**Kodierrichtlinien für das Projekt Amberg**

Inhalt

[Allgemeines 1](#_Toc468899124)

[Klammern 1](#_Toc468899125)

[Kommentare 2](#_Toc468899126)

[Formatierung 2](#_Toc468899127)

[Namensgebung 2](#_Toc468899128)

[Camel Case 2](#_Toc468899129)

[Pascal Case 2](#_Toc468899130)

[Oberfläche 3](#_Toc468899131)

[HTML 3](#_Toc468899132)

[CSS 4](#_Toc468899133)

# Allgemeines

Programmiersprachen PHP (PHP5), MySQL, HTML5, CSS3, JavaScript(nur wenn unbedingt nötig/nur wenn PHP nicht ausreicht oder es verkompliziert)

Codiersprache : Englisch

Entwicklungsumgebung : frei wählbar (Atom, Notepad ++, …)

Server: xampp – phpmyadmin

Oberfläche: Deutsch

# Klammern

* Geschweifte Klammern in neuer Zeile oder erste Klammer hinter Anweisung

private void line\_Changed(object sender, EventArgs e)

**{**

**}**

private void line\_Changed(object sender, EventArgs e)**{**

**}**

* Setze optionale Klammern **immer**

If (Bedingung)

{

Anweisung;

}

# Kommentare

* Funktionen/Klassen immer mit Kommentaren versehen, zwecks Verständlichkeit
* Kommentare möglichst oberhalb der Anweisungen platzieren, und nicht in der selben Zeile

# Formatierung

* Neuen Block mit einem Tab einrücken

# Namensgebung

## Camel Case

Erster Buchstabe klein, darauffolgende Worte groß

zB. backColor

## Pascal Case

Erster Buchstabe groß, darauffolgende Worte groß

zB. BackColor

(<https://msdn.microsoft.com/en-us/library/x2dbyw72%28v=vs.71%29.aspx>)

|  |  |  |
| --- | --- | --- |
| Art | Namensgebung | **Beispiel** |
| Klasse | **Pascal Case** | **public** class StreamReader |
| Interface | **Pascal Case**  Mit großem I beginnen | **public** **interface** IEnumerable |
| Namespace | **Pascal Case** | **namespace** System**.**Security |
| Funktionen | **Pascal Case** | ToString**();** |
| Eigenschaften | **Pascal Case** | **public** int Length  **{**  get lenght**;**  **}** |
| Events | **Pascal Case**  -ing bevor  -ed danach | WindowClosed**()**  WindowClosing**()** |
| Quelldateien | **Pascal Case**  Nur eine Klasse pro Quelldatei, die den Namen der Klasse/Interface entspricht. |  |
| Enum | **Pascal Case** | **public** enum FileMode  **{**  Append**,**  **...**  **}** |
| Private Felder | **Camel Case** | **private** string familyName; |
| Funktions Parameter | **Camel Case** | Type GetType(string typeName) |
| Lokale Variablen | **Camel Case** | **Int** index |
| Klassenvariablen | Camel Case + this | **this.index** |

## Oberfläche

Benennung der Oberflächenelemente mit jeweiligen Typ Präfix ( vgl. Oberflächenprogrammierung)

|  |  |
| --- | --- |
| Button | btn |
| RadioButton | rbtn |
| ComboBox | cmb |
| ListBox | lst |
| TextBox | txt |
| Label | lbl |
| GroupBox | grp |
| CheckBox | chk |
| Form | frm |
| Control | ctl |
|  |  |
|  |  |

Bsp.: btnOkay, txtName

\*Anmerkung: Beispiele sind bis hier hin auf Basis von C#, können aber so auch in PHP umgesetzt werden.

# HTML

* Neue Blöcke mit Großbuchstaben kommentieren (<!--NAVBAR-->)
* Komplizierte Blöcke zusätzlich mit Beschreibung kommentieren (<!--Komplizierte Navb….-->)
* IDs immer klein schreiben (bei Bedarf mit – verbinden)

Bsp. :

<div id="footnote-collection">

<button id="btnOkay-form1">

< button id="btnOkay">

* Klassen immer klein schreiben (bei Bedarf mit – verbinden)

Bsp. :

<div class="div-class">

* Sobald in einer Komponente mehrere „Anweisungen“/Bausteine enthalten sind, diese **immer** in neuer Zeile öffnen und schließen.

Bsp.:

<!doctype html>

<html>

<head>

</head>

<body>

<header>

<h1>Name unserer Seite</h1>

</header>

<nav>

<h2>Navigation</h2>

<ul>

<li><a...> Link 1</a></li>

<li><a...> Link 2</a></li>

<li><a...> Link 3</a></li>

</ul>

</nav>

</body>

</html>

# CSS

* Nach jedem Klassen/IDs eine Leerzeile
* Schließende Klammer immer in neuer Zeile
* Zusammenhängende Klassen/IDs bei Beginn Kommentieren (mit 5 Strichen vorher/nachher kennzeichnen)
* Zusammenhängende Klassen/IDs untereinander
* Unverständliche/kompliziertere Klassen/IDs kurz erklären/kommentieren