Universität Leipzig
Softwaretechnik- Praktikum
Wintersemester 2014/2015

Qualitätssicherung

Projekt	Virtuelles Partizipatorisches Museum (PVM)
Gruppe	swp15-pvm
Verantwortlich	Falco Kirchner
Projektleiter	Alexander Lieder
Erstellt am	11.01.2015

Inhaltsverzeichnis

1 Dokumentationskonzept

Da in unserem Projekt insbesondere die Implementierungsaufgaben auf mehrere Personen aufgeteilt werden, ist es wichtig, dass der geschriebene Programmcode von mitarbeitenden schnell und richtig verstanden werden kann. Dafür und auch um die Dokumente letztendlich komfortabel zusammenführen zu können, ist verständliche und vor allem einheitliche Dokumentation von großem Vorteil.

Für diesen Zweck verwenden wir PHPDoc und richten uns nach den OntoWiki Coding Standards. Jede Dokumentation, jeder Kommentar, und jeder Datei-, Klassen-, Funktions-, Variablen-, Konstanten- und Funktionsname ist dabei in englischer Sprache anzugeben.

1.1 Extrahierbare Quellcodedokumentation

Allgemeine Formatvorgaben

- Einrückungen geschehen durch 4 Leerzeichen. (nicht mittels Tabulator)
- Zeilen sollten nicht länger als 100 Zeichen sein. Können aber bis zu 120 Zeichen enthalten, wenn das dem Verfasser sinnvoller als ein Umbruch erscheint.
- Dateien, die ausschließlich PHP-Code enthalten sind nicht mit dem abschließenden Schlusstag "?>" zu versehen.

Dokumentation

Jede Datei, Klasse und Funktion ist mein einem Dokumentationsblock, wie in folgendem Schema, zu versehen, der sie in Prosa beschreibt. Die Dokumentation findet in englischer Sprache statt.

Dateien

```
/**

* This file is part of the {@link [link]} project.
*
* @copyright Copyright (c) 2014, {@link http://aksw.org AKSW}
* @license http://opensource.org/licenses/gpl-license.php GNU General
Public License (GPL)
*/
```

Klassen

```
/**
* Short description of class
*
* Long description of class (if any) ...
*
* @category [category]
* @package [package]
* @author [autorname] <[e-mail]>
*/
```

Funktionen

```
/**
*Short description of function
*@param passed parameter values (if any!)
*@return return values (if any!)
*@throws exception throws in this funktion (if any!)
*/
```

1.2 Entwurfs-Dokumentation

Allgemeine Vorgaben

- Methoden bzw. Funktionen und Klassen sind so kurz wie möglich zu gestalten.
- Die Wiederverwendbarkeit von Klassen und Funktionen sollte eine wichtige Rolle im Implementierungsprozess spielen.
- Jede Klasse ist in einer eigenen Datei aufzuführen.
- Funktionen und Klassen, sind direkt nach oder noch während ihres Schreibens zu dokumentieren.

Dateiennamen

Dateiennamen sind aus Buchstaben, Zahlen und Bindestrichen zusammenzusetzen. Von der Verwendung von Zahlen wird allerdings abgeraten.
Dateiennamen sind im UpperCarmelCase anzugeben.
Beispiel: `ThisIsAFile`

Klassennamen

Klassennamen sind aus Zeichen der Standat-ASCII-Tabelle zusammenzusetzen. Sonderzeichen und Umlaute sind nicht zu benutzen und auch von der Verwendung von Zahlen wird abgeraten.
Die Wahl möglichst bezeichnender (sprechender) Klassennahmen ist höher priorisiert als der Verwendung möglichst kurzer Bezeichner.
Beispiel: `ThisIsAClass`

Variablennamen

Variablennamen sind aus Buchstaben und Zahlen und Unterstrichen zusammenzusetzen. Von der Verwendung von Zahlen wird allerdings auch hier

abgeraten. Auf für variablen sind sprechende Bezeichner zu wählen. Ausnahmen sind in Schleifen verwendete Variablen.
Variablennamen sind im LowerCarmelCase anzugeben.
Beispiel: `thisIsAVariable`

