UPGRADE VON OJS 2-PLUGINS NACH OJS 3

OJS-de.net

Mit der Veröffentlichung von OJS 3 wurden grundlegende Änderungen in der Konzeption und der Struktur der Software eingeführt. Diese haben Auswirkungen auf die Kompatibilität von bestehenden und die Entwicklung von neuen Plugins. Diese Handreichung dokumentiert wesentliche Änderungen und deren Berücksichtigung bei der Anpassung und Neuentwicklung von Plugins für OJS 3.

Inhalt

Einleitung

OJS 3 unterscheidet sich grundlegend von seinem Vorgänger OJS 2. So wurden, unter anderem basierend auf Vorschlägen aus der OJS-Community, eine Reihe von Verbesserungen sowie neue Funktionen eingeführt. Wichtige neue Funktionen sind zum Beispiel:

New Features in OJS 3

- Redaktionelle Diskussionen
- Flexibler Workflow
- Flexible Rollen/Rechte
- Flexibler Zugang zu rollenspezifischen Funktionen

- Unbegrenzte Anzahl von Einreichungsdateien
- Anpassbare Benutzerschnittstelle
- Verbesserte Designvorlagen
- Responsives Design
- Vereinfachte Registrierung

Darüber hinaus wurden die Codebasen von OJS und OMP (Open Monograph Press) zusammengeführt und die Organisationstruktur von Dateien und Ordnern sowie die Klassenhierarchie teilweise geändert. Diese Änderungen haben einen wesentlichen Einfluss auf die Anpassung und die Neuentwicklung von Plugins.

Einfache Plugins können gegebenenfalls noch ohne oder mit kleineren Änderungen übernommen werden, die meisten Plugins müssen aber an die neuen Strukturen angepasst werden.

Mit OJS 3 wurde außerdem die „*Plugin-Galerie*“ eingeführt, über die Plugins, nach Validierung durch PKP, direkt zur Installation bereitgestellt werden können.

Der Aufbau von OJS-Plugins

Plugins werden in OJS in Kategorien eingeteilt, die ähnliche Funktionen erfüllen und über gemeinsame Elemente verfügen. Jede Plugin-Kategorie hat eine zugehörige Basisklasse, welche wiederum von der Plugin-Klasse abgeleitet wird. Die Plugin-Kategorien von OJS sind:

- Metadaten-Plugins
- Plugins für Zugangsberechtigungen
- Block-Plugins
- Gateways
- Generische Plugins
- Import-/Export-Plugins
- Plugins für OAI-Metadaten-Formate
- Plugins für Zahlungsweisen
- Plugins für öffentliche Kennungen
- Bericht-Plugins
- Designvorlagen-Plugins

Die Plugins befinden sich im Ordner „*ojs/plugins*“, wo jeder Plugin-Kategorie ein Unterordner zugewiesen ist.

Alle OJS-Plugins haben einen ähnlichen Grundaufbau und eine vergleichbare Datei- und Verzeichnisstruktur. Die Dateien und Verzeichnisse für ein typisches, einfaches OJS-Plugin mit

Benutzerschnittstelle, wie z.B. das „*Developed By*“-Plugin, sind in Abbildung 1 dargestellt.

