Miplex2 Dokumenatation

Martin Grund
grund@miplex.de

22.09.04

Inhaltsverzeichnis

Einführung

Anfangs sei kurz erläutert, worum es bei Miplex2 eigentlich geht und was diese Dokumenatation leisten kann und was nicht.

Miplex2 ist ein Content Management System (CMS). Es ist dafür da, Seiten einer Webseite auszuliefern und diese für den Administrator der Seite möglichst einfach wartbar zu sein. Dabei erhebt Miplex2 keinerlei Anspruch auf Vollständigkeit, sowie Fehlerfreiheit.

Doch was kann Miplex2 eigentlich?

Miplex2 verwaltet die Inhalte der Webseite nicht etwa in einer Datenbank, sondern in einer XML Datei. Durch diesen Vorteil, ist Miplex2 völlig unabhängig von den unterliegenden Strukturen. Ja ok, PHP in der Version größer 4.2 sollte schon installiert sein, aber auch hier werden keine Erweiterungen benutzt, die nicht unbedingt vorhanden sein müssen. Durch die Verwaltung der Daten innerhalb einer XML Datei, können diese auch offline – sprich nicht direkt im Administrationsbereich von Miplex2 – editiert werden. Um Miplex2 nun zu aktualisieren muss nur die gewünschte Inhaltsdatei mit der neuen Version überschrieben werden.

Was kann Miplex2 nicht?

So viel ist das eigentlich nicht. Miplex2 ist nicht dafür gedacht, mehrere Domains innerhalb einer Miplex2 Struktur zu verwalten. Leider ist auch nocht nicht vorherzusagen, wie sich Miplex2 verhält, wenn mehr als xy Benutzer gleichzeitig darauf zugreifen. Und im übrigen befindet sich Miplex2 noch in der Entwicklung.

Wozu ist dann diese Doku hier gut?

Diese Dokumenatation soll als Beschreibung der Sachverhalte dienen, wie Miplex2 eigentlich funktioniert. Des weiteren, ist es eine kleine Referenz der großen Miplex2 Klassen. Und es soll auch ein Tutorial bieten, wie man Miplex2 mit seinen eigenen Erweiterungen besser machen kann.

Einführung für Programmierer

Seitendarstellung von A bis Z

In diesem Kapitel geht es primär darum zu verstehen, wie der gesamte Ablauf vom Request der Seite durch den Browser bis zur endgültigen Darstellungen vorn Statten geht.

Beginnen wir mit dem Start eines jeden Durchlaufes in der Datei `index.php`. Damit alles gut läuft, wird Output Buffering angeschaltet. Als nächstes wird ein neues Objekt vom Typ `Session` erzeugt. Als Parameter wird der Pfad zur Konfigruationsdatei übergeben. Diese Datei ist ein serialisiertes Objekt vom Typ `MiplexConfig`. Damit nun auch Inhalt ausgegeben werden kann, wird mittels der Methode `getActPage()` die aktuelle Seite bestimmt. Diese wird nun an das Template Objekt übergeben und danach wird die Seite ausgegeben.

Dies ist der Ablauf zur Generiereung der Seite so kurz wie möglich beschrieben, doch was passiert nun in den Einzelheiten?

Schauen wir uns also an, was passiert, wenn ein neues Session Objekt erzeugt wird. Dieses Session Objekt ist relevant für alles was bei Miplex2 passiert. Es ist überall vorhanden und speichert alle Daten, die von dem System benötigt werden und stellt diese als einheitliches Interface dem Programmierer zur Verfügung.

