

MyCoRe Developer Guide 2.0

Inhaltsverzeichnis

1 Gemeinsame Entwicklungsplattformen.....	3
1.1 SourceForge.....	3
1.2 Subversion-Zugang.....	3
2 Festlegungen für die Code Entwicklung.....	4
2.1 Vorbemerkungen.....	4
2.2 Encoding.....	4
2.3 Code Formatierung.....	4
2.4 Compilieren und Code-Test.....	5
2.5 Kommentare im Code.....	7
3 Namenskonventionen.....	9
3.1 Benennung der Properties.....	9
3.2 Benennung von I18N Übersetzungen.....	10
3.3 Stylesheets.....	11
4 Struktur einer Anwendung.....	11
4.1 MyCoRe.....	11
4.2 DocPortal.....	13
4.3 Module.....	13
5 Dokumentation.....	14
5.1 Allgemeines.....	14
5.2 Dokumente.....	14
6 Tests.....	14
6.1 Allgemeines.....	14
6.2 Stufe 0 - Commit während einer Komponentenentwicklung.....	15
6.3 Stufe 1 – Bugfix.....	15
6.4 Stufe 2 - Komponente ist fertig entwickelt.....	15
6.5 Stufe 3 - Release oder Snapshot.....	16

6.6 Test der Distribution..... 21

1. Gemeinsame Entwicklungsplattformen

1.1. SourceForge

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

<http://sourceforge.net/projects/mycore>

1.2. Subversion-Zugang

Der Quellcode der MyCoRe-Kerns und einiger Anwendungen, z. B. der Beispielanwendung DocPortal, wird auf einem Subversion-Server an der Universität Duisburg-Essen verwaltet.

Aktuell gib es zwei Zugangsmöglichkeiten: über HTTP im lesenden Zugriff und über SSH mit Schreibrechten (sofern der Entwickler Zugang zum System hat [der Server wird von Frank Lützenkirchen verwaltet]).

<http://www.mycore.de/svn/>

<svn+ssh://server.mycore.de/svn/>

Der Code ist innerhalb eines Projektes in die Zweige branches – für Releases, tags – für Snapshots und trunk – für den aktuellen HEAD-Zweig unterteilt.

Hinweis für Kommandozeilen-Nutzer

Dateien in der SVN-Arbeitskopie nie direkt mit Betriebssystemmitteln kopieren, löschen und verschieben. Stattdessen auf jeden Fall auf die `svn` Kommandos `cp`, `rm`, `mv` zurück.

Hinweis für Eclipse-Nutzer

Ab sofort sollten Dateien nicht mehr mit Cut & Paste sondern vielmehr über die Team-Funktionen kopiert werden, damit die Histories nicht verloren gehen!

MIME-Typ für alle neuen Dateien

Alle neuen Dateien, die bei MyCoRe mit Subversion verwaltet werden müssen das SVN Property `svn:mime-type` ausgefüllt haben. Bei Mime-Typ `text/*` muss zusätzlich noch das Property `svn:eol-style` beschrieben sein. Für die häufigsten Dateiarnten gibt es eine Konfiguration zum [Herunterladen](#) (`./filecollection/svn-config.txt`), die ans Ende der Datei `config` im Verzeichnis `.subversion` des Nutzerverzeichnisses angefügt wird. Der entsprechende Bereich trägt den Namen `[auto-props]`.

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 Subversion 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 wurden. 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 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 160
- Kommentare nicht umformatieren

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. 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](#) (../filecollection/mycore-javastyle.xml) herunterladen und importieren.

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 muss das Plugin für die Integration von Subversion installiert werden. Da inzwischen Subversive für die Integration in Eclipse vorgesehen ist, sollte dies dem Projekt Subclipse vorgezogen werden. Die Installation ist auf der Homepage beschrieben.

[Homepage von Subversive](http://www.eclipse.org/subversive/) (<http://www.eclipse.org/subversive/>)

Dann 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/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.

.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 (<http://www.junit.org/index.htm>) (<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 Datenbankabfragen angewiesen sind, sollten die Klasse `org.mycore.common.MCRHibTestCase` erweitern. Alle anderen erweitern `org.mycore.common.MCRTestCase`.

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 kodierten Vorgänge gut beschreiben und für andere nachvollziehbar machen.

