Kodierrichtlinien

FH-Complete Technikum Wien

FH Technikum Wien
Wien, 16. November 2012

Inhaltsverzeichnis

1 Übersicht

1.1 Motivation

Dieses Dokument wurde erstellt um eine einheitlich Enwicklungsumgebung zu schaffen und die Vorgaben der Entwickler zu erhalten.

Vorteile bei der Einhaltung der Richtlinien sind eine raschere Einarbeitung ins System sowie der Erweiterung von Programmteilen.

1.2 Entwicklungsumgebung

Die Standard Entwicklungsumgebung am Technikum-Wien ist Eclipse PDT (PHP Development Tools Project) oder Netbeans.

http://www.eclipse.org/pdt/ (Stand 16.11.2012)
http://netbeans.org/features/php/ (Stand 16.11.2012)

1.3 Ordnerhierarchie

Ordnerhierarchie des FH-Complete Systems

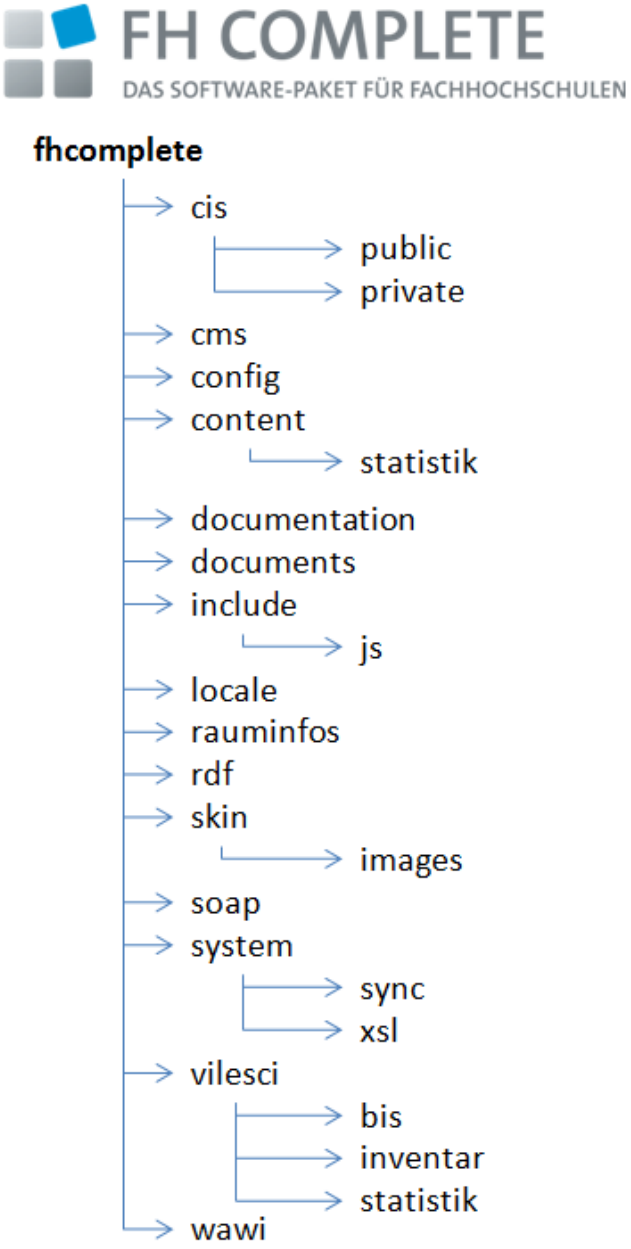

Abbildung 1.1: Ordnerhierarchie von FH-Complete

/**cis:** Dateien speziell für das CIS System

- /**public:** öffentlich sichtbar
- /**private:** Student/User muss eingeloggt sein um Content zu sehen (.htaccess)

/**cms:** Dateien für das CIS Content Management System und Document Management System

/**config:** In diesem Ordner sind alle Konfigurationsdateien gespeichert (CIS, WAWI, VILESCI, Global)

/**content:** XUL Dateien FAS / TEMPUS / PLANNER

- /**statistik:** Statistiken

/**dokumentation:** Dokumentation und Benutzerhandbücher aller Systeme

/**documents:**

/**include:** Klassen und FH Spezifisches

- /**js:** JavaScript Bibliotheken (JQuery)
- /**tw:** Menüeinträge für Technikum Wien

/**locale:** Sprachdateien für FAS oder Sprachenmodul

/**rauminfos:** Rauminfos aller Räume in HTML Files

/**rdf:** RDF Files, zur PDF Erstellung benötigt (Zeugnis, Bestellschein,..)