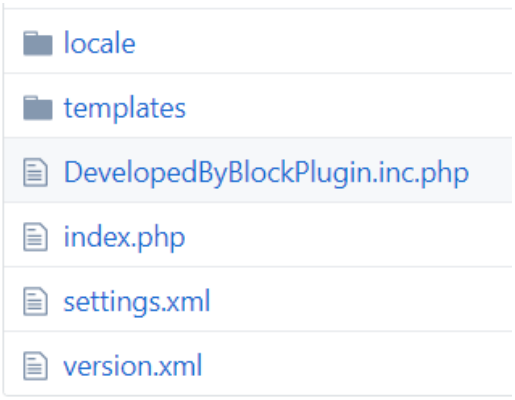

Abbildung 1 Typische Verzeichnis- und Dateistruktur eines Plugins

Minimal notwendig für ein Plugin ohne Benutzerschnittstelle (wie z.B. das Acron-Plugin) sind nur die Dateien version.xml, index.php und [PluginName]Plugin.inc.php sowie der Ordner locale. Je nach Funktionalität eines Plugins können weitere Dateien und Ordner benötigt werden. Tabelle 1 zeigt eine Auflistung weiterer, in Plugins häufig vorkommender Dateien und Ordner inklusive ihrer Funktionen.

Tabelle 1 Dateien und Ordner welche häufig in einem OJS-Plugins vorkommen

Datei/Ordner	Funktion
index.php	erstellt das Plugin-Objekt
[PluginName]Plugin.inc.php	Hauptklasse des Plugins, aus ihr wird das Plugin-Objekt erstellt
version.xml	liefert Meta-Informationen zum Plugin (z.B. Name, Plugin-Kategorie, Versionsnummer, Veröffentlichungsdatum, …)
locale	stellt Sprachvarianten bereit
settings.xml	Standard-Plugin-Einstellungen
[PluginName]SettingsForm .inc.php	stellt das Formular für die Plugin-Einstellungen zur Verfügung

Datei/Ordner	Funktion
[PluginName]Handler.inc.php	bearbeitet die Seitenaufrufe des Plugins
schema.xml	beschreibt die Datenbank-Tabellen, die für das Plugin angelegt werden
composer.json	Konfiguration der externen Plugin-Abhängigkeiten. Wird von composer verwendet (nicht verfügbar in OJS 2)
js	Javascript-Code
templates	neue Designvorlagen, OJS-Designvorlagen die vom Plugin überschrieben werden
classes	zusätzliche Klassen, z.B. Datenbankschnittstelle, pluginspezifische Klassen
controllers/grid	Klassen für die Erstellung von Grids
filter	spezifische Datenobjektfilter
schema	Metadatenschema
images	Bilder
css	pluginspezifische Stylesheets

Anpassungen spezifischer Dateien

Die Basisdateien version.xml und [PluginName]Plugin.inc.php

Passen Sie in der Datei „*versions.xml*“ insbesondere die Versionsnummer und das Veröffentlichungsdatum an.

Falls Ihre Plugin-Klasse einen Konstruktor verwendet, passen Sie in der Datei „*[PluginName]Plugin.inc.php*“ den Namen des Konstruktors wie im Abschnitt „Klassenkonstruktoren“ beschrieben an.

Verwenden Sie in Ihrer Klasse Management-Aktionen, so müssen Sie in der Datei „*[PluginName]Plugin.inc.php*“ die

Methode „*getManagementVerbs()*“ Ihres alten OJS 2-Plugins durch die neue Methode „*getActions()*“ ersetzen.

Beispiele hierfür finden Sie in den in Tabelle 2 aufgeführten Plugins.

Tabelle 2 Plugins mit Management-Aktionen

Plugin	OJS-Version	
UsageStats	2.4.8	3.1.2
Shariff	2.4.8	3.x

Der locale-Ordner

Der locale-Ordner enthält die Sprachversionen Ihres Plugins. Prüfen Sie, ob alle für die neue OJS-Version nötigen Übersetzungen vorhanden sind.

Der css-Ordner

Falls Sie in Ihrem Plugin Stylesheets verwenden, müssen Sie diese gegebenenfalls an das neue Default-Design von OJS 3 oder das verwendete Theme-Plugin anpassen.

Der classes-Ordner

Der classes-Ordner enthält zusätzliche Klassendefinitionen, welche von Ihrem Plugin benötigt werden.

Im Rahmen der Zusammenführung der Codebasen von OMP und OJS wird die Variable „*$journalId*“ in „*$contextId*“ umbenannt. Da der Prozess der Umbenennung noch nicht vollständig abgeschlossen ist, sollten Sie bei Neuentwicklungen oder Anpassungen von Plugins alle Klassen auf die neue Namenskonvention überprüfen.