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.

```
/*
 * $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;
<br/>
import java.io.File;
<br/>
/**
 * This class implements all methode for ...
 *
 * @author Jens Kupferschmidt
 * @version $Revision: 1.49 $ $Date: 2006/05/17 11:49:32 $
 */
final public class MCR... {
    ...

```

Mindestkommentar im Java Quellcode

```

/**
 * This method ..
 * <code>*
 * @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) {
    &emsp;&emsp;&emsp;
}

```

2.5.3. 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: 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 gibt 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.

3. Namenskonventionen

3.1. Benennung der Properties

MyCoRe-Properties beginnen immer mit MCR, darauf folgt ein Punkt und dann der Komponentename (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
allgemeine Konfiguration	MCR.ConfigurationMCR.XMLPa	
ACL	MCR.Access	

Classification Browser	MCR.ClassificationBrowser	
Classification Editor	MCR.ClassificationEditor	
Commandline Tool	MCR.CLI	
Editor	MCR.EditorFrameworkMCR.Edit	
Event Handler / Searcher	MCR.EventHandlerMCR.Searcher	
File Upload	MCR.FileUpload	
Goggle Indexer	MCR.GoogleSitemap	
IFS	MCR.IFS	
Index Browser	MCR.IndexBrowser	
Layout Service	MCR.Request	
Mail System	MCR.Mail	
Metadaten-Modell	MCR.Metadata	
OAI	MCR.OAI	
Simple Workflow	MCR.SWF	
Stores und Manager	MCR.Persistence	
Textextrakter	MCR.TextFilterPlugin	
Upload Applet	MCR.UploadApplet	
URI Resolver	MCR.URIResolver	
URN	MCR.URN	
User Administration	MCR.Useradmin	
User Kernsystem	MCR.Users	
WebService	MCR.WebService	
Zip Tool	MCR.Zip	
Z3950	MCR.z3950	

Tabelle 3.1: Liste der Property-Präfixe

3.2. Benennung von I18N Übersetzungen

Alle Übersetzungsvariablen erhalten ein einheitliches Namensschema. Alle Übersetzungen von Texten des Kerns und häufig benutzte Texte erhalten den Anfang common., Variablen der Module den Modulnamen und Teile der Anwendung den

Anwendungsnamen.

Beispiel:
common.cancel = Abbrechen
swf.object.edit = Editieren Objekt
docportal.title.label = Titel

3.3. 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. Struktur einer Anwendung

Eine MyCoRe-Anwendung gliedert sich in den Kern- und Modulbereich von MyCoRe sowie den Anwendungs- und Modulbereich der Applikation, als Beispiel DocPortal. Die gedachte Aufteilung wird im folgenden näher beschrieben.

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.

4.1.1. Definitionen

- **Allgemeiner Bestandteile** sind die Teile des MyCoRe-Kerns, welche von vielen/allen Komponenten und Modulen verwendet werden und die elementaren Grundfunktionalitäten von MyCoRe bereitstellen. U. a. beinhaltet dies die APIs zur Speicherung der Daten, Klassifikationen, der ACLs und Nutzer, Store-Zugriffe und Klassen, die allgemeiner Natur sind und als API-Funktionalitäten bereitgestellt

werden (MCRServlet, XML-Parser usw.).

- **Komponenten** sind Teile des MyCoRe-Kerns welche als Funktionalität in sich abgrenzbar sind. Sie bauen auf den **Allgemeinen Bestandteilen** auf und enthalten alle zur Funktionalität gehörenden Klassen und Dateien. Alle Komponenten haben eine fest vorgegebene Struktur (siehe unten) und können über ein Property ausgeschaltet werden (wie bisher bei den Modulen des Kerns). Die in den Komponenten enthaltenen Vorlagen (Templates) können durch anwendungsseitige Dateien überschrieben werden. Komponenten werden vom Entwickler-Team bereitgestellt und sind Bestandteil von mycore.jar. Komponenten werden über einen automatischen Prozess in die Anwendung integriert, wenn sie nicht explizit deaktiviert wurden.
- **Module** sind austauschbare/abschaltbare Teile der Nutzeranwendung, welche alle wichtigen Dinge zur Gestaltung konkreter Anwendungen enthalten, die nicht als generisch anzusehen sind (Datenmodelle, statische Seiten, Layouts usw.). Module werden von den Anwendern entwickelt und bauen auf dem mycore.jar auf. Module können auch Komponenten überschreiben bzw. ergänzen. Die Dateibaumstruktur vom Modulen und Komponenten soll analog zueinander sein.