/**skin:** CSS Datein

- /**images:** Grafiken und Bilder für die Systeme

/**soap:** Alle Dateien für Webservices

/**system:** Verwaltungsskripte

- /**sync:**
- /**xsl:**

/**vilesci:** Dateien speziell für VILESCI

- /**bis:** Bismeldungsrelevanten Daten
- /**inventar:** Inventarprogramm
- /**statistik:** Statistiken

/**wawi:** Files für WaWi

1.4 Serverübersicht

Sirene: CIS, FHComplete, Moodle, Opus
Fhctemp: FAS, Tempus, Vilesci
Theseus: Datenbankserver (Postgres, MySQL)
Calva: Entwicklungsserver

2 Entwicklung-Software

2.1 Namensgebung

Eine einheitliche Form der Namensgebung für Variablen, Konstanten und anderer Komponenten erleichter es dem Entwickler, den Code des anderen zu verstehen.

2.1.1 Dateinamen

Alle Dateinamen haben die für ihren Dateityp **bestimmte Endung**. So haben z.B. HTML Dateien die Endung .html oder PHP Dateien .php. Es gibt jedoch Datentypen die eine **eigene Schreibweise** haben. Hier eine Aufzählung der wichtigsten:

Klassen:	<klasse>.class.php
PHP-File:	<name>.php
HTML-File:	<name>.hmtl
RDF-File:	<name>.rdf
JavaScript:	<name>.js
CSS:	<name>.css
XUL:	<name>.xul
Config:	<name>.inc.php

Hierzu sind Endungen wie <name>.rdf.php oder <name>.js.php erlaubt.
Es sind nur **alphanumerische Zeichen, Underscores und Trennstriche** erlaubt. Beispiele für gültige Namen sind:

- studiengang.class.php
- bestellung.rdf.php
- tablesort.css
- kontakt.js.php
- ToDo_CIS.html

2.1.2 Variablen

Alle Variablen müssen mit **Kleinbuchstaben beginnen** und der „**camelCaps**“ Namenskonventation folgen. Das bedeutet, wenn ein Variablenname aus mehereren Namen besteht muss der Anfangsbuchstabe von jedem **neuen Wort großgeschrieben** werden. Repräsentiert die Variable eine ID so wird die Endung ID mit _ angefügt. Variablennamen sind so **kurz** und so **verständlich** wie möglich zu halten. Variablen wie „$i“ und „$j“ dürfen nur bei Loops verwendet werden.

Variablen in Klassen müssen immer so heißen wie sie in der Datenbank angelegt wurden.

```
$anzahlVariablen
$datum
$person_id
```

2.1.3 Konstanten

Pfade werden in Konstanten immer mit / beendet

```
define('SERVER_ROOT','http://calva.technikum-wien.at/');
```

2.1.4 Session Variablen

Session Variablen die nur ein bestimmtes **Modul** betreffen, werden folgendermaßen benannt: **[Modulname]** / **[Name der Variable]**. Andere Session Variablen, die das ganze System betreffen werden ohne Modulnamen geschrieben.

```
$_SESSION['cms/menu']
$_SESSION['wawi/user']
$_SESSION['user']
```

Eine Session Variable wird **immer klein geschrieben**.