Wenn wir uns nun den Konstruktor anschauen sehen wir (`Session.class.php`, `Session()`), das zusätzlich zum ersten Parameter noch ein zweiter existiert, der regelt, ob die Darstellung im Frontend oder im Backend oder sonstwo erfolgt. Nun werden mehrere Methoden aufgerufen, die ein Mindestmaß an Funktionalität der Klasse zur Verfügung stellen. Diese Methoden sind `setConfig()`, `setSite()` und `getSmartyObject()`. Die Methode `setConfig()` dient dazu, die serialisierte Konfiguration wieder in ein Objekt zu verwandeln und in der Session Klasse zu registrieren. Nun haben wir also ein Konfigruation erzeugt und wissen schon ein wenig mehr bescheid. Nun geht es also daran zu erfahren, was eigentlich in unserem Datenspeicher, sprich unserer XML Datei enthalten ist. Dazu dient die Methode `setSite()`. Innerhalb dieser Methode wird eine neues Objekt vom Typ MiplexDatabase erzeugt, dass uns all die Dinge abnimmt, die direkte Manipulationen an der XML Datei betreffen. Als Parameter wird dem Konstruktor von MiplexDatabase einmal die Konfigruation von Miplex2 übergeben und andererseits ein Wert, der bestimmt, ob der Inhalt der Content Elemente in der Struktur gespeichert werden sollen, oder ob diese dann manuell geladen werden sollen. Dieser Parameter ist zur Zeit noch händisch direkt auf 1 festgelegt, was bedeutet, dass der Inhalt in der Struktur gespeichert wird und damit weiger Zugriffe auf die XML Datei anfallen. Der Nachteil davon ist, dass nun mehr Inhalt in dem Objekt gespeichert werden muss. Sollte es Performance Einbußen geben, kann man an dieser Stelle nachschrauben, aber zu Zeit kann aus der Seite der Inhalt noch nicht ausgelesen werden, also am besten merken und ignorieren. Das erzeugte Objekt wird nun in der Session gespeichert und als nächste wird die Seitenstruktur mittels der Methode getSiteStructure() ausgelesen und in der Session verankert. Damit kommen wir wieder zurück in den Konstruktor des Session Objektes und zu der Methode getSmartyObject(). Diese hat nicht mehr Sinn, als ein Objekt der Template Engine zu erstellen, das zu endgültigen Darstellung der Seite dient. Um mehrsprachigkeit zu ermöglichen wird nun

noch ein letztes Objekt in der Session registriert und zwar der M2Translator. Diese Klass dient dazu einen String in die gewünschte Landesprache zu übersetzen. Als Parameter wird dem Konstruktor nur das Landeskürzel der gewünschten Sprache übergeben. In Zukunft soll dies auch im Backend in der Konfigruation einstellbar sein.

Nun haben wir also das Session Objekt erzeugt und kommen damit zurück in unsere index.php Datei. Damit wir aber nun Content aus der Seite auslesen können rufen wir eine bestimmte Methode der Session auf: getActPage().

Damit gehen wir also wieder in unser Session Objekt. Hier sehen wir wieder einen versteckten Parameter. Mittels diesem Parameter könnten wir steuern, welchen Inhalt wir wirklich ausgeben wollen, den der gefordert ist, oder den, den wir übergeben haben. Ist der Parameter wie in unserem Fall nun null, wird aus dem Array der Servervariablen die REQUEST_URI ausgelesen und verarbeitet. Damit wir sinnvoll arbeiten können, entfernen wir also zunächst das hinten anstehende ".html", falls noch Parameter der Seite übermittelt wurden, löschen wir diese aus dem String und behalten diese aber bei uns um sie später der Seite zu übergeben. Nun haben wir also einen String der Form: "/miplex2/index.php/seite1/seite2". als erstes entfernen wir das Stammverzeichnis und das index.php bleibt also nur noch übrig "seite1/seite2" Daraus erstellen wir ein Array und rufen mit diesem als Parameter die Funktion getRequestedPage() auf. Im übrigen wird die Zeichnefolge "seite/seit2..." immer als Pfad einer Seite bezeichnet. Dieser Begriff wird später, wenn man den Quelltext durchsucht immer wieder auftauchen. Dieser Pfad liefert eindeutig einen Knoten innerhalb der XML Datei zurück.

Um nun die Funktion getRequestedPage() zu verstehen, bedarf es einer kurzen Erläuterung der internen Datenspeichererung der gesamten Seite. Dies wird innerhalb des Objekts als Array gespeichert und innerhalb des Array liegen wiederum Objekte vom Typ PageObject. Gibt es für eine Seite wieder darunter liegende Seiten, wird die Variable "hasChildPage" der Seite auf 1 gesetzt. Innerhalb der "subs" Variable finden sich nun wieder in einem Array die darunter liefegenden Seiten. Somit lässt sich also aus einer Seite immer das darunter liegede Strukturbild aufbauen.