Falls Sie Data-Access-Objekt-Klassen verwenden (Datenbankschnittstelle, DAO-Dateien), überprüfen Sie die Schreibweise der Spaltennamen in Ihren SQL-Anweisungen. Auch hier werden gemäß der neuen Namenskonvention die Referenzen auf „*journal_id*“ durch „*context_id*“ ersetzt.

Falls Sie in Ihrem Plugin von der Klasse Form abgeleitete Klassen verwenden, z.B. [PluginName]SettingsForm.inc.php, so müssen Sie die in OJS 2 verwendete Methode „*display()*“ durch die Methode „*fetch()*“ ersetzen. Beispiele für diese Anpassung finden Sie in den in Tabelle 3 aufgelisteten Plugins.

Tabelle 3 Beispiele für die Anpassung von Form-basierten Klassen.

Plugin	OJS-Version	
UsageStats	2.4.8	3.1.2
Shariff	2.4.8	3.x

Der templates-Ordner

Der templates-Ordner enthält alle Designvorlagen Ihres Plugins. Falls Ihr Plugin OJS-Templates überschreibt, überprüfen Sie, ob die Original-Designvorlagen verändert wurden und passen Sie Ihre abgeleiteten Designvorlagen entsprechend an.

Falls Sie in Ihrem Plugin die [PluginName]SettingsForm-Klasse verwenden, müssen Sie im templates-Ordner die zugehörige Datei [PluginName]SettingsForm.tpl anpassen.

In OJS 2 wurden die Kopf- und Fußzeilen der Plugin-Einstellungen direkt in die [PluginName]SettingsForm.tpl-Datei eingebunden (siehe z.B. UsuageStatsSettingsForm.tpl [OJS 2.4.8]). In OJS 3 werden die Kopf- und Fußzeilen stattdessen automatisch eingebunden (vergleiche UsageStatsSettingsForm.tpl [OJS 3.1.2]).

Allgemeine Anpassungen

Klassenkonstruktoren

Migration nach PHP 7

Da OJS 3.1.2 PHP in der Version 7 verwendet, müssen alle Plugins gemäß der Anforderungen von PHP 7 angepasst werden.

Eine wesentliche Anpassung betrifft dabei die Namenskonvention für PHP-Klassenkonstruktoren. Während bis PHP 5.6 Konstruktoren den Namen der Klassen haben durften (Konstruktoren vom Typ PHP 4, Codesegment 1),

erwartet PHP 7 den Namen „*__contruct*“ für alle Konstruktoren (Codesegment 2). Falls keine Funktion mit dem Namen „*__construct*“ gefunden wird, wird eine „*E_DEPRECATED*“-Warnung ausgegeben.

```php
<?php
	class foo {
		function foo() {
			echo 'I am the constructor';
		}
	}
?>
```

Codesegment 1 Konstruktor vom Typ PHP 4

```php
<?php
	class foo {
		function __construct() {
			echo 'I am the constructor';
		}
	}
?>
```

Codesegment 2 Konstruktor vom Typ PHP 7

Templates und Stylesheets

Eine wesentliche Änderung in der Struktur von OJS, welche für die Entwicklung von Plugins von Bedeutung ist, betrifft die Trennung von Ressourcen zur Anpassung der Benutzeroberflächen. Während in OJS 2 Stylesheets und Templates für Frontend und Backend gemeinsam verwaltet wurden, sind diese in OJS 3 getrennt. Templates für das Frontend sind nun im Ordner „*../templates/frontend*“ zu finden.

Grids als Designelemente in OJS 3

Obwohl Grids als Designelemente in OJS 2 bereits zur Verfügung standen (vergleiche Klassenreferenzen im Kasten rechts), fanden sie kaum Verwendung. Ihr Funktionsumfang wurde in OJS 3 stark erweitert, sodass sie nun ein wesentlicher Bestandteil der neuen Benutzeroberfläche in OJS 3 geworden sind.

