- Qualitätssicherungskonzept -

Kita Tauschbörse

Version: 1.1

Projektbezeichnung	Kita Tauschbörse	
Projektleiter		
Verantwortliche	K. Jakob, R. Rößling, A.Sifring	
Erstellt am		
Zuletzt geändert	16.04.12 20:17	
Bearbeitungszustand		in Bearbeitung
		vorgelegt
	X	fertig gestellt
Dokumentablage		
V-Modell-XT Version	1.3	

INHALTSVERZEICHNIS

1 DOKUMENTATIONSKONZEPT

Eine umfassende Dokumentation ist integraler Bestandteil einer produktiven Entwicklung und späteren Wartbarkeit unseres Projektes.

1.1 Kommentierung

Um diese erreichen zu können, empfiehlt sich der Einsatz zweier Hilfsmittel. Einerseits die Nutzung von Kommentaren direkt im Quelltext, andererseits ein Dokumentationsgenerator, der aus speziellen Informationen, die innerhalb desselben notiert werden, eine externe, übersichtliche Dokumentation generiert, die dem Projekt zur Erklärung beigefügt wird.
Quasi jede Programmiersprache stellt den Kommentar als Sprachmittel zur Verfügung. Er dient dazu, Anmerkungen zum Programm direkt in den Quelltext schreiben zu können. Die Kommentare werden bei der Weiterverarbeitung zu einem Programm ignoriert, haben also keinen Einfluss auf das entstehende Programm und dienen nur zur Erläuterung bzw. Beschreibung des Quelltextes, um das Verständnis für denselben zu verbessern. Wir werden solche Kommentare in allen von uns verwendeten Sprachen zum Einsatz bringen.
Es existieren für viele Sprachen eignene Dokumentationsgeneratoren mit eigener Syntax für die Informationen innerhalb des Quelltextes. Für die in PHP zu implementierenden Teile des Projektes werden wir phpDocumentor[1] (kurz: phpDoc) verwenden. Er kann die Dokumentation in verschiedensten Formaten von HTML über PDF bis hin zu Windows Hilfedateien (chm) generieren und bietet zudem eine Vielzahl von Designvorlagen für diese Dokumente an.
Einen nicht ganz so flexibelen Dokumentationsgenerator werden wir für JavaScript verwenden. Jsdoc-toolkit bietet nur die Generierung von HTML oder XML Dokumenten an, ist aber ebenso vorlagenbasiert wie phpDocumentator.

1.2 Code-Konventionen

Neben der Kommentierung ist es wichtig, sich in der Entwicklung an gewisse Standards bezüglich des Quelltextes zu halten bzw. sich gewisse Standards selbst zu setzen.
Zuvorderst ist festzustellen, dass die gängige Sprache in der Programmierung Englisch ist. Es wäre natürlich möglich, deutsche Bezeichner zu verwenden, würde aber eine mögliche Weiterentwicklung und Wartung auf den deutschen Sprachraum einschränken. Auch wenn man Hilfe für ein Problem in der Entwicklergemeinde der jeweiligen Programmiersprachen sucht, sind deutsche Bezeichner im Quellcode nicht von Vorteil.
Es muss darauf geachtet werden, für Klassen, Methoden und Variablen „sprechende Bezeichner" zu verwenden. Das bedeutet beispielsweise, dass keine kryptischen Namen gewählt werden, sondern Methodennamen beschreiben, was diese Methoden tun und Variablennamen beschreiben was die Variablen beinhalten. Diese Bezeichner können gegebenenfalls abgekürzt werden, wenn die intendierte Bedeutung erhalten bleibt. Bei der Schreibweise werden wir uns ferner an den geltenden Code-Standards der jeweiligen Sprachen orientieren[2,3,6].Insbesondere bei PHP achten wir zusätzlich auf die Konventionen, die das Zend Framework setzt.
Speziell für uns getroffen haben wir die Entscheidung, dass wir öffnende geschweifte Klammern direkt auf die Funktion, Schleife etc. folgen lassen und nicht erst in der neuen Zeilen. Unsere Einrückung wird mit 4 Leerzeichen geschehen und nicht in Form eines Tabulators. Für einige Komponenten ist eine spezielle Benennung der Klassen notwendig, damit der sogenannte Autoloader des Zend Frameworks Klassen automatisch nachladen kann, sollten sie benötigt werden.
Auf eine Besonderheit der beiden Sprachen JavaScript und PHP muss besonders eingegangen werden. Beide haben eine schwache Typisierung, man legt den Inhalt ihrer Variablen also nicht explizit fest. Innerhalb des Entwicklungsprozesses kann es so für Entwickler schwierig werden, herauszufinden, welchen Inhalt eine Variable innerhalb einer Funktion hat, die vielleicht von einem anderen Entwickler geschrieben worden ist. Um

diesem Problem zu begegnen, sollte der Quelltext angemessen ausdokumentiert werden, besonders an Stellen, an denen nicht auf Anhieb klar wird, welchen Inhalt gewisse Variablen haben könnten.

1.3 Technische Dokumentation

Neben der Quelltextdokumentation wird ebenfalls eine technische Dokumentation des Projektes angefertigt werden, welche in einem versionierten Wiki auf unserer Projektseite auf sourceforge.net abgelegt wird. Diese technische Dokumentation beinhaltet eine genaue Erläuterung der Konzepte des Projektes aus rein technischer Sicht und vermittelt einen Überblick über unser Projekt, um eine schnelle Orientierung neuer Entwickler unabhängig von der Quelltextbasis zu ermöglichen. Da diese Dokumentation, anders als die quelltextnahe Dokumentation, nicht mittels eines Dokumentationsgenerators erstellt werden kann, muss diese Dokumentation manuell gepflegt und auf dem neusten Stand gebracht werden.
Eine weitere Unterstützung bietet abseits der Quelltextdokumentation auch unser Versionskontrollsystem, dem das Projekt unterliegt. In diesem kann man jede Änderung und Erweiterung nachvollziehen und sich so ein Bild vom gesamten Entwicklungsprozess machen.

