



Developer Guide

RELEASE 2.0

30. August 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
1.1	SourceForge.....	1
1.2	CVS-Zugang.....	1
2	Festlegungen für die Code Entwicklung.....	2
2.1	Vorbemerkungen.....	2
2.2	Encoding.....	2
2.2.1	Allgemeines.....	2
2.2.2	Konfiguration unter Eclipse.....	2
2.3	Code Formatierung.....	2
2.3.1	Allgemeines.....	2
2.3.2	Konfiguration unter Eclipse.....	2
2.4	Compilieren und Code-Test.....	3
2.4.1	Allgemeines.....	3
2.4.2	Nutzung der Entwicklungsumgebung Eclipse.....	3
2.4.3	Test mit JUnit.....	4
2.5	Kommentare im Code.....	4
2.5.1	Allgemeines.....	4
2.5.2	Kommentare im Java-Code.....	5
2.5.3	Kommentare in den Stylesheets.....	7
2.5.4	Logging.....	7
3	Namenskonventionen.....	8
3.1	Benennung der Properties.....	8
3.2	Stylesheets.....	10
4	DocPortal vs. MyCoRe.....	11
4.1	MyCoRe.....	11
4.2	DocPortal.....	11
5	Dokumentation.....	12
5.1	Allgemeines.....	12
6	Tests.....	13
6.1	Allgemeines.....	13
6.2	Stufe 0 - Commit während einer Komponentenentwicklung.....	13
6.3	Stufe 1 - Bugfix.....	13
6.4	Stufe 2 - Komponente ist fertig entwickelt.....	13
6.5	Stufe 3 - Release oder Snapshot.....	15
6.5.1	Test des Commandline Interfaces.....	15
6.5.2	Test der Web-Anwendung.....	17
6.5.3	Test des WCMS.....	18
6.5.4	Test sonstiger Funktionen.....	18
6.6	Test der Distribution.....	19
7	Anhang.....	20
7.1	Abbildungsverzeichnis.....	20
7.2	Tabellenverzeichnis.....	21

1 Gemeinsame Entwicklungsplattformen

1.1 SourceForge

Für die Präsentation und den Download von Releases und Distributionen wurde ein Projekt auf den Server von SourceForge¹ 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 Verfügung². 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 Lützenkirchen³ 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 Internet⁴ zu finden.

¹<http://sourceforge.net/projects/mycore>

²siehe MyCoRe-UserGuide

³luetzenkirchen@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 Pakete⁵. Für die Anwendung sollten nur Pakete aus application/build/lib, nicht aus dem MyCoRe-Baum, integriert werden.

⁵z. 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 JUnit⁶. 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 parameter 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 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 sollen, aus.

3 Namenskonventionen

3.1 Benennung der Properties

MyCoRe-Properties beginnen immer mit MCR, dann ein Punkt, 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.Configuration MCR.XMLParser	
ACL	MCR.Access	
Classification Browser	MCR.ClassificationBrowser	
Classification Editor	MCR.ClassificationEditor	
Commandline Tool	MCR.CLI	
Editor	MCR.EditorFramework MCR.Editor	
Event Handler / Searcher	MCR.EventHandler MCR.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-Prefixe

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 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 in 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'.
 - Erzeugen der HSQLDB mit 'ant create.metastore'.
 - 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 hsqldb-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 den 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.

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.
- Tests der Web-Anwendung (Klassifikationseditor)
- Tests der Web-Anwendung (Nutzerverwaltung)
- 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
- Test OAI
- Test MyCoRe Remote Access
- Test der Vererbung

6.6 Test der Distribution

Die Tests sind für Unix/Linux und Windows durchzuführen!

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

7 Anhang

7.1 Abbildungsverzeichnis

7.2 Tabellenverzeichnis

Tabelle 3.1: Liste der Property-Prefixe.....	9
--	---