Universität Leipzig
Softwaretechnik-Praktikum
Sommersemester 2014

Qualitätssicherungskonzept

Projekt	Graphical SPARQL Builder
Gruppe	s14.swp.gsb
Verantwortlich	Felix Helfer, Siegfried Zötzsche
Erstellt am	20. Januar 2014

Inhaltsverzeichnis

1 Dokumentationskonzept

1.1 Interne Dokumentation

1.1.1 Sprache

Bezeichner und Kommentare sind auf Englisch zu verfassen. Umlaute und Eszett sind im Quelltext zu vermeiden.

1.1.2 Coding Standard

Das Projekt folgt den Google Style Guides für JavaScript [1] und HTML [2].
Darüber hinausgehende Konventionen, sowie ausgewählte der sich daraus ergebenden Quelltextkonventionen sind im Folgenden aufgeführt.

Einrückung (JavaScript, HTML) Eine Einrückungsebene wird durch zwei Leerzeichen ausgezeichnet. Tabulatoren werden nicht zur Einrückung verwendet.

Sprechende Variablennamen (JavaScript) Der Name einer Variable soll ihren Verwendungszweck erkennen lassen.

Variablendeklaration (JavaScript) Variablen werden zu Beginn ihres jeweiligen Gültigkeitsraums deklariert, z. B. am Beginn einer Funktion.

Geschweifte Klammern öffnen auf gleicher Zeile (JavaScript) Gruppierungen durch geschweifte Klammern werden auf der Zeile geöffnet, auf der die Gruppe definiert wird. Z. B.

```
if (something) {
  // ...
} else {
  // ...
}
```

an Stelle von

```
if (something)
{
  // ...
} else
{
  // ...
}
```

Bezeichner (JavaScript) Funktionen und Variablen beginnen mit Kleinbuchstaben. Klassennamen beginnen mit Großbuchstaben. Die Trennung von Worten erfolgt durch Binnenmajuskel (auch als Camel Case bezeichnet). Konstanten werden komplett mit Großbuchstaben, getrennt durch Unterstrich, benannt. Optionale Funktionsargumente beginnen mit "opt_".
Beispiele:

- `functionNamesLikeThis`
- `variableNamesLikeThis`
- `ClassNamesLikeThis`
- `EnumNamesLikeThis`
- `methodNamesLikeThis`
- `CONSTANT_VALUES_LIKE_THIS`
- `foo.namespaceNamesLikeThis.bar`
- `filenameslikethis.js` oder `fileNamesLikeThis.js` (für Dateinamen soll in diesem Projekt, abweichend vom Google JavaScript Style Guide, auch die Verwendung von Binnenmajuskeln zugelassen werden)

Funktions- und Klassenlänge (JavaScript) Die Länge einer einzelnen Funktion ist auf 15 Zeilen beschränkt, die einer Klasse auf 120.

Apostroph gegenüber Anführungszeichen bevorzugen Soweit möglich wird ' bevorzugt gegenüber " verwendet. Die Verwendung von " ist jedoch in Fällen wie dem folgenden Beispiel akzeptabel.

```
var string = 'Toller String';
var json = '{"foo":"bar"}'; // In JSON
var htmlElement = '<a href="example.com">Example</a>'; // Oder HTML-Elementen
```

Nur Kleinbuchstaben (HTML) Der gesamte Quelltext ist in Kleinbuchstaben zu verfassen. Ausgenommen sind lediglich Kommentare und Literale als Attribut-Werte.

1.1.3 Dokumentationsgenerator

Zum Generieren der Quelltext-Dokumentation für JavaScript wird der Dokumentations-Prozessor JSDoc3 verwendet [3]. JSDoc bezeichnet nicht nur die Software zum generieren der Dokumentation sondern auch die gleichnamige Metasprache zum Kommentieren von JavaScript. Da der in diesem Projekt verwendete Google JavaScript Style Guide die Verwendung von JSDoc vorschreibt liegt die Dokumentations-Generierung mittels JSDoc3 nahe.

1.1.4 Qualitätssicherung

Als Werkzeug zur Qualitätssicherung kommt JSLint zum Einsatz. JSLint überprüft JavaScript-Quelltext auf syntaktische Fehler und stilistische Schwächen und ist sowohl als Online-Tool [4] als auch als Kommandozeilen-Tool verfügbar [5]. JSLint bietet eine Reihe von einstellbaren Optionen, beispielsweise zur Festlegung der gewünschten Einrückung und soll als Hilfsmittel verstanden werden, die selbst auferlegten Quelltextkonventionen einzuhalten.

1.2 Produktions-Dokumentation

Die Arbeit am Projekt wird sowohl durch die wöchentliche Aufwandserfassung als auch durch die Verwendung des Versionskontrollsystems Git dokumentiert.

1.3 Entwurfs-Dokumentation

Die Entwurfs-Dokumentation dient sowohl den Projektteilnehmern zum Festhalten getroffener Struktur- und Design-Entscheidungen als auch Aussenstehenden als Einführung in das Projekt. Die Entwurfs-Dokumentation ist auch als Technische Dokumentation der grundlegenden Systemarchitektur zu verstehen.
Sie umfasst eine Beschreibung der Zielstellung und Motivation des Projekts. Sie beschreibt den äußerlichen Funktionsumfang des Systems, also die Perspektive des Anwenders. Weiter werden die angewendeten Struktur- und Entwurfsprinzipien, sowohl des Gesamtsystems als auch von Teilsystemen, beschrieben, erläutert und begründet. Dies dient als Einstiegsshilfe in das Softwaresystem, bevor sich den Details der Umsetzung zugewendet wird. Ebenfalls dokumentiert wird das Datenmodell und das Testkonzept der Software. Ein Glossar ist ebenfalls Bestandteil der Entwurfsdokumentation.

