Sebastian Meyer, SLUB Dresden

Leitfaden für Goobi-Entwickler

Dieses Dokument beschreibt die Grundsätze der gemeinsamen Software-Entwicklung für Goobi. Dazu werden die generelle Entwicklungsmethodik, verbindliche Standards wie zum Beispiel Coding Guidelines sowie die zu verwendenden Werkzeuge und Kommunikationsmittel vorgestellt.

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

Dezentrale, funktionsgetriebene Entwicklung

Die Software-Entwicklung erfolgt im Goobi-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 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). Existieren für ein Produkt mehrere Entwicklungslinien (*Series*), so gibt es auch mehrere Hauptentwicklungszweige – entscheidend ist also, in welche Entwicklungslinie die eigene Entwicklung einfließen soll.

Sämtliche Entwicklungsvorhaben werden vor Beginn der Arbeiten öffentlich dokumentiert oder zumindest einem zuständigen Release Manager mitgeteilt. Die Release Manager behandeln diese Mitteilungen auf Wunsch vertraulich und geben sie nur an Dritte weiter, wenn andernfalls eine Entwicklung doppelt erfolgen würde.

Nach Abschluss der Programmierung und erfolgreichem Test der Änderungen wird der Zweig dann zur Übernahme (*Merge*) in den Hauptentwicklungszweig freigegeben. Die Release Manager unterziehen den Quellcode einer abschließenden Revision (*Review*), 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, sollte eine Freigabe jedoch spätestens 6 Monate nach Erzeugung des Entwicklungszweigs aus dem Hauptentwicklungszweig erfolgen**. Eine Ausnahme bilden reine Fehlerbehebungen, die stets sofort nach Fertigstellung freigegeben werden müssen, sofern sie einen Fehler in einem öffentlichen Entwicklungszweig betreffen!**

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 Komponeten der Goobi 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 bilden Konstanten, die immer in Großbuchstaben geschrieben werden.
- Generell ist objektorientiert zu entwickeln. Große Klassen und umfangreiche Methoden sind zugunsten einer größeren Granularität zu vermeiden. Jede Datei sollte nur eine Klasse beinhalten und entsprechend der Klasse benannt sein.
- Die Zeichenkodierung ist generell UTF8.
- 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`).

Goobi.Presentation

Die Präsentationskomponente der Goobi 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[1] bindend. Zusätzlich gelten folgende, eventuell abweichende Vereinbarungen:

- 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 4.4 oder höher. Wann immer möglich, sind die APIs von TYPO3 und Goobi 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 Goobi-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!

Goobi.Production

Die Produktionskomponente der Goobi Digitalisierungssuite bildet eine Webapplikation für einen Applikationsserver wie Tomcat oder Jetty. Generell werden hier die Coding Standards von Scott Ambler[2] aus bindend betrachtet. Zusätzlich gelten folgende, eventuell abweichende Vereinbarungen:

- 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.

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

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

Werkzeuge und Kommunikation

Die zentrale Entwicklungsplattform bildet Launchpad[3]. Dort befinden sich das öffentliche Quellcode-Repository, ein Bug- und Featuretracker, ein rudimentäres Diskussionsforum und Mailinglisten sowie ein „Blueprints“ genanntes Werkzeug zur Diskussion und Erarbeitung neuer Konzepte. Dem zugrunde liegt das Versionsverwaltungswerkzeug Bazaar. Zusätzlich ist es möglich, über Launchpad die Release Manager und das Goobi Community Board zu kontaktieren. Zur Nutzung von Launchpad ist ein kostenloser Account notwendig.

Generell sind ausnahmslos alle Programmfehler im Bugtracker des Launchpads 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.

Zur Übernahme in den Hauptentwicklungszweig freigegebener Quellcode ist zwingend in Form eines auf dem Hauptentwicklungszweig basierenden Bazaar-Branchs auf Launchpad bereitzustellen und mittels der dort verfügbaren Funktion „propose for merging“ an die Release Manager zu melden. Quellcode-Lieferungen auf anderem Weg können nicht berücksichtigt werden.

Mit jedem Launchpad-Account können beliebig viele Quellcodezweige in Launchpad eingestellt werden. Dies erfolgt über die Angabe des eigenen Benutzernamens sowie des Projektnamens, dem der Quellcode-Zweig zugeordnet ist, und einer frei wählbaren Bezeichnung. Zum Beispiel würde `lp:~sebastian-meyer/goobi-presentation/bug123456` einen Quellcode-Zweig des Nutzers `sebastian-meyer` namens `bug123456` erzeugen und diesem dem Projekt `goobi-presentation` zuordnen. Grundsätzlich sind persönliche Quellcode-Zweige stets öffentlich lesbar, aber nur vom Benutzer selbst beschreibbar. Die eigentliche Entwicklung kann auch dezentral in einem privat angelegten Zweig stattfinden, der erst nach Abschluss der Programmierung zur Übernahme in den Hauptentwicklungszweig in Launchpad eingestellt wird.

[3] https://launchpad.net/goobi