2.1.5 Funktionen und Methoden

Funktionsnamen müssen immer mit einem **Kleinbuchstaben beginnen**. Wenn eine Funktion aus mehreren Namen besteht, muss der Anfangsbuchstabe von jedem **neuen Wort großgeschrieben** werden. Funktionen und Methoden müssen so **klar** wie möglich bezeichnet werden, damit anhand des Funktionsnamen jeder versteht, wofür diese bestimmt sind. Optionale Übergabeparameter müssen immer mit null initialisiert werden.
Hier ein paar **Beispiele** für gültige Funktionsnamen:

```
getAllDetailsFromBestellung()
load()
saveTags($tag, $visible=null)
```

2.2 Kommentare

Kommentare die nur eine Zeile lang sind sollen mit // beginnen. Für Kommentare, die länger als eine Zeile sind gilt als Start die /* Zeichenfolge und als Beendigung */.

2.2.1 Klassen

Jede Klasse muss im Minimum einen **Docblock** mit einer Beschreibung enthalten. Mit einem Docblock Kommentar bezeichnet man spezielle Kommentare, die sich automatisch generieren um PHP Abschnitte genauer zu kommentieren. Diese beginnen mit /** und es können spezielle tags wie @param oder @return benutzt werden.

```
/**
 * Short description for class
 *
 * Long description for class (if any)...
 */
```

2.2.2 Funktionen und Methoden

Jede Funktion oder Klassenmethode muss im Minimum einen **Docblock** mit den folgenden Tags enthalten: **Beschreibung, Parameter und Rückgabewerte**.

```
/**
 * Short description for the function
 *
 * Long description for the function (if any)...
 *
 * @param  array  $array  Description of array
 * @param  string $string Description of string
 * @return boolean
 */
```

2.2.3 Variablen

Jede Klassenvariable muss im Minimum einen **Docblock** oder **normalen** Kommentar enthalten welche den Typ der Variable ersichtlich macht. Wenn erforderlich kann dieser auch eine **Beschreibung** der Variablen enthalten:

```
/**
 * Variable Description
 * @var array
 */
```

2.2.4 Programmkopf

```
/* Copyright (C) 2011 Technikum-Wien
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of the
 * License, or (at your option) any later version.
```

```
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307,USA.
 *
 * Authors: Christian Paminger <christian.paminger@technikum-wien.at>,
 *     Andreas Oesterreicher <andreas.oesterreicher@technikum-wien.at> and
 *     Karl Burkhart <burkhart@technikum-wien.at>.
 */
```

Zu beginn jedes Skriptes soll ganz oben der **GNU Header** eingefügt werden.
Im **Authors Block** werden nur Programmierer erwähnt die das aktuelle File auch wirklich bearbeitet haben.

2.3 Programmierstil

2.3.1 Strings

Ein String wird mit echo ausgegeben und sollte grundsätzlich mit **einfachen Apostrophen**(single quotes) abgegrenzt werden. Dies hat den Vorteil, dass PHP Code im String **nicht ausgeführt wird** und so z.B. HTML Ausgaben einfacher und schneller sind. Eine Folge von auszugebenden Strings kann mit Beistrich getrennt werden (anstatt mit . verknüpft), wodurch die Ausgabe beschleunigt wird

```
$message = 'Hello World';
echo '<div style="float: right;">',$message,'</div>';
echo '<div style="float: right;">'.$message.'</div>';
```

2.3.2 IF - ELSE

Um Blöcke zu gruppieren, können **zusätzliche Klammern** eingesetzt werden. Bei längeren Bedingungen werden alle Bedingungen **untereinander dargestellt**, um die Lesbarkeit zu erhöhen. Die öffnende geschweifte Klammer wird immer in die **nächste separate Zeile geschrieben**. Die abschließende geschweifte Klammer erhält ebenfalls immer eine separate Zeile. Jeder Code innerhalb der geschweiften Klammern muss **einen Tabulator** (standardmäßig vier Leerzeichen) eingerückt werden.

```
if($user == 'Herbert')
{
   echo 'Hallo Herbert';
}
```

```
else
{
   echo 'Falscher User';
}
```

Bei sehr kurzen „if-else“ Konstrukten kann auch der **dreifach konditionale Operator** verwendet werden:

```
$sampleVar = isset($_GET['sampleVar']) ? $_GET['sampleVar'] : '';
```