4.1.2. Struktur des mycore-Verzeichnisses

Der mycore-Verzeichnisbaum ist ein Projektbereich, aus welchem die für die Anwendung relevante Datei mycore.jar erzeugt wird. Alle Teile unterstützen nur die Sprachen Englisch und Deutsch. mycore hat nachfolgende Struktur:

```

mycore
|
| \ build (mit ant generierte Daten)
| \ \ classes (Java-Klassen)
| \ \ javadocs (generierte JavaDocs)
| \ \ lib (Download-Bereich der externen Jars)
| \ components (alle Komponenten gemäß Definition von oben
| \ (ex. Module) und inkl. Migration)
| \ config (alle generischen Konfigurationen)
| \ resources (werden nicht übersetzt, sondern so wie sie sind nach
| \ build/classes kopiert - z.B. Hibernate-Mappings)
| \ schema (generische DTD- und XSD-Schema-Dateien)
| \ sources (Common parts - Sources)
| \ tests (JUnit Test Sources)
| \ xsl (alle generischen Stylesheets)
| \ Dateien build.xml, changelog.txt, license.txt

```

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 möglich sind. Dateien die dementsprechend thematisch zusammengehören, sollen wenn möglich auch zusammen liegen. Beispiel ist das Modul für das Datenmodell (`docportal/modules/docportal`)

4.3. Module

Module in MyCoRe können in mehreren Arten auftreten:

1. **Einfache Module** sind klar gekapselte funktionelle Einheiten, deren Bestandteile bis auf anzupassene Konfigurationen im Kern bereits vorhanden sind. Für die Konfiguration gibt es bereits Standardvorgaben. Die Module können über den Konfigurations- bzw. build-Prozess der Anwendung einfach integriert werden. Einfache Module werden im Kern automatisch mit compiliert.
2. **Nicht installierte Module** haben den selben Umfang wie einfache Module, sie müssen aber auf Grund von Lizenzbestimmungen erst bei der Erstellung der konkreten Anwendung im Kern durch Nutzung von ANT-Targets mit aktiviert werden. Nicht installierte Module werden im Kern erst automatisch mit compiliert, wenn sie aktiviert wurden.
3. **Anwendungsmodule** bezeichnen alle Module, die ausschließlich in einer speziellen Anwendung benutzt werden. Dazu gehört i.d.R. ein Modul, welches das konkrete Datenmodell von der Basisanwendung abgrenzt. Weiterhin können auch andere, nur in der Anwendung genutzte spezielle Funktionalitäten in einem Modul platziert werden.

Die nachfolgenden Festlegungen sind für jeden Modul umzusetzen:

1. Ein Module beinhaltet eine abgeschlossene Funktionseinheit, die auf dem Kern basiert und die bei Bedarf abgeschaltet bzw. nicht genutzt werden kann.
2. Die Auswahl der Module und deren Reihenfolge erfolgt über das Property **MCR.Modules.MyCoRe** für den Kern und **MCR.Modules.Application** für die Anwendung.
3. Alle Module stehen sowohl im Kern als auch in der Anwendung im Verzeichnis *modules*. Jedes Modul befindet sich dort in einem Verzeichnis mit dem entsprechenden Modulnamen.
4. Unterhalb dieses Ordners sind folgende Verzeichnisnamen festgelegt:

config	Konfigurationsdaten
lib	zusätzliche Bibliotheken

sources	Java-Quellcode
webpages	statische Web-Seiten
xsl	XSLT-Stylesheets

5. System-Properties sind immer einzutragen unter *config* in die Datei `mycore.MODULNAME.properties`. Alle Properties haben den Anfang `MCR.MODULNAME...=`. I18N-Properties beginnen immer mit `MODULNAME....`
6. Eine erforderliche `web.xml` steht zudem immer unter *config*.
7. Alle Funktionalitäten sind im Programmier Guide zu dokumentieren.
8. Das Modul ist möglichst in die Beispielanwendung DocPortal zu integrieren.
9. Eine `*.txt`-Datei sollte die notwendigen nicht automatischen Konfigurationsschritte zur Integration des Moduls in eine Anwendung beschreiben.

5. Dokumentation

5.1. Allgemeines

Die Dokumentation von MyCoRe und die Unterstützung der Nutzer erfolgt über mehrere Medien.