Konstanten
Konstanten sind ausschließlich aus Großbuchstaben, Zahlen und Unterstrichen zusammenzusetzen.
Beispiel: `THIS_IS_A_CONSTANT`

Funktionen
Funktionsnamen folgen dem gleichen Schema wie Variablennamen, allerdings sind hier keine Unterstriche zu verwenden.
Beispiel: `thisIsAFunktion()`

Klammern
Öffnende geschweifte Klammern stehen immer in der gleichen Zeile, wie die zugehörige Kontrollstruktur, Klassen- oder Funktionsdeklaration. Schließende geschweifte Klammern stehen in jeweils eigenen Zeilen.
Öffnende runde und eckige klammern sind direkt nach Anweisungs- oder Variablennamen zusetzen. Direkt nach einer öffnenden runden oder eckigen Klammer und direkt vor einer runden oder eckigen schließenden Klammer sind ebenfalls keine Leerzeichen zu setzen.
Rückgabewerte sind nicht zu klammern
Beispiele: `function(arg1, arg3);` `array[index]`

Strings
Strings, die keine Variablen enthalten sind in einzelne Hochkommata („'“) zu setzen. Alle anderen Strings sind in doppelte Hochkommata („"“) zu setzen.
Beispiel: `'This is a String'`

Kommentare
Zusätzlich zur Dokumentation, hat zusätzlich eine Kommentation der Quellcodes stattzufinden. Diese erfolgt direkt über dem kommentierten Anweisungen in einer eigenen Zeile beginnend mit zwei Schrägstrichen („//“) für einen einzeiligen oder beginnend mit Schrägstrich und Stern („/*“) und endend mit Stern und Schrägstrich („*/“) für einen mehrzeiligen Kommentar. Ein Kommentar beginnt stets mit einem Leerzeichen.
Beispiel:
```
// This is a Comment for command1
command1;
/* This is a Comment
   for command2
command2;
```

Null-Wert
Der Null-Wert wird klein geschrieben
Beispiel: `null`

2 Testkonzept

Da unser Projektteam für umfangreiche Test nicht ausreichend besetzt ist und wir ohnehin mit einer recht stabilen und häufig getesteten Plattform, nämlich WordPress, arbeiten, werden wir von aufwändigen Tests absehen und uns auf manuelle Tests beschränken.

2.1 Komponententest

Nach Fertigstellung einer Komponente, eines Bausteines unseres Projektes, wird dieser auf Fehler und mögliche Fehlerquellen beim weiteren Verwenden dieses Moduls getestet. Da es sich jeweils nur um Softwarebausteine handelt und die Tests direkt nach deren Fertigstellung stattfinden, die jeweiligen Programmierer sich also nicht erneut in den Programmcode einarbeiten müssen, lassen sich Fehler schnell lokalisieren.

2.2 Integrationstest

Beim Integrationstest, werden nun die einzeln getesteten Komponenten unseres Projektes in ihrer Zusammenarbeit bewertet und optimiert. Hierbei liegt natürlich ein Hauptaugenmerk auf den Komponentenschnittstellen, und der Kommunikation der Komponenten untereinander. Ein solcher Test ist immer dann sinnvoll, wenn eine neue Komponente fertiggestellt und in das bestehende Projekt eingebettet wurde beziehungsweise werden soll.

2.3 Systemtest

Auf zahlreiche Komponenten- und Integrationstests folgt letztendlich ein abschließender Systemtest, der jede Komponente noch einmal einzeln, aber vor allem alle Projektbausteine in ihrer finalen Zusammenarbeit prüft.

3 Organisatorische Festlegungen

Die Aufgabenverteilung und Besprechungen und Präsentationen über erarbeitete Dokumente findet weiterhin mittels wöchentlicher Treffen (weekly scrums) statt. Innerhalb der Gruppe geschehen Absprachen via E-Mail, Skype und im OLAT. Des Weiteren solle ein einheitlicher Kenntnisstand in der Projektgruppe herrschen und deshalb Einzel- bzw. Teilaufgaben für die jeweils anderen Mitglieder sichtbar sein.