DokuWiki Plugins - Grundlagen

Dieses Projekt kann keine vollständige Einführung in die PHP- und DokuWiki-Plugin-Programmierung sein. Wir bewegen uns innerhalb eines bestehenden Frameworks von DokuWiki Funktionen und Rahmenbedingungen und lernen, uns in dieser Umgebung zurechtzufinden, um unsere Ziele zu erreichen. **„Learning as we go“.** Ein solches Vorgehen fällt vielen analytisch denkenden Menschen manchmal schwer.

Installation und Test des Plugin Skeletts

- Melde dich per ssh auf dem Übungsserver an
- Wechsle in das Verzeichnis `~/public_html/wiki/lib/plugins`
- Klone das Repo der Projektvorlage: `git clone https://codeberg.org/ironiemix/dokuwiki-plugin-projekt.git`
- Benenne das entstehende Plugin Verzeichnis um: `mv dokuwiki-plugin-projekt/ projekt`

Das Verzeichnis eines DokuWiki-Plugins muss immer genau so heißen, wie das Plugin in den php-Dateien auch benannt ist. Da das Vorlagen Plugin dort „projekt“ heißt, muss es sich also in einem Ordner mit den Namen „projekt“ befinden, damit es funktionieren kann.

Füge auf deiner Startseite im Wiki den Code

```
{{projekt>einTest}}
```

ein und betrachte die Ausgabe. Das sollte in etwa so aussehen:

Erklärungen

Das „projekt“ Plugin ist ein sogenanntes „Syntax-Plugin“, es erweitert die Dokuwiki Syntax um eigene Befehle. In unserem einfachen Fall findet lediglich eine Ersetzung statt: Wenn in der Wiki Seite der Code

```
{{projekt>optionen}}
```

gefunden wird, wird dieser durch einen durch das Plugin bestimmten Text ersetzt. Im Quelltext des Plugins wird gesteuert, nach welchen Code-Mustern in der Wiki-Seite gesucht wird, in welcher Weise gefundene Muster verarbeitet werden und was anstelle des gefundenen Musters ausgegeben werden soll.

Dieser Vorgang wird in der Datei `syntax.php` im Plugin Verzeichnis gesteuert, dabei spielen vor allem die drei Methoden `connectTo`, `handle` und `render` eine Rolle, die jedes DokuWiki-Syntax Plugin haben muss. Der prinzipielle Ablauf ist der folgende:

(A1)

Öffne die Datei `syntax.php` im Editor deiner Wahl. Der Quelltext ist mit Kommentaren versehen - vollziehe den im Diagramm oben dargestellten Ablauf beim finden des Musters und dessen Ersetzung im Quelltext nach.

Überprüfe deine Erkenntnisse, indem du das Plugin veränderst:

- Verändere den Code in deiner Wikiseite und versuche den regulären Ausdruck zu verstehen, der das Pattern-Matching macht: Was kannst du verändern, das nicht? Warum?
  - Kannst du das > Zeichen weglassen? Kannst du die Option nach dem > Zeichen weglassen?
  - Kannst du die geschweiften Klammern weglassen? Nur eine Klammer?
  - Kannst du anstelle von `projekt` vor dem> Zeichen etwas anderes schreiben?
  - Ändere den regulären Ausdruck so ab, dass nach dem > auch nichts stehen kann.
- Ändere die Ausgabe so ab, dass anstelle der Liste eine Tabelle ausgegeben wird:

- Ändere das Plugin so ab, dass es auf das Muster `::mondial>.+?::` reagiert. Mach die Änderung anschließend rückgängig.

(A2)

Baue eine Entscheidung in die Ausgabe ein:

```
 {{projekt>tabelle}}
```

soll eine Tabelle mit Befehl und Optionen ausgeben (s.o.)

```
{{projekt>liste}}
```

eine Liste wie zu Beginn.

(A3)

Ein Syntax Plugin kann auch mit mehr als einem Pattern verbunden werden. Auf diese Weise kann man auch mehrere „Ersetzungs-Befehle“ in einem Plugin implementieren, indem man in `connectTo` mehrere Muster suchen lässt:

```
$this->Lexer->addSpecialPattern('\{\{ptabelle>.*?\}\}', $mode,
'plugin_projekt');
$this->Lexer->addSpecialPattern('\{\{pliste>.*?\}\}', $mode,
'plugin_projekt');
```

Übernehme diese beiden Zeilen in deine connectTo-Methode und steuere die Ausgabeart nun über den Befehl anstelle der Option.

Styling der Ausgaben

Um den im Plugin erzeugen und dann im Browser angezeigten HTML-Code noch mit eigenen Styles zu versehen, reicht es, im Plugin Verzeichnis eine Datei `style.css` anzulegen, in denen man CSS-Regeln festlegen kann. Styles in dieser Datei werden von DokuWiki beim Seitenaufruf automatisch mitgeliefert.

(A4)

Erstelle eine Datei `style.css` im Plugin Verzeichnis mit folgendem Inhalt:

```
.projekt-befehl {
    color: red;
}

.projekt-option {
    color: green;
}
```

Passe dann die Ausgabe im Plugin so an, dass der Befehl zwischen

```
<span class="projekt-befehl>...</span>
```

steht und die Option entsprechend und überprüfe das Ergebnis.

Die CSS-Dateien werden von Dokuwiki lange im Cache gehalten. Wenn aktualisierte CSS-Regeln nicht im Browser ankommen, kann man das Löschen des Caches erzwingen, indem man auf dem Übungsserver den Zeitstempel der Datei `../wiki/conf/dokuwiki.php` aktualisiert: `touch ~/public_html/wiki/conf/dokuwiki.php`.

Sichern des bearbeiteten Plugins in einem eigenen Git-Repo

Da die Plugin-Vorlage aus einer Quelle geklont wurde, auf die du keine Schreibrechte besitzt, sorgen wir jetzt noch dafür dass du eine weiteren Entwicklern in einem eigenen Repo weiterverfolgen kannst.

Lege zunächst ein leeres Git-Repo auf Gitea an. Im Beispiel verwende ich `https://gitea.schule.social/sbel/dokuwiki-plugin-frank.git`. Die Adresse zeigt Gitea nach dem Erstellen des Repos in der Kurzanleitung an:

Jetzt wechselt man in das Plugin-Verzeichnis auf dem Übungsserver und setzt dort eine neue Adresse für den „origin“ des Repos:

```
git remote set-url origin
https://gitea.schule.social/sbel/dokuwiki-plugin-frank.git
```

Von nun an kann man das Repository auf den eigenen Origin pushen - du hast einen Fork erstellt.

<html> <script id=„asciicast-5AXcMQhGvS1VZv5LPZuVDlJUZ“ src=„https://asciinema.org/a/5AXcMQhGvS1VZv5LPZuVDlJUZ.js“ async></script> </html>