2 Testkonzept

Ein Projekt diesen Umfangs bedingt auch ein hohes Maß an Komplexität – was wiederum unvermeidlich zu Fehlern im Entwicklungsprozess führt. Somit ist es unabdingbar, einem entgegenwirkenden Konzept für ein kontinuierliches Testen zu folgen, welches dazu dienen soll, Fehler frühzeitig und systematisch zu identifizieren und effektiv zu entfernen. Dafür werden folgende unterschiedliche Testphasen unterschieden:

Komponententest Komponententests (oder auch Unit-Test) betrachten einzelne Komponenten der Anwendung isoliert vom Gesamtkonstrukt. Dabei werden besonders programmierte Methoden, Klassen, Skripte unter verscheidenen Einflussparametern auf die Probe gestellt, also komponenteninterne Fehler gesucht. Durch diese losgelöste Bearbeitung wird ein externer Einfluss so weit wie möglich ausgeschlossen.

Integrationstest Ein Integrationstest betrachtet vor allem die Interaktion mehrerer Komponenten untereinander, also deren Funktionsschnittstellen, bzw. Kommunikation

untereinander. Ein solcher Test ist damit besonders dann sinnvoll, wenn eine einzelne Komponente fertiggestellt wurde, und es nun auf deren Einbettung in bereits vorhandenen Anwendungsstrukturen ankommt. Somit sollten fertige Komponenten stets auf reibungslose Integration geprüft werden.

Systemtest Beim Systemtest schließlich werden alle Komponenten der Anwendung zusammengeführt und auf Funktionsfähigkeit geprüft. Als Spezialfall des Systemtest kann man hierbei den Abnahmetest ansehen, der quasi den finalen Test vor der Abnahme des Programms darstellt.
Für die Qualitätssicherung dieses Projekt werden also zunächst einzelne Komponenten auf ihre 'interne Qualität' hin getestet, und schließlich die Zusammenführung dieser. Für Komponenten- bzw Integrationstests bietet AngularJS dabei bereits eigene, potente Werkzeuge an [6]. Da es das vorgesehene Framework des Projekts ist, liegt es nahe, auch dessen interne Tests zu nutzen. Alternativ, bzw. ergänzend wäre QUnit eine zweites mögliches Unit-Test-Framework für JavaScript [7]. Für Systemtests dagegen bietet Protractor ein Testframework für AngularJS-Applikationen [8]. Dabei kann die Anwendung direkt im Browser getestet und mit ihr aus der Sicht eines Users interagiert werden. Schließlich stehen natürlich auch Debugging-Tools von Entwicklungsumgebungen wie etwa Eclipse zur Verfügung. Wünschenswert bei der Durchführung von Tests ist eine größtmögliche Automatisierung, besonders im Falle der Komponententests (bzw. Unittests), um den nötigen Arbeitsaufwand möglichst gering zu halten. Identifizierte Fehler sind zu dokumentieren, sowie, wenn möglich, deren Ursachen und Bereinigung. Dadurch kann die Bereinigung ähnlicher Probleme erleichtert werden, und auch bei der Suche als Anhaltspunkte für mögliche Fehlerquellen dienen.
Für dieses Projekt durchaus relevant ist außerdem, als Variation des Systemtests, der sogenannte DAU-Test (Dümmster anzunehmender User), welcher den Test der Anwendung durch einen außenstehenden, der Thematik möglichst fremden, Laien vorsieht, um besonders Nutzerfreundlichkeit, Intuitivität und auch mögliche Ausfälle bei falscher Benutzung auf die Probe zu stellen.

3 Organisatorische Festlegungen

Die bereits etablierten, wöchentlichen Gruppentreffen sollen weiterhin stattfinden, in der vorlesungsfreien Zeit wird die Teilnehmerzahl dabei angepasst an tatsächlich anwesende Mitglieder und den generellen Bedarf für ein gemeinsames Treffen. Sie dienen der Besprechung und Verteilung von zu bearbeitenden Aufgaben, auftretenden Problemen und bearbeiteten Ergebnissen. Außerdem ermöglichen sie, falls nötig, eine Absprache mit den Betreuern. Außerdem können Absprachen über EMail, WhatsApp und Skype erfolgen.
Es sollte ein einheitlicher Kenntnisstand aller Mitglieder angestrebt werden und die einzelnen Teilarbeiten auch während der Bearbeitung des Einzelnen möglichst einsehbar zur Verfügung stehen. Dies soll besonders der gegenseitigen Unterstützung dienen. Es ist natürlich aber unabdingbar, bei Problemen mit der zugeteilten Aufgabe, Zeitmangel, etc. rechtzeitig Bescheid zu geben, sodass entsprechende Hilfe / Anpassungen der Aufgaben-

verteilung möglich sind. Besprochenes und Bearbeitetes wird im Wiki, bzw. besonders im Bezug auf Quellcode mithilfe des Versionsverwaltungssystems git festgehalten.
Alle Projektteilnehmer sollen sich an die vereinbarten Dokumentationsvorgaben halten. Dies sollte von den entsprechenden Verantwortlichen stichprobenartig geprüft werden, besonders vor wichtigen Abgabeterminen. Die wöchentliche Aufwandserfassung ist von jedem Mitglied im entsprechende GoogleDoc rechtzeitig zu aktualisieren.

Literatur

[1] http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml

[2] http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml

[3] https://github.com/jsdoc3/jsdoc

[4] http://www.jslint.com

[5] http://www.jshint.com/install/

[6] http://docs.angularjs.org/guide/dev_guide.unit-testing

[7] http://qunitjs.com/

[8] https://github.com/angular/protractor