1. **bisher:** Dokumente im OpenOffice- und PDF-Format als Bestandteil der Releases.
neu: Die MyCoRe-Dokumentation wird zur Zeit für die kommende Version MyCoRe 2.0 überarbeitet und mit Apache Forrest erstellt
<http://www.mycore.de/documentation/documentation/index.html>
(`../documentation/index.html`)
2. Die MyCoRe-Web-Seite <http://www.mycore.de/documentation> (`../index.html`) als Web-Auftritt - jetzt auch **neu** über Apache Forrest
3. Die Know-How- und Dokumentationsseiten der MyCoRe-Wiki-Installation <http://cmswiki.rrz.uni-hamburg.de/hummel/MyCoRe/Dokumentation> als Plattform für ein Wissensforum zum Erfahrungsaustausch der Anwender.
4. Die Mailing-Listen für Benutzer und Entwickler zur direkten Kommunikation.

5.2. Dokumente

Dokumentationen zum MyCoRe-Projekt sind zukünftig als XML-Dateien mit Apache Forrest zu erstellen. Die Sprache der Dokumente ist wahlweise Deutsch oder Englisch.

6. Tests

6.1. Allgemeines

Zur Gewährleistung einer Mindestqualität sind mit dem aktuellen Code-Stand je nach

Entwicklungsstufe die nachfolgenden Tests durchzuführen. Voraussetzung für die Tests ist die Integration des JAI-Moduls in die Testumgebung, um auch Fehler im Bildbetrachter (ImageViewer) zu erkennen. Es wird vorausgesetzt, dass im Testsystem der vollständige aktuelle Codestand vorliegt. Nach jedem Arbeitsschritt ist zu prüfen, ob dieser erfolgreich und vollständig abgearbeitet wurde. Zur Kontrolle sollte der Log-Level auf DEBUG geschaltet werden und jede Kommandoausgabe in eine Datei geschrieben werden.

6.2. Stufe 0 - Commit während einer Komponentenentwicklung

- Neues compilieren der MyCoRe-Kern Quellen mit vorherigem Löschen der alten Daten durch 'ant clean jar'.
- Neues Compilieren der DocPortal Quellen durch 'ant jar'.

6.3. Stufe 1 – Bugfix

- Test der Stufe 0 durchführen
- Gefixte Komponenten und alle ggf. mit betroffenen Teile individuell testen.

6.4. Stufe 2 - Komponente ist fertig entwickelt

- Test der Stufe 0 durchführen
- Vollständige Neuinstallation der DocPortal-Anwendung bis zur Betriebsbereitschaft mit vorherigem Löschen der alten Daten durch 'ant clean clean.data'. Dazu sind die folgenden Schritte auszuführen:
 - Anlegen der Verzeichnisse mit 'ant create.directories'.
 - Erzeugen der XML-Schemas mit 'ant create.schema'.
 - Neues compilieren der DocPortal Quellen durch 'ant jar'.
 - Erzeugen der CLI-Scripts mit 'ant create.scripts'
 - Starten der HSQLDB mit 'build/bin/hsqldbstart.sh' bzw. 'build\bin\hsqldbstart'.
 - Test der Verbindung zur HSQLDB mit 'build/bin/hsqldbadmin.sh' bzw. 'build\bin\hsqldbadmin' (Server - localhost:8298).
 - Start des CLI mit 'build/bin/mycore.sh' bzw. 'build\bin\mycore'. Prüfen mit 'help' ob alle CLI-Teile geladen wurden (z. B. Image-Kommandos).
 - Laden der Standard-User mit 'ant create.users' (testet 'load permissions data from file ...', 'init superuser', 'change to user ...', 'create group data from file ...', 'create user data from file ...' und 'check user data consistency').
 - Start des CLI mit 'build/bin/mycore.sh' bzw. 'build\bin\mycore'. Prüfen mit 'list all users' und 'list all groups' ob alle User-System-Teile vollständig geladen wurden.

