# Leitfaden für Kitodo-Entwickler

Dieses Dokument ist gemäß §11 der Vereinssatzung auf Beschluss der Mitgliederversammlung des Vereins Kitodo. Key to digital objects e. V. am 31.07.2014 in Kraft getreten und ersetzt mit diesem Stichtag die bisher gültige, von der Mitgliederversammlung beschlossene Fassung vom 17.09.2012.

Dieses Dokument beschreibt die Grundsätze der gemeinsamen Software-Entwicklung für Kitodo. Dazu werden die generelle Entwicklungsmethodik, verbindliche Standards wie zum Beispiel Coding Guidelines sowie die zu verwendenden Werkzeuge und Kommunikationsmittel vorgestellt. Der Leitfaden richtet sich sowohl an individuelle Software-Entwickler, die am Kitodo-Quellcode arbeiten möchten, als auch an Entscheider, die Entwicklungsaufträge erteilen möchten und dabei die Kompatibilität der beauftragten Arbeiten zum quelloffenen Community-Produkt Kitodo gewährleisten wollen.[1]

Die Festlegungen gelten für alle im Umfeld von Kitodo gemeinschaftlich entwickelten Komponenten (Kitodo.Production, Kitodo.Presentation, etc.) soweit sie für das jeweilige Produkt anwendbar sind. Nicht aus der gemeinschaftlichen Entwicklung hervorgegangener Quellcode, der zu einem Kitodo-Produkt beigesteuert wird, muss mindestens die beschriebenen Standards und Bedingungen zur Übernahme von Quellcode erfüllen.

## Dezentrale, funktionsgetriebene Entwicklung

**Die Software-Entwicklung erfolgt im Kitodo-Kontext generell dezentral und funktionsgetrieben.** Nur die jeweils für das Produkt verantwortlichen Release Manager erhalten schreibenden Zugriff auf den zentralen Hauptentwicklungszweig. Alle Weiterentwicklungen und Fehlerbehebungen finden in Quellcode-Zweigen (*Branch*) statt, die jeweils aus dem aktuellen Hauptentwicklungszweig erzeugt werden müssen. **Entwicklungszweige, die nicht aus dem Hauptentwicklungszweig erzeugt wurden, können nicht wieder in diesen übernommen werden!** Ein Entwicklungszweig sollte immer nur zur Erledigung eines einzigen, in sich abgeschlossenen Arbeitspakets dienen (also der Behebung genau eines Fehlers oder der Realisierung einer Funktionalität).

Nach Abschluss der Programmierung und erfolgreichem Test der Änderungen wird der Zweig dann zur Übernahme in den Hauptentwicklungszweig freigegeben (*Pull-Request*). Die Release Manager unterziehen den Quellcode einer abschließenden Revision, wobei die Einhaltung formaler Kriterien geprüft und ein oberflächlicher Funktionstest durchgeführt wird. **Die Revision ersetzt keine ordentlichen Funktions- und Leistungstests, diese liegen in der Verantwortung der Entwickler!** Verläuft die Revision positiv, übernehmen die Release Manager den Quellcode in den Hauptentwicklungszweig, verläuft sie negativ, geben sie den Entwicklungszweig zur Nachbearbeitung an die Entwickler zurück.

Über den Zeitpunkt der Freigabe zur Quellcode-Übernahme entscheidet generell der Entwickler, so dass zum Beispiel kommerzielle Entwicklungen durchaus eine gewisse Zeit zurückgehalten werden können. Da mit zunehmendem zeitlichem Abstand die Quellcode-Übernahme aufwändiger wird,

[1] Gemäß §3.5 der Satzung haben sich Mitglieder des Vereins *Kitodo. Key to digital objects e.V.* insbesondere zur Einhaltung der in diesem Dokument beschriebenen Regeln verpflichtet. Dies gilt auch bei der Vergabe von Entwicklungsaufträgen an Dritte.


c/o Sächsische Landesbibliothek -
Staats- und Universitätsbibliothek Dresden
Post: 01054 Dresden Besucher: Zellescher Weg 18 | 01069 Dresden

sollte eine Freigabe jedoch spätestens 3 Monate nach Fertigstellung der Entwicklung erfolgen.[2] **Eine Ausnahme stellen reine Fehlerbehebungen dar, die stets sofort nach Fertigstellung freigegeben werden müssen, sofern sie einen Fehler in einem öffentlichen Entwicklungszweig betreffen!**

## Koordinierung der Entwicklungsvorhaben

**Sämtliche Entwicklungsvorhaben sollten öffentlich dokumentiert und müssen einem zuständigen Release Manager mitgeteilt werden.** Die Release Manager behandeln diese Mitteilungen auf Wunsch vertraulich. Dies dient der Vermeidung von Parallelentwicklungen und somit einer stärkeren Bündelung von Entwicklungsressourcen der Community sowie einer verlässlichen Release-Planung.

