVS_RSH=ssh
CVSROOT=:ext:mcr_username@server.mycore.de:/cvs
export CVS_RSH CVSROOT
```

Es empfiehlt sich natürlich zuerst die MyCoRe-Quellen herunterzuladen:

```
cvs -d:ext:mcr_username@server.mycore.de:/cvs checkout mycore
```

Danach können sie mit

```
cvs -d:ext:mcr_username@server.mycore.de:/cvs commit -m "Kommentar"
file
```

Daten einstellen.

Voraussetzung ist ein CVS-Login, welches Sie bei Frank Lützenkirchen[3] beantragen können. Soll in ein bestehendes Projekt eine neue Datei integriert werden, so legt man sie zunächst lokal im vorgesehenen (und bereits ausgecheckten) Verzeichnis an. Dann merkt man sie mittels `cvs add <filename>` vor. Um sie dann global zu registrieren, erfolgt ein (verkürzt): `cvs commit <filename>`. Neue Unterverzeichnisse werden auf die gleiche Weise angelegt. Weitere und sehr ausführliche Informationen sind im Internet[4] zu finden.

[1] http://sourceforge.net/projects/mycore
[2] siehe MyCoRe-UserGuide
[3] luetzenkirchen@bibl.uni-essen.de
[4] http://www.cvshome.org/docs/
http://cvsbook.red-bean.com/translations/german/
http://panama.informatik.uni-freiburg.de/~oberdiek/documents/OpenSourceDevWithCVS.pdf
http://www.selflinux.org/selflinux/pdf/cvs_buch_kapitel_9.pdf

2 Festlegungen für die Code Entwicklung

2.1 Vorbemerkungen

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

2.2 Encoding

2.2.1 Allgemeines

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

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

2.2.2 Konfiguration unter Eclipse

Die Einstellung erfolgt in Eclipse im jeweiligen Projekt unter:

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

2.3 Code Formatierung

2.3.1 Allgemeines

Um bei der Arbeit mit dem CVS-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 160

2.3.2 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 → 160
7. Die Vorlage wird dann als mycore gespeichert.

2.4 Compilieren und Code-Test

2.4.1 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

ant clean jar

erfolgen. So können Fehler in der Abhängigkeit schneller gefunden werden.

2.4.2 Nutzung der Entwicklungsumgebung Eclipse

Wie bereits oben erwähnt, leistet die Entwicklungsumgebung Eclipse nicht nur hilfreiche Dienste bei der Formatierung des Java-Codes. Mit Ihr kann auch die Syntaxprüfung der Java-Klasse wie auch ihre Einbettung in das Gesamtprojekt leicht überwacht werden. Dazu sind einige Einstellungen erforderlich. Diese sollten, wie auch alle anderen Eclipse-Einstellungen, sowohl für das MyCoRe-Kernprojekt wie auch für die Anwendungen erfolgen.

Zuerst sollte die Datei .project 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/modules/... zu verwenden sind. Hinzu kommen noch ggf Systemweite Pakete[5]. Für die Anwendung sollten nur Pakete aus application/build/lib, nicht aus dem MyCoRe-Baum, integriert werden.

[5] z. B. aus /usr/share/java

.project Datei:

```
<?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>
```

2.4.3 Test mit JUnit

Eclipse integriert bereits ein Produkt namens JUnit[6]. 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 folgend 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.

2.5 Kommentare im Code

2.5.1 Allgemeines

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

[6] http://www.junit.org/index.htm

2.5.2 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.

Mindestkommentar im Java Quellcode:

```
/*
 * $RCSfile: MCR....java,v $
 * $Revision: 1.49 $ $Date: 2006/05/17 11:49:32 $
 *
 * 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: 2006/05/17 11:49:32 $
 */
final public class MCR...  {
    ...
```

```
/**
 * 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) {
    }
}
```

2.5.3 Kommentare in den Stylesheets

Auch alle XSLT-Dateien sollten 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: 2006/10/12 11:52:14 $ -->
<!-- ============================================= -->
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
>
...
</xsl:stylesheet>
```

2.5.4 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 sollen, aus.

3 Namenskonventionen

3.1 Benennung der Properties

MyCoRe-Properties beginnen immer mit MCR, dann ein Punkt, dann der Komponentenname (z.B. ACL, IFS ...). Alle weiteren Bezeichner sind mit Punkt voneinander getrennt und beginnen mit einem Grossbuchstaben.

Beispiel:
MCR.IFS.FileMetadataStore.Class

Properties von Drittanbietern (z.B. Log4j, Hibernate) behalten ihre Konventionen bei.

Komponente	**Property-Kürzel**	**Kommentar**
Internal File System	IFS	
...	ACL	
...	...	
...	...	

[ToDo: Jens] Liste vervollständigen

3.2 Stylesheets

Stylesheets in MyCoRe und DocPortal sind einheitlich zu benennen. Sie sollten (wenn möglich) komplett klein geschrieben werden. Die Verwendung eines Präfix wie mcr, MyCoRe oder mycore entfällt, auch werden keine Unterstriche genutzt.

Beispiele:
results.xsl
user.xsl
editor.xsl

Stylesheets für das Layout der Webseiten ...
[ToDo: Thomas] Namenskonvention bzgl. RootTag

4 DocPortal vs. MyCoRe

Welche Komponenten, Dateien etc. kommen in den Kern und welche in die Beispielanwendung MyCoRe?

4.1 MyCoRe

Die vollständige Funktionalität von MyCoRe ist im Kern. Damit ist MyCoRe an sich keine lauffähige Anwendung, sondern nur ein Rahmenwerk für solche. Alles was der Anwendungsentwickler an Funktionen in seiner eigenen Anwendung verwenden kann, und zu 99% nicht anfassen bzw. ändern muss, ist Teil des Kerns.

4.2 DocPortal

DocPortal bildet als mitgelieferte Beispielanwendung eine Vorlage für eigene Anwendungen. Daher sind in DocPortal alle anwendungsspezifischen Dateien abzulegen und solche, die die Anwendung(datei)struktur vorgeben bzw. direkt damit zusammenhängen.

DocPortal soll klar strukturiert werden, so dass mit kleinen HowTo's Änderungen z.B. am Layout oder am Datenmodell zu machen sind. Dateien die dementsprechend thematisch zusammengehören sollten wenn möglich auch zusammen liegen. Beispiel ist das Modul für das Datenmodell (docportal/modules/datamodel)

5 Dokumentation

5.1 Allgemeines

Dokumentationen zum MyCoRe-Projekt sind als Open Office Dokument zu erstellen und in PDF-Form zu speichern. Die Sprache der Dokumente ist wahlweise Deutsch oder Englisch.

6 Anhang

6.1 Abbildungsverzeichnis

6.2 Tabellenverzeichnis