Doch nun zurück zu unserer Funktion. Wie haben getRequestedPage ein Array übergeben, dass zusammengesetz den Pfad zu unserer gewünschten Seite ergibt. Wie gesagt muss jeder Seitenalias innerhalb seines Zweiges eindeutig sein, dass bedeutet, dass keine Schwesterknoten denselben Alias haben darf. Nun wird das Array umgedreht um es wie einen Stack zu behandeln und von oben herab wird überprüft, ob es für das gesuchte Element ein passendes Gegenstück gibt, ist dies der Fall, wir diese Seite als zu untersuchende Seite gespeichert und die Untersuchung beginnt mit dem nächsten Element aus unserem Array. Wenn das Array abgearbeitet ist und kein Fehler aufgetreten ist, dann sind wir fertig, ist jedoch ein Fehler eingetreten, so bricht die Methode ab und liefert false zurück, sonst das gewünschte Page Objekt.

Damit kommen wir als zurück zur Methode getActPage(), nun wird das Egebnis aus getRequestedPage() in der Klassenvariable currentPage gespeichert und an das aufrufende Programm zurückgegeben.

Für den Fall, dass das Array der zu bestimmenden Seite lehr war, wird automatisch das erste Element aus dem PageObject Array ausgegeben, was der ersten Seite entspricht. Dieser Fall tritt genau dann auf, falls die Startseite angewählt wurde.

Das an die index.php gelieferte PageObject wird nun an das Template übergeben und eigentlich wäre der Fall nun erledigt, wäre da nicht noch ein interessanter Aufruf innerhalb des Templates.

Es stellt sich nämlich die Frage, wie der Inhalt nun tatsächlich auf die Seite kommt, da wir ja nur ein Objekt einer Klasse übergeben haben und keinen String, der direkt ausgegeben wird. An dieser Stelle kann man nur sagen, ein Glück dass wir Smarty haben, denn Smarty kann auch problemlos mit Objekten umgehen.

Erweiterungen verstehen und programmieren

Einführung für Redakteure

Referenz

Session.class.php

Autor: Martin Grund <grund@miplex.de>

Datum: 22.9.2004

Die Session Klasse organisiert und verwaltet alle Arbeiten während einer Session des Benutzers. In der Session sind Referenzen auf alle anderen benötigten Objekte hinterlegt, dass sie von anderen Klassen referenziert werden können, solange das Session Objekt bekannt ist.

Klassenvariablen

- $site - die gesamte Struktur der Seite
- $config - die Konfiguration von Miplex2
- $currentPage - ein Verweis auf den aktuellen Eintrag der Seite
- $mdb - Das MiplexDatabase Objekt
- $i18n - das Lokalisierungs Objekt
- $user - der aktuelle Username, Standard ist „nobody“
- $smarty - das Objetk der Template Engine
- $lang - die aktuelle Sprache des Backends
- $type - Typ der Sessin (Frontend oder Backend)

Methoden

Session()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$configFile	String	Ja	N/a	Der Pfad zur Konfigurationsdatei
$type	String	Nein	„frontend“	Der Typ des Backends

Dies ist der Konstruktor der Klasse. Hier werden alle nötigen Variablen initialisiert um eine korrekte Session zu erzeugen.

setSite()

Ein neues Objekt der MiplexDatabase wird erzeugt und in der Session registriert. Danach wird die gesamte Struktur der Seite ausgelesen.

setConfig()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$configFile	String	Ja	n/a	Der Pfad zur Konfigurationsdatei

Die Konfiguration von Miplex2 wird in dieser Funktion geladen. Es wird die angegebene Datei ausgelesen und deserialisiert. Aus diesem Datenstrom wird wieder ein Objekt vom Typ MiplexConfig erstellt.

getSmartyObject()

Hier wird ein Objekt der Template Engine erzeugt. Mit diesem Objekt wird die komplette Ausgabe von Miplex2 gesteuert. Die Konfiguration des Objektes wird durch verschiedene Variablen in der Konfigurationsdatei gesteuert.

getActPage()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$requestUri	String	ja	n/a	Die aktuell angeforderte Seite.