- Laden der Klassifikationen mit 'ant create.class' (testet 'update all classifications from directory ...').
 - Test der korrekten Installation der Klassifikationen mittels *select*-Statement über das hsqladmin-Tool.
 - Erzeugen des Applet-Keys mit 'ant create.genkeys'.
 - Erzeugen der Web-Applikation mit 'ant webapps'.
 - Starten der Anwendung mit 'build/bin/jettystart.sh' bzw. 'build\bin\jettystart'.
 - Aufruf von statischen Seiten, einer Suchmaske, einer Klassifikationsauswahl und dem Benutzerwechsel (wechseln zu author1A).
- Laden und Testen der Standardbeispieldaten in die Anwendung. Diese stehen in der gesonderten MyCoRe-Komponente Content. Dazu sind die folgenden Schritte auszuführen:
 - Stoppen der Anwendung mit 'build/bin/jettystop.sh' bzw. 'build\bin\jettystop'.
 - Laden der Institutionen mit 'ant load.institution' in content/defaultsample (testet 'update object from file ...').
 - Laden der Autoren mit 'ant load.author' in content/defaultsample (testet 'update object from file ...').
 - Laden der Dokumente mit 'ant load.document' in content/defaultsample (testet 'update object from file ...').
 - Laden der Objekte mit 'ant load.derivate' in content/defaultsample (testet 'update derivate from file ...').
 - Starten der Anwendung mit 'build/bin/jettystart.sh' bzw. 'build\bin\jettystart'.
 - Test der Anwendung im Browser: Aufruf der Suche, Navigation in der Trefferliste, Anzeige von Einzeltreffern und Detaillisten. Suche nach Volltexten und Anzeige von Bildern mit dem ImageViewer. Navigation in Klassifikationen.
 - Stoppen der Anwendung mit 'build/bin/jettystop.sh' bzw. 'build\bin\jettystop'.
 - Test der Datensicherung mit 'build/bin/Save.sh' bzw 'build\bin\Save'. Prüfen der gespeicherten Daten im Verzeichnis *save* inklusive der korrekten ACLs (testet 'export object ... to directory ... with ...', 'export derivate ... to directory ... with ...', 'export all classifications to ... with ...', 'export all groups to file ...', 'export all users to file ...' und 'export all permissions to file ...').
 - Intensiver Test der neuen Komponenten.
 - Dokumentation der Komponente auf Vollständigkeit prüfen

6.5. Stufe 3 - Release oder Snapshot

Die Tests sind für Unix/Linux und Windows durchzuführen!

- Test der Stufe 1 durchführen
- Test des Commandline Interfaces
- Test der Web-Anwendung
- Test des WCMS
- Test sonstiger Funktionen

6.5.1. Test des Commandline Interfaces

- Tests des CLI (Nutzerverwaltung)
 - Start des CLI mit 'build/bin/mycore.sh' bzw. 'build\bin\mycore'.
 - Login mit 'login administrator'.
 - Sichern des Users author1A mit 'export user author1A to file author1A.xml'.
 - Sichern des Users author1B mit 'export user author1B to file author1B.xml'.
 - Sichern der Gruppe authorgroup1 mit 'export group authorgroup1 to file authorgroup1.xml'.
 - Löschen einer Gruppe mit 'delete group authorgroup1'. Das Kommando darf NICHT funktionieren!
 - Löschen eines Nutzers mit 'delete user author1A'.
 - Löschen eines Nutzers mit 'delete user author1B'.
 - Löschen einer Gruppe mit 'delete group authorgroup1'. Das Kommando muss jetzt funktionieren!
 - Import der Gruppe authorgroup1 mit 'import group data from file authorgroup1.xml'.
 - Import des Nutzers author1A mit 'import user data from file author1A.xml'.
 - Import des Nutzers author1B mit 'import user data from file author1B.xml'.
 - Test des Datenbestandes mit 'list all users' und 'list all groups'.
 - Beenden des CLI mit 'quit'.
- Tests des CLI (Klassifikationsverwaltung)
 - Start des CLI mit 'build/bin/mycore.sh' bzw. 'build\bin\mycore'.
 - Sichern einer Klassifikation (ohne aktive Verweise) mit 'export classification DocPortal_class_00000009 to . with save'.
 - Löschen einer referenzierten Klassifikation mit 'delete classification DocPortal_class_00000006'. Das Kommando darf NICHT funktionieren!
 - Löschen einer nicht referenzierten Klassifikation mit 'delete classification DocPortal_class_00000009'. Das Kommando muss funktionieren!
 - Laden der gespeicherten Klassifikation mit 'load classification from file