Gibt es nur eine ausführende Zeile, so können die geschweiften Klammern auch **weggelassen werden**.

```
if($user == 'Herbert')
   echo 'Hallo Herbert';
else
   echo 'Falscher User';
```

2.3.3 Datenbankabfragen

- Datenbankabfragen sind grundsätzlich über die Datenbankklasse abzusetzen.
- Um SQL-Injections zu verhindern, sind die Paramter mittels der Funktion db_add_param zu escapen. (Mögliche Übergabeparameter sind FHC_INTEGER, FHC_STRING und FHC_BOOLEAN)
- SQL Statements sind mit Strichpunkt abzuschließen.
- Vor jedem Tabellennamen ist das entsprechende Schema anzugeben
- Schlüsselwörter (zB SELECT, WHERE, FROM, GROUP BY) sind groß zu schreiben

```
$qry = "SELECT * FROM public.tbl_person WHERE
person_id=".$db->db_add_param($person_id, FHC_INTEGER, false).';';
```

Boolean Attribute sind ber die Datenbankklasse zu parsen um inkompatibilitäten mit verschiedenen DB-Systemen zu verhindern.

```
$aktiv = $db->db_parse_bool($row->aktiv);
```

2.3.4 HTML-Ausgaben

HTML Attribute sollen generell mit doppelten Hochkomma umschlossen werden. Variablen die innerhalb des HTML-Codes ausgegeben werden sind mit der Funktion convert_html_chars zu escapen. Dies ist vorallem wichtig bei Strings die den Typ Text oder varchar in der Datenbank haben.

```
echo '<div style="float: right;">',$basis->convert_html_chars($message),'</div>';
```

2.3.5 Sonstiges

- **Geschwungene Klammern**
  Um einen Block zu definieren werden die geschwungenen Klammern **immer** in der neuen Zeile gesetzt.
- **Tabulator**
  Es wird immer wenn möglich mit Tabulatoren eingerückt. Tabulatorbreite = 4 Leerzeichen.

3 Musterklasse

```
<?php
/* Copyright (C) 2011 Technikum-Wien
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of the
 * License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307,USA.
 *
 * Authors: Christian Paminger <christian.paminger@technikum-wien.at>,
 *      Andreas Oesterreicher <andreas.oesterreicher@technikum-wien.at> and
 *      Karl Burkhart <burkhart@technikum-wien.at>.
 */
 require_once(dirname(__FILE__).'/basis_db.class.php');


/**
 * Klasse Zweck liest und manipuliert Daten aus bis.tbl_zweck
 */
class zweck extends basis_db
{
    /**
     * True wenn neues Objekt, False wenn Objekt schon vorhanden
     * @var boolean
     */
    public $new = true;

    /**
     * Zweck-Objekt Array
     * @var array
     */
    public $result = array();
```

```
    /**
     * Gibt alle Zweckbeschreibungen zurueck, bei Erfolg True
     *
     * @param  integer $id  Id die geladen warden soll, wenn keine id uebergeben wurd
     * @return boolean
     */
    public function getAll($id = null)
    {
       return true;
    }
}
?>
```

4 Code Reviews

20. Mai 2011 - abschlusspruefung.class.php

- Getter und Setter
  - **Wichtige** Attribute protected
  - Realisierbar mit **__set** und **__get**

```
public function __set($name, $value)
{
   $this->vars[$name] = $value;
}
```

- Klasse soll alles erledigen
  - Konstruktor setzt $new = true
  - Funktion load() setzt $new = false
  - Save() unterscheidet true oder false (insert oder update)
- Allgemein
  - Immer require_once() verwenden
  - Im Skriptkopf nur den Autor vermerken, der File wirklich bearbeitet hat
  - Über Klasse Kommentarblock

10. Juni 2011

- Allgemein
  - Ordnerstruktur verbessern
    - local/cms Dateien in -> local/[sprache] verschieben
  - Phrasenmodul
    - Groß- Kleinschreibung bei Name z.B.: ['global/fehlerBeiDerParameteruebergabe']
    - Keine Sonderzeichen in Name
  - Variablen
    - Kein _ vor private oder protected Variablen
    - Docblock Kommentare bei Variablen nicht ntig, Kommentar mit **Datentyp in Datenbank** reicht