Klassenreferenz GridBodyElement in OJS 2

Klassenreferenz GirdBodyElement in OJS 3

Bei der Anpassung Ihres Plugins an OJS 3 sollten Sie überprüfen, wo Sie die Funktionalität und Benutzerfreundlichkeit Ihres Plugins durch die Verwendung von Grid-Elementen verbessern können.

Als Beispiel vergleichen wir die Plugin-Versionen des Static-Pages-Plugins.

In OJS 2 wurde die Form-Klasse verwendet, um die Benutzerschnittstelle zu gestalten (Abbildung 2). Der zugehörige Code ist auf Github zu finden (https://github.com/pkp/ojs/tree/ojs-stable-2_4_8/plugins/generic/staticPages). Die Klasse StaticPagesEditForm wird dort von der Form-Klasse abgeleitet.

Abbildung 2 Benutzerschnittstelle des Static-Pages-Plugins für OJS 2

In OJS 3 verwendet das Static-Pages-Plugin einen GridHandler, welcher in der Methode „*setupGridHandler*“ der Klasse StaticPagesPlugin initialisiert wird (siehe Code auf Github: https://github.com/pkp/staticPages). Abbildung 3 zeigt die resultierende Benutzerschnittstelle.

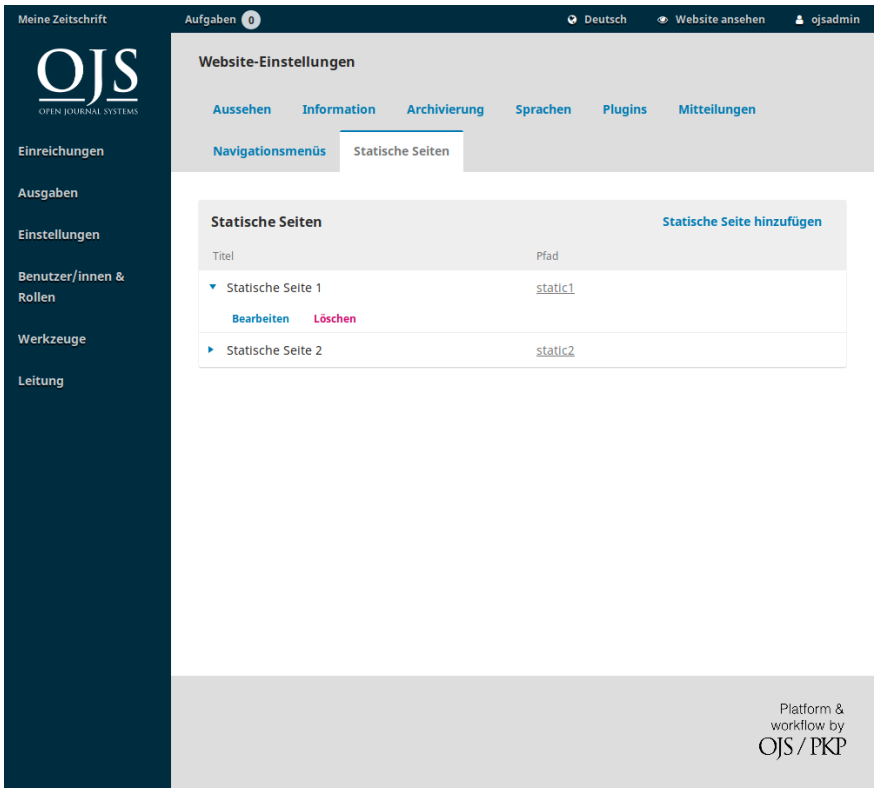

Abbildung 3 Benutzerschnittstelle des Static-Pages-Plugins für OJS 3