- DocPortal_class_00000009.xml'.
- Update der gespeicherten Klassifikation mit 'update classification from file DocPortal_class_00000009.xml'.
 - Beenden des CLI mit 'quit'.
- Tests des CLI (Rechteverwaltung)
 - Start des CLI mit 'build/bin/mycore.sh' bzw. 'build\bin\mycore'.
 - Auflisten aller Permissions mit 'list all permissions'.
 - Export aller Permissions mit 'export all permissions to file permission.xml' und prüfen der Ausgabe.
 - [ToDo - 2007-08-29 - Jena] Test der 'update permission'-Kommandos beschreiben
 - Beenden des CLI mit 'quit'.
 - Tests des CLI (Objekte und Derivate)
 - Start des CLI mit 'build/bin/mycore.sh' bzw. 'build\bin\mycore'.
 - Anzeige der letzten verwendeten MCRObjektID mit 'get last ID for base DocPortal_document'. Es wird 'DocPortal_document_07910403' ausgegeben.
 - Anzeige der nächsten MCRObjektID mit 'get next ID for base DocPortal_document'. Es wird 'DocPortal_document_07910404' ausgegeben.
 - Reparieren der Indizes aller Dokumente mit 'repair metadata search of type document'.
 - Reparieren des Dokuments DocPortal_document_07910403 mit 'repair metadata search of ID DocPortal_document_07910403'.
 - Reparieren des Derivate DocPortal_derivate_00410903 mit 'repair derivate search of ID DocPortal_derivate_00410903'
 - Löschen des Derivates DocPortal_derivate_00410902 mit 'delete derivate DocPortal_derivate_00410902'.
 - Löschen des Objektes DocPortal_document_00410902 mit 'delete object DocPortal_document_00410902'.
 - Löschen des Objektes DocPortal_document_00410901 mit 'delete object DocPortal_document_00410901' (ohne vorherigem Löschen des Derivates).
 - Prüfen der relevanten HSQLDB-Tabellen und des Lucene-Eintrages.
 - Beenden des CLI mit 'quit'.

6.5.2. Test der Web-Anwendung

- Starten der Anwendung mit 'build/bin/jettystart.sh' bzw. 'build\bin\jettystart'.

- Tests der Web-Anwendung (statische Seiten)
 - Es sind alle Seiten der sitemap.xml Anzeige durchzutesten.
- Tests der Web-Anwendung (Suche/Präsentation/Navigation)
 - Suche nach Institutionen
 - Suche nach Personen, weiter Suchen nach verlinkten Dokumenten
 - Suche nach Dokumenten ohne Parameter und Navigation in der Trefferliste. Navigation aus der Anzeige heraus zur Trefferliste und zur Detailansicht. Anzeige von Dokumenten mit Bildern (bei integriertem ImageViewer).
 - Suche nach Dokumenten mit Parametern und Volltextsuchbegriffen, z. B. 'Hühnerstall', 'Huhn*', usw.
 - Test der Zugriffsrechte auf Objekte mit Wechsel der User (z. B. Suche nach 'Umbau').
 - Navigation über den Personenindex und von dort zu Dokumenten (z. B. Trappe).
 - Navigation in den Klassifikationen (z. B. Herkunft, Format).
- Tests der Web-Anwendung (Editor-Funktionen)
 - Login als User 'author1A'.
 - Neuanlegen einer Institution, Bearbeiten der Daten im Workflow, Löschen der Institution aus dem Workflow.
 - Neuanlegen einer Institution, ändern der ACLs (mehrer Versionen durchspielen), Hochladen der Institution, Bearbeiten der Daten, Suchen nach den Daten und Löschen der Daten aus dem Server.
 - Neuanlegen einer Person, Bearbeiten der Daten im Workflow, Löschen der Institution aus dem Workflow.
 - Neuanlegen einer Person, ändern der ACLs (mehrer Versionen durchspielen), Hochladen der Person, Bearbeiten der Daten, Suchen und Navigieren nach den Daten, Setzen der ACL-Permission 'deletedb' für 'author1A' und Löschen der Daten aus dem Server.
 - Neuanlegen eines Dokuments (mit Link auf einen Personendatensatz), Bearbeiten der Daten im Workflow, Hinzufügen eines Derivates, Hinzufügen einer Datei zum Derivate, Umbenennen des Derivates und Löschen des Dokuments aus dem Workflow.
 - Neuanlegen eines Dokuments (mit Link auf einen Personendatensatz), Hochladen des Dokuments, Bearbeiten der Daten, Hinzufügen eines Derivates, Hinzufügen einer Datei zum Derivate, Umbenennen des Derivates, Löschen einer Datei aus dem Derivate und Löschen des Dokuments aus dem Server.
 - Logout als 'gast'
- Tests der Web-Anwendung (Klassifikationseditor)