07. Oktober 2011 - rdf.class.php

- SOAP
  - username / passwort optional in WSDL
  - Objekt Array -> Complextype wenn mehere Parameter übergeben werden
- XUL
  - menuitemID = Element-Modul z.B.: menuitem-projekt
  - Die Trennung erfolgt mit -, da beim markieren davor und dannach abgebrochen wird
- RDF
  - kein /**alle** mehr -> sondern http://www.technikum-wien.at/[Datenbankname]
- Adressoperator &
- statische Variablen
- Configs säubern und global.conf [include] für globale Konfiguration bereitstellen

11. November 2011 - bestellung.php - Projekt zu Bestellung zuordnen

- Anführungszeichen bei echo
  - Es sollen die Einfachen Anführungszeichen verwendet werden [’] und zur Stringverkettung der Beistrich [,]

    ```
    echo 'Username: ',$user,' eingeloggt';
    ```
- Einfache Apostroph [’] in String escapen -> htmlentities => eigene Funktion bauen
- Magic Quotes -> deprecated
- Session Authentifizierung im CIS / Kalenderschnittstellen
- Cross Side Scripting Schwächen ausarbeiten

16. Dezember 2011 - addslashes - String escapen

- Einfache Anführungszeichen bei echo und doppelte für Eigenschaften
- Eine funktion db_null_value soll in die DB Klasse implementiert werden
  - Bei einem leeren String wird null geschrieben
  - es soll ein Parameter $quote übergeben werden(defaultwert true) -> wenn false wird nicht gequotet (für Integer)
- Daten die aus DB kommen, sollen mit escaped werden(doublequotes ["] werden escaped) -> Quellcode soll nicht veränderbar werden - Hierfï¿œr steht in der Basisklasse die funktion convert_html_chars zur Verfï¿œgung
- ! Auch quoten bei Integerzahlen in DB query? !!

10. Februar 2012 Escapen von DB Parametern - functions.inc.php ausmisten

- Zum Escapen von DB Parametern steht nun eine neue Funktion zur Verfï¿œgung: $db->db_add_param($var, $type, $nullable)

- Boolean Parameter ueber DB-Klasse parsen, damit keine Probleme mit anderen Datenbanken auftreten. (zB t/f bei Postgres 0/1 bei MySQL)
  - Boolean die aus der Datenbank gelesen werden, sollen durch die funktion $db->db_parse_bool($var) geparst werden
  - Boolean die in die Datenbank geschrieben werden, sollen die Funktion $db->db_add_param($var, $type, $nullable) verwenden
- functions.inc.php säubern
  - Datumsfunktionen in datum.class.php auslagern
  - eventuelle Auslagerung der Authentifizierungsfunktionen in eine eigene Klasse

09. August 2012 Strichpunkt bei SQL Statements

- SQL Statements im Code sind mit Strichpunkt am Schluss abzuschließen

08. November 2012 - XSS Lücken und WYSIWYG

- XSS-Lücke im CIS wo es möglich ist fremde Webseiten in den main Frame einzuschleusen
  - Lösung: Überprüfung ob APP_ROOT in URL vorkommt - wenn nicht dann Seite nicht laden und Email an uns senden
- In den LV Infos können Tags wie eingegeben werden -> soll überarbeitet werden
- Anstatt dass jedes Jahr das gesamte CIS ins Archiv kopiert wird, sollen nur mehr die LV Infos weggespeichert werden
- Im WYSWYG Editor kÃ¶nnen Tags eingegeben werden. Auch <script> Tags
  - SaveHTML darüberlaufen lassen
  - vor Speichern in DB durch Methode Tags rausfiltern
- EMail Signaturen sollen in Zukunft als Phrase ausgelagert werden -> Auch fÃŒr Community leichter zum Editieren

- Nächstes Codereview: Basis Klasse und Stundenplanklasse (Zusammenhang zu anderen Klassen)