An diese Methode wird eine Url übergeben, die aktuell angefordert ist. Nun wird aus dieser Url der Pfad extrahiert, der die aktuelle Seite beschreibt. Zurückgegeben wird dann ein PageObject der aktuellen Seite.

return – PageObject

getRequestedPage()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$tmpUri	Array	ja	n/a	Ein Array mit den Einträgen in der reihenfolge des aktuellen Pfades

Der Funktion wird ein Array übergeben, dass die Einträge enthält, die der Reihenfolge des gewünschten Pfades entsprechen. Dieses Array wird nun verarbeitet um die korrekte Seite innerhalb der Struktur zu finden. Dabei wird ausgenutzt, dass zu jedem Eintrag immer nur ein oder kein passendes Gegenstück existieren muss. Ist die Session ein Frontend Objekt, dann werden die Shortcuts verfolgt, sonst nicht. Im Fehlerfall (z.B. die Seite existiert nicht) wird false zurückgegeben, sonst die gewünschte Seite als PageObject.

return – false | PageObject

getArrayId()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$pageObjects	Array	ja	n/a	Ein Array aus PageObjects
$alias	String	ja	n/a	Der Alias zu der gewünschten Seite

Wird diese Funktion aufgerufen, dann wird ein Array übergeben und ein Alias. Nun wird das Array aus PageObjects nach dem Alias durchsucht. Wird dieser gefunden, gibt die Funktion einen Integer Wert größer als 0 zurück. Im Fehlerfall wird -1 zurückgegeben.

saveAndResetSite()

Mittels dieser Funktion wird die aktuelle Seite gespeichert und das XPath Objekt zurückgesetzt, die Struktur und die Konfiguration neu eingelesen.

MiplexDatabase

Autor: Martin Grund <grund@miplex.de>

Datum: 22.9.2004

Diese Klasse dient dazu sämtliche Zugriffe auf die XML Datei abzufangen und für die Benutzer in eine einfacherere Form zu bringen. So kann die gesamte Datei ausgelesen werden, aber auch nur einzelne Teile oder es können Änderungen an der XML Datei vorgenommen werden.

Klassenvariablen

- $miplexConfiguration – eine Referenz auf die aktuelle Konfiguration
- $xmlFileName – Der Dateiname der Inhaltsdatei
- $xPathHandle – Handle auf das XPath Objekt
- $storeContentInStructure – Soll der Content in der Struktur gespeichert werden
- $error – Letzte Fehlermeldung
- $site – Die Seitenstruktur

Methoden

MiplexDatabase()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$config	MiplexConfig	ja	n/a	Referenz auf die Konfigruation
$storeContentInStructure	Integer	nein	0	Soll der Inhalt in der Struktur gespeichert werden.

Der Konstruktor der Klasse. Sie dient dazu alle vorbereitenden Tätigkeiten auszuführen, damit ein problemloser Zugriff auf die XML Datei erfolgt.

reset()

Wenn die XML Datei gespeichert wird, muss diese neu ausgelesen werden, damit die Änderungen auch an die $site Variable übergeben werden, damit die Session Klasse diese auslesen kann. Wurde die XML Datei ausgelesen, dann wird die Struktur der Seite auch komplett neu bestimmt.

getSiteStructure()

Diese Funktion ließt die XML Datei aus und erzeugt aus den Eingaben ein Array aus PageObjects. Diese wird nach Abschluss der Methode zurückgegeben. Das Array hat die Struktur der Seite.

return – Array PageObject

getSectionRecursive()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$path	String	ja	n/a	Der zum Kontext gehörige Pfad.

Diese Funktionn ruft sich selbst rekursriv auf, um alle Kind-Seiten eines bestimmten Ausgangspunktes zu finden. Dabei wird als Parameter der Ausgangskontext übergeben und der dazugehörige Pfad.

getSection()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$debug	Integer	nein	0	Debugging

Gibt zu einem bestimmten Punkt innerhalb der XML Datei die passende Section aus. Falls $storeContentInStructure auf 1 gesetzt ist, wird auch der Inhalt ausgelesen, falls 0, dann nicht.

getContentAtPosition()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes

Diese Funktion ermittelt zu dem übergebenen Parameter $context der Fomr /section[1]/ content[1] den Inhalt des Contentbereichs ohne die CDATA Tags.

stripCdata()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$text	String	ja	n/a	Text mit CDATA Tags

Diese Funktion entfernt von beliebigen Text die CDATA Tags, falls diese vorhanden sind.

replaceContent()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$content	String	ja	n/a	Der neue Text

Diese Funktion erstezt Text, der durch $context eindeutig bestimmt ist durch einen neun Text. Dabei wird um den neuen Text noch eine CDATA Section gelegt, damit der Inhalt nicht vom Parser erfasst wird.

return – Context | False

addContent()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$position	Integer	ja	n/a	Die neue Position des CE
$content	String	ja	n/a	Der Inalt des CE
$attribute	Array	ja	n/a	Ein Array aus Attributen des CE
$node	Object	nein	null	Wenn ein echter Node direkt aus dem Baum übergeben wird, braucht der angegebene node nicht mehr vorbereitet zu werden.