**Da für die Revision von Code-Freigaben und die Übernahme in den Hauptentwicklungszweig Ressourcen des Release Managements benötigt werden, diese aber in der Regel zeitnah erfolgen müssen, sollte der Entwicklungsplan größerer Vorhaben frühzeitig mit dem Release Management abgestimmt werden.** Nur dann kann das Release Management entsprechende Ressourcen zur jeweils benötigten Zeit widmen und dadurch zu einem verzögerungsfreien Projektverlauf beitragen.

## Coding Guidelines

**Entwicklungszweige sollten nur die zur Erreichung des Entwicklungsziels notwendigen Änderungen beinhalten.** Rein kosmetische Anpassungen (z.B. der Formatierung) oder nicht-funktionale Ergänzungen (z.B. der Dokumentation) von ansonsten unveränderten Quellcode-Bereichen sollten daher in einem eigenen Entwicklungszweig vorgenommen werden und nicht mit funktionalen Änderungen einhergehen, um bei der Code-Übernahme in den Hauptentwicklungszweig die in ihrer Funktionalität veränderten Code-Bereiche schnell identifizieren zu können. Auch sollten nicht mehrere Entwicklungsziele innerhalb desselben Entwicklungszweigs realisiert werden. Dies vereinfacht Funktions- und Leistungstests, verbessert die Transparenz des Entwicklungsprozesses und erleichtert die Revision durch die Release Manager.

**Je nach verwendeter Technologie und Produkt sind zudem die folgenden Coding Guidelines einzuhalten.** Diese Standardisierung soll es den Entwicklern erleichtern, sich in fremdem Quellcode zurechtzufinden, und eine gleichbleibend hohe Code-Qualität gewährleisten.

### Allgemeingültige Standards

Diese Vereinbarungen gelten unabhängig von der verwendeten Programmiersprache und für alle Komponenten der Kitodo Suite gleichermaßen.

- Generell ist die Arbeitssprache Englisch, was insbesondere bei der Wahl von Variablen-, Klassen- und Methodennamen sowie der Quellcode-Dokumentation zu berücksichtigen ist.
- Die einschlägigen Regeln zur Wahl von sprechenden Bezeichnern und deren Schreibweise (`camelCase` oder `CamelCase`) sind zu beachten! Eine Ausnahme stellen Konstanten dar, die immer in Großbuchstaben geschrieben werden.
- Generell ist objektorientiert zu entwickeln. Große Klassen und Methoden sind zu vermeiden. Jede Datei sollte nur eine Klasse beinhalten und entsprechend der Klasse benannt sein.
- Die Zeichenkodierung ist generell UTF8, Zeilenumbrüche werden im Unix-Format kodiert, Einrückungen erfolgen mit Tabs[3]. Jede Datei ist mit einem Zeilenumbruch abzuschließen.

[2] Siehe auch §3.5 der Vereinssatzung, wonach diese Frist insbesondere für Vereinsmitglieder gilt.

- Jede Klasse, jede Methode und jede Klassenvariable ist im phpDoc- bzw. JavaDoc-Standard zu dokumentieren. Zusätzlich sollte im Quellcode ausführlich von Zeilenkommentaren zur Dokumentation Gebrauch gemacht werden.

- Statt öffentlicher Variablen sollten immer Getter- und Setter-Methoden verwendet werden. Setter sind immer `void`-Methoden, Getter haben keine Parameter. Innerhalb einer Klasse sollte der Zugriff auf private Variablen immer direkt erfolgen.

- Schleifen (`for`, `while`), Bedingungen (`if`, `else`) und vergleichbare Codeblöcke müssen immer in geschweiften Klammern eingeschlossen werden.

- Exceptions (`error`) müssen immer aufgefangen und geeignet behandelt, mindestens aber im Log registriert werden. Letzteres gilt auch für unkritische Programmfehler (`warning`) und Hinweise (`notice`).

- Alle Änderungen werden in den Commit Messages kurz, aber aussagekräftig dokumentiert. Ein Changelog innerhalb der Quellcodedateien wird dagegen nicht geführt.

## Kitodo.Presentation

Die Präsentationskomponente der Kitodo Digitalisierungssuite bildet eine Erweiterung (*Extension*) für das freie CMS TYPO3. Diese Extension enthält selbst alle wesentlichen Funktionsmodule einer modernen Digitalen Bibliothek, stellt zugleich aber auch eine umfangreiche API zur Entwicklung ergänzender Module (in Form eigener Extensions) bereit. Für die Entwicklung sind die Coding Guidelines des TYPO3-Projekts[4] bindend. Zusätzlich gelten folgende, eventuell abweichende Vereinbarungen:

- Neue Frontend-Plugins und Backend-Module sollten in der Regel in Form eigener Extensions umgesetzt werden, um eine modulare Zusammenstellung der benötigten Komponenten zu ermöglichen. Nur Basisfunktionalitäten sollen Bestandteil der Extension `dlf` sein.