1.4 Benutzerdokumentation

Nach der erfolgreichen Implementierungsphase unseres Projektes werden wir eine Benutzerdokumentation anlegen, basierend auf den Informationen des Wikis, welche zusätzlich Anleitungen zur Installation und Benutzung des Projektes beinhalten soll.
Diese Dokumentation wird dabei den Schwerpunkt weg von kleinsten technischen Details in Richtung Anwendbarkeit verschieben. Denn es geht dem Benutzer primär um die Frage, wie er das Projekt produktiv einsetzen kann.

2 TESTKONZEPT

Einen nicht minder wichtigen Bestandteil der Entwicklung stellt ein durchdachtes Testkonzept dar. Durch dieses wird eine relative Fehlerfreiheit ermöglicht und die Software auf Belastbarkeit, Wartbarkeit usw. getestet. Wir halten eine Unterteilung in vier Testphasen für nützlich und sinnvoll.

2.1 Komponententest

In dieser Testphase prüft man einzelne Klassen und Funktionen auf Fehler. Dabei werden sämtliche Teile des Quelltextes automatisch getestet, wobei eine vollständige Abdeckung aller Programmzeilen des Quelltexts angestrebt wird. Die Tests und die getesteten Quelltextteile werden dabei von separaten Mitgliedern der Projektgruppe erstellt, so dass keine impliziten Annahmen aus der Implementierung über die zu testenden Quelltextteile getroffen werden können.[4]
Die Automatisierung wird in zwei Formen angestrebt. Zum Einen durch automatische zufallsgesteuerte Tests, bei denen die Eingabedaten anhand eines Generators erzeugt werden und die Ausgabedaten auf bestimmte Eigenschaften überprüft werden und zum Anderen durch Tests mit festen Ein- und Ausgaben.
Für den in PHP geschriebenen Quelltext verwenden wir PHPUnit[5]. Da der JavaScript Teil überschaubar ausfällt und nur eine spezielle - bereits anderweitig ausführlich getestete - Bibliothek anspricht, sind hier in unseren Augen Unit Tests unnötig, da eine manuelle Überprüfung im Rahmen des Möglichen bleibt.

2.2 Integrationstest

Diese Phase beinhaltet eine aufeinander abgestimmte Reihe von Einzeltests, unter anderem Funktionstests und Schnittstellentests, deren Aufgabe es ist, verschiedene, voneinander abhängige Komponenten des Programms im Zusammenspiel zu testen. Es wird angestrebt, bei jeder Fertigstellung einer Komponente einen Integrationstest mit den anderen schon fertiggestellten Komponenten durchzuführen.

2.3 Systemtest

Beim Systemtest werden alle Module zusammengesetzt und das gesamte Softwareprodukt wird getestet, diesmal nicht aus der Sicht der Entwickler sondern aus der Sicht der Benutzer. Dafür wird das System in einer Testumgebung installiert, die die Benutzerumgebung simuliert, um spezifische Probleme erkennen zu können. Es wird auf Erfüllung der gesamten Anforderungen - funktionale wie nicht-funktionale -, die im Pflichtenheft stehen, geprüft und alle Softwarefunktionen durchgeführt.

2.4 Abnahmetest

Beim Systemtest werden alle Komponenten zusammengesetzt und das gesamte Projekt wird getestet. Dieses Mal jedoch nicht aus der Sicht der Entwickler sondern aus der Sicht der Benutzer. Dafür wird das Projekt in einer Testumgebung installiert, die die Benutzerumgebung simuliert, um spezifische Probleme erkennen zu können. Es wird auf Erfüllung der gesamten Anforderungen geprüft und alle Funktionen werden durchgeführt.

3 ORGANISATORISCHE FESTLEGUNGEN

- Der gesamte Quelltext, Versionshistorie (Mercurial Repository) sowie alle im Rahmen des Projektes erstellten Dokumente werden dem Auftraggeber für Wartung, Weiterentwicklung etc. zugänglich gemacht.
- Das in diesem Dokument festgelegte Dokumentationskonzept ist konsequent einzuhalten. Bei oftmaliger Verletzung der genannten Regeln muss der jeweilige Projektteilnehmer darauf hingewiesen werden, um weiteren Verletzungen der Regeln entgegenzuwirken.
- Der Verantwortliche für Qualitätssicherung und Dokumentation (Robert Rößling) wird damit beauftragt, unter Mithilfe aller Projektteilnehmer, die technische Dokumentation zusammenzutragen und die Benutzerdokumentation zu erstellen. Weiterhin hat er die Aufgabe die Qualität aller Dokumentationen zu sichern und gegebenenfalls den Forderungen anzupassen.
- Der Projektleiter (Franz Teichmann) ist damit beauftragt, Zeitpläne während der Entwicklung zu entwerfen, Fristen zu setzen und auf deren Einhaltung zu bestehen, um so einen reibungslosen Entwicklungsprozess zu gewährleisten.
- Es werden wöchentliche Treffen abgehalten, um den aktuellen Projektstatus zu besprechen und die Aufgaben für die folgende Woche zu besprechen und zu verteilen,

4 QUELLEN

[1] phpDocumentor Seite: 3
[2] Konventionen Seite: 3
[3] Konventionen Seite: 3
[4] Softwaretest Seite: 5
[5] PHPUnit Seite: 5
[6] Konventionen Seite: 3
[7] JsUnit Seite: 5