Diese Funktion fügt inhalt zu einer Sektion hinzu. Dabei müssen verschiedene Fäller unterschieden werden. Es ist noch keine Inhaltselement vorhanden, das Inhaltselement soll als neues erstes Element eingefügt werden und das Element soll an einer beliebigen anderen Position eingefügt werden. Sind schon andere Elemente enthalten werden die neuen Elemente jeweils hinter dem bestehenden einfgefügt

return – Context | False

addSection()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$type	String	ja	n/a	Inner / After
$attributes	Array	ja	n/a	Attribute der Sektion

Mittels diese Funktion wird eine neue Section eingefügt. Entweder direkt nach dem angegebenen Kontext oder innerhalb des Kontextes.

return – Context | False

getContentAttributes()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes

Liefert die Attributes eines Content Elements zurück.

return – Array | False

getSectionAttributes()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes

Liefert die Attribute einer Section zurück.

return – Array | False

editSectionAttributes()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$attributes	Array	ja	n/a	Die Attribute der Section

Funktion zum Editieren der Attribute der Section, vorhandene Attribute werden gelöscht und durch die neuen Attribute überschrieben. In dem Array der Attribute müssen alle Attribute die die Section bestimmen enthalten sein

return - Boolean

editContentAttributes()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$attributes	Array	ja	n/a	Die Attribute der Section

Funktion zum Editieren der Attribute des CE, vorhandene Attribute werden gelöscht und durch die neuen Attribute überschrieben. In dem Array der Attribute müssen alle Attribute die das CE bestimmen enthalten sein

return - Boolean

saveXML()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$beautify	Integer	nein	1	Soll die XML Ausgabe neu formatiert werden.

Funktion zum Abspeichern der im Speicher liegenden XML Struktur.

return - Boolean

saveXMLAndReloadSiteStructure()

Die XML Struktur wird abgespeichert und die Seitenstruktur wird neu geladen und in der Seite verankert.

cdataSection()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$string	String	ja	n/a	Der Text, um den die CDATA Tags gelegt werden sollen.

Diese Funktion legt um den übergebenen String CDATA Tags.

return - String

getContextFromPath()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$path	String	ja	n/a	Der gewünschte Pfad

Diese Funktion ermittelt anhand des übergebenen Pfades den eindeutigen Context innerhalb der XML Datei.

return - String

editContentElement()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$content	String	ja	n/a	Der Inhalt des CE
$attributes	Array	ja	n/a	Attribute des CE

Diese Funktion editiert ein CE in dem es den Text ersetzt und die Attribute erneuert.

prepareContentNode()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$content	String	ja	n/a	Der Inhalt des CE
$attributes	Array	ja	n/a	Die Attribute des CE

Diese Funktion erzeugt aus den Parametern einen neuen XML String, der den neuen Content Knoten beschreibt.

return - String

prepareSectionNode()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$attributes	Array	ja	n/a	Die Attribute der Section

Diese Funktion erzeugt aus den Parametern einen neuen XML String, der den neuen Section Knoten beschreibt.

return - String

removeChild()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes

Diese Funktion löscht einen Kindsknoten, der eindeutig durch den Kontext bestimmt ist.

return - Boolean

moveContent()

Name	*Type*	*Erforderlich*	*Standard*	*Beschreibung*
$context	String	ja	n/a	Der Kontext des Ausgangspunktes
$direction	Integer	ja	n/a	Die Richtung der Bewegung

Diese Funktion bewegt einen Bereich der XML File in eine bestimmte Richtung. Dabei wird sich die Tatsache zu nutze gemacht, dass eine Verschiebung in Richtung -1 identisch zu einer Verschiebung des darunter liegenden Knoten in Richtung + 1 ist. Die Funktion testet alle Sonderfälle ab und liefert einen Wahrheitswert zurück.

return - Boolean