- Die Extension `dlf` stellt den Kern der Präsentationsschicht dar und muss bei eigenen Extensions deshalb stets als Abhängigkeit in `ext_emconf.php` genannt werden. Daraus ergibt sich auch die Systemanforderung von PHP in Version 5.3 oder höher und TYPO3 in Version 4.5 oder höher. Wann immer möglich, sind die APIs von TYPO3 und Kitodo zu nutzen.

- Frontend-Plugins sollten immer von der Klasse `tx_dlf_plugin`, Backend-Module dagegen von der Klasse `tx_dlf_module` abgeleitet werden, die wiederum jeweils von den entsprechenden Standardklassen von TYPO3 abgeleitet sind.

- Die Klassenvariable `$prefixId` muss stets `tx_dlf` lauten, um einen gemeinsamen Namensraum für alle Eingabevariablen zu gewährleisten. Dadurch kann jedes Plugin bei Bedarf auf die Eingaben jedes anderen Kitodo-Plugins zugreifen.

- Sollten Eingriffe in den Prozessablauf der Kernkomponenten nötig sein, sind stets die Release Manager zu kontaktieren. Diese bemühen sich dann um eine entsprechende Anpassung der Kernkomponenten bzw. die Implementierung eines Eingriffspunkts (*Hook*). Nur in begründeten und mit den Release Managern vereinbarten Ausnahmefällen dürfen Klassen der Kernkomponenten als XCLASSES überschrieben werden!

---

[3] Aufgrund eines qualifizierten Minderheitsvotums steht die Regelung der Einrückungen bis zur nächsten Beschlussfassung unter Vorbehalt.

[4] http://typo3.org/documentation/document-library/core-documentation/doc_core_cgl/current/

## Kitodo.Production

Die Produktionskomponente der Kitodo Digitalisierungssuite bildet eine Webapplikation für einen Applikationsserver wie Tomcat oder Jetty. Generell werden hier die Coding Standards von Scott Ambler[5] als bindend betrachtet. Zusätzlich gelten folgende, eventuell abweichende Vereinbarungen:

- Klassendateien sind entsprechend ihrer Package-Zugehörigkeit in der Verzeichnisstruktur abzulegen (z.B. Klasse Foo, Package `org.Kitodo.bar`: `/src/org/Kitodo/bar/Foo.java`).
- Der Quellcode muss kompatibel zum Java6-Standard sein.
- Bezeichnungen für Datenbanken, Tabellen und Spalten sind durchgängig klein zu schreiben.
- Es müssen immer typsichere Listen verwendet werden (`ArrayList<Element> list;` `HashMap<Integer, String> map;`).
- Für Schleifen sollten Java5-Konstrukte für typsichere Listen benutzt werden.
- Wenn möglich sollte statt `null` eine leere Liste zurückgegeben werden.

## Werkzeuge und Kommunikation

**Die zentrale Entwicklungsplattform bildet GitHub[6].** Dort befinden sich das öffentliche Quellcode-Repository, ein Bug- und Featuretracker sowie ein Wiki zur Diskussion und Erarbeitung neuer Konzepte. **Dem zugrunde liegt das Versionsverwaltungswerkzeug Git.** Zusätzlich ist es möglich, über GitHub die Release Manager zu kontaktieren. Zur Nutzung von GitHub ist ein kostenloser Account notwendig.

**Generell sind ausnahmslos alle Programmfehler im Bugtracker von GitHub zu melden, wobei sicherheitskritische Bugs als solche markiert und dann nur von den Release Managern eingesehen werden können.** Auch Entwicklungsvorhaben sind nach Möglichkeit immer dort zu dokumentieren, mindestens aber vertraulich den zuständigen Release Managern mitzuteilen, um unwissentliche Doppelarbeit zu vermeiden.

**Zur Übernahme in den Hauptentwicklungszweig freigegebener Quellcode ist zwingend in Form eines auf dem Hauptentwicklungszweig basierenden Git-Branchs auf GitHub bereitzustellen und mittels der dort verfügbaren Funktion „pull request“ an die Release Manager zu melden. Quellcode-Lieferungen auf anderem Weg können nicht berücksichtigt werden.[7]** Die eigentliche Entwicklung kann auch unter Beachtung der Anforderungen an eine dezentrale, funktionsgetriebene Entwicklung in einem privat angelegten lokalen Zweig stattfinden, der erst nach Abschluss der Programmierung zur Übernahme in den Hauptentwicklungszweig in GitHub eingestellt wird.

*Stand: Juni 2016*

[5] http://www.ambysoft.com/downloads/javaCodingStandards.pdf

[6] https://GitHub.com/Kitodo

[7] Siehe auch §3.5 der Vereinssatzung, wonach sich Vereinsmitglieder zu einer aktiven Unterstützung des Release Managements verpflichten.