- Login als User 'administrator'.
- Auswahl Menüpunkt Klassifikationseditor starten.
- Neue Klassifikation erstellen (ID DocPortal_class_00000100), Klassifikation speichern, Klassifikationsbeschreibung ändern, zwei neue Kategorien hinzufügen, Reihenfolge tauschen, Kategorie editieren, 'empty' Kategorie löschen, Klassifikation speichern, exportieren und wieder löschen.
- Logout als 'gast'.
- Navigation in den Klassifikationen.
- Tests der Web-Anwendung (Nutzerverwaltung)
 - Aufruf Startseite, Aufruf Login als anderer User, 'Abbrechen', Aufruf Login als anderer User, Login als User 'administrator', Rückkehr zur Anwendung, Benutzer wechseln -> 'Abbrechen', Benutzer abmelden.
 - Login als User 'administrator' und Rückkehr zur Anwendung.
 - Passwort ändern und 'Abbrechen' drücken.
 - Passwort ändern und 'Ändern' drücken. Login mit dem neuen Passwort.
 - Benutzerdaten anzeigen lassen.
 - Neue Gruppe 'test' anlegen, Gruppe administrieren und 2 Benutzer 'test' zuweisen. Neuen Benutzer 'otto' für die Gruppe 'test' anlegen.
 - Logout als 'gast'.
- Tests der Web-Anwendung (ACL-Editor)
 - Login als User 'administrator'.
 - ACL-Editor starten, nach MCRObjectID DocPortal_author_00410901 filtern, 'writedb' auf 'SYSTEMRULE000000006' setzen, OK, Logout als 'gast', Suche nach Person 'Kupferschmidt' – muss jetzt von 'gast' editierbar sein.
 - Login als User 'administrator'.
 - ACL-Editor starten, Rule-Editor aufrufen, 'SYSTEMRULE0000000002' um Gruppe 'test' erweitern, OK. Login as 'test' ant try to change sample data.
 - Logout als 'gast'.
- Tests der Web-Anwendung (Broadcasting Modul)
 - Login als User 'administrator'
 - Module-Broadcasting Monitoring -> edit, 'Power' -> on, ' Message Header' und 'Message Tail' ausfüllen, 'Nachricht an Gruppe' -> admingroup -> ...text... , OK. Login als User 'administrator' (sollte Nachricht erhalten).
 - Module-Broadcasting Monitoring -> edit, 'Power' -> off
- Tests der Web-Anwendung (WebCLI Module)

- Starten des Web-Commandline-Interfaces
- Auswählen von Kommandos wie 'list all users' und kritisches prüfen des Ergebnisses
- Web-Commandline-Interfaces schließen
- Login als gast und erneuter Versuch, das Tool zu starten (darf nicht gehen).
- Stoppen der Anwendung mit 'build/bin/jettystop.sh' bzw. 'build\bin\jettystop'.

6.5.3. Test des WCMS

- [ToDo - 2007-08-29 - Jena]

6.5.4. Test sonstiger Funktionen

- Test Google-Servlet
 - Aufruf von http://localhost:8291/sitemap_google.xml, es sollte eine korrekte Sitemap zurückgeliefert werden.
- Test OAI
 - [ToDo - 2007-09-25 - Rostock]
- Test Z3950
 - [ToDo - 2007-09-25 - Rostock]
- Test MyCoRe Remote Access
 - Starten der Anwendung mit 'build/bin/jettystart.sh' bzw. 'build\bin\jettystart'.
 - ant webservice.deploy
 - Suchmaske nach Dokumenten aufrufen, 'diesem und ausgewählten Servern' sowie 'Lokal via Webservice' und 'DocPortal Sample Server' markieren und die Suche starten. Es müssen die Beispieldaten je dreimal als Treffer erscheinen. Navigation in den Treffern testen.
 - Stoppen der Anwendung mit 'build/bin/jettystop.sh' bzw. 'build\bin\jettystop'.
- Test der Vererbung
 - [ToDo - 2007-09-25 - Leipzig]

6.6. Test der Distribution

Die Tests sind für Unix/Linux und Windows durchzuführen!

- Test der Installation des Basispaketes
- Test der Installation der Content-Pakete
- Test gemäß Punkt 'Test der Web-Anwendung'
- Test des Entfernens der Distribution