Internationalisierung in PHP-Projekten

Matthieu-P. Schapranow, Christian Schubert, André Wendt
{matthieu.schapranow|christian.schubert|andre.wendt}@hpi.uni-potsdam.de

23. Dezember 2005

Eine Ausarbeitung zur Veranstaltung

„Konzepte und Methoden der Webprogrammierung“

Prof. Dr. sc. nat. Christoph Meinel und Dipl.-Informatiker Mathias Kutzner

im Wintersemester 2005/2006 am

Hasso-Plattner-Institut für Softwaresystemtechnik gGmbH

Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam / Deutschland

Telefon: +49 331 55 09 - 0
Telefax: +49 331 55 09 - 129

http://www.hpi.uni-potsdam.de

Kurzfassung

Die vorliegende Ausarbeitung stellt die Konzeption zur Erstellung eines Übersetzungs-Moduls für Web-Anwendungen in der Programmiersprache PHP 5 dar.
Die Referenz-Implementierung wird für den Einsatz im Kontext des Tele-Teaching-Portals `http://www.tele-task.de` erstellt und ist an die spezifischen Bedürfnisse angepasst. Dabei verwendet der Entwurf das Konzept zweier voneinander getrennter Wörterbücher, auf die mittels ADOdb datenbankunabhängig zugegriffen werden soll. Das *System-Wörterbuch* dient der Speicherung von kurzen obligatorischen Übersetzungen gruppiert nach Namensräumen. Hingegen stellt das *Dokumenten-Wörterbuch* alle mehrsprachigen Inhalte abstrahierend auf Dokumentenebene dar und enthält jeweils mindestens das Dokument in der Ursprungssprache sowie eine variierende Anzahl von Übersetzungen des Ausgangsdokuments.

Schlüsselwörter Internationalisierung, Übersetzer-Modul, ADOdb, PHP 5, System-Wörterbuch, Dokumenten-Wörterbuch, Web-Anwendungen, `tele-task.de`

Abstract

The given paper describes the concept for a database independent translator module for web applications written in PHP 5 using ADOdb. The reference implementation is created for the tele-teaching portal `http://www.tele-task.de` and is optimized for this specific context. The given approach uses two independent dictionaries. On the one hand, the *system dictionary* contains short mandatory translations grouped by unique namespaces. On the other hand, the document dictionary stores all remaining contents abstracted on document level. Thus, it contains each document at least in its source language, but may contain a various number of corresponding document translations.

Keywords Internationalization, Translator module, ADOdb, PHP 5, system dictionary, document dictionary, web applications, `tele-task.de`

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Internationalisierung

Internationalisierung bezeichnet die Anpassung von Produkten, insbesondere Software, für andere Sprachen und Kulturen. In der Softwareentwicklung wird damit die Trennung von Sprachdaten von der Anwendungslogik der Software beschrieben. Dazu werden Zeichenketten und andere länderspezifischen Daten wie z. B. Zahlenformate, die die Software an den Benutzer ausgibt, so aus dem Quellcode ausgelagert, dass der Quellcode für eine derartige Anpassung der Software nicht modifiziert werden muss.

1.1 Gründe für Internationalisierung

Mit der fortschreitenden Globalisierung erhöhen sich die Anforderungen an Softwarehersteller, ihre Produkte möglichst schnell für verschiedene Sprachen und Kulturen anzupassen. Dazu gehört unter anderem die Übersetzung der Menübeschriftungen, Hilfetexte und Meldungen. Wird der auszugebende Text nun vom Programmcode getrennt und in global nutzbaren Ressourcen, sogenannten *Language Packs* abgelegt, beschränken sich die Änderungen auf diese modularen Ressourcen.

Darüber hinaus ermöglicht die Trennung von Sprachdaten und Anwendungslogik, Übersetzungen ohne Programmierkenntnisse vornehmen zu können, so dass die Menge der Übersetzungen nicht auf die Sprachkenntnisse der Programmierer beschränkt bleibt. Die Übersetzung kann damit durch Dolmetscher oder Freiwilligen, anstatt durch teure Softwareentwickler vorgenommen werden, das einen deutlichen Kostenvorteil darstellt.

Die für die Internationalisierung nötige Aufteilung der Software kann des Weiteren helfen, die Software generell modular aufzubauen und somit Erweiterungen mit ihren eigenen Übersetzungen zuzulassen.

1.2 Vergleich von Projekten mit und ohne Internationalisierung

Ein beispielhafter Aufbau von Software ohne Internationalisierung ist in Abbildung 1 zu erkennen. Dabei ist der Programmcode mit Sprachdaten untrennbar vermischt. Die Entwicklung der Software kommt theoretisch mit einer Rolle aus: dem Programmierer. Der Programmcode generiert die Ausgabe, die dem Nutzer angezeigt wird.

Dagegen ist ein beispielhafter Aufbau von Software mit Internationalisierung in Abbildung 2 gegeben. Programmcode und Sprachdaten sind getrennt und erfordern für die Entwicklung der Software mindestens zwei Rollen: Programmierer und Übersetzer. Der Programmcode generiert ebenso wie in Projekten ohne Internationalisierung die Ausgabe für den Benutzer, nur greift er in diesem Fall auf die getrennt abgelegten Ressourcen zu. Dabei spielt ihr Speicherort keine Rolle, sie können sich sowohl auf dem lokalen als auch auf einem entfernten Computer befinden, solange die Software Zugriff darauf erhält. Der Programmierer hat die Aufgabe, den Programmcode zu erstellen und zu modifizieren, während dem Übersetzer voller Zugriff auf die Sprachdaten gewährt wird. Ein Fehler in den Sprachressourcen kann nicht zu einem Softwarefehler führen. Dagegen können Softwarefehler behoben werden, ohne die Sprachdaten modifizieren zu müssen.

Durch den Einsatz von Internationalisierung erhöht sich einerseits die Wartbarkeit und Wiederverwenbarkeit der generierten Software. Andererseits werden versehentliche Änderungen am erstellten Programmcode vermieden, da er für die Übersetzung nicht modifiziert werden muss.

1.3 Verbreitete Probleme der Internationalisierung

Der Einsatz von Internationalisierung bei der Softwareentwicklung kann auch zu Problemen führen. Je mehr Sprachen eine Software unterstützt, desto schwieriger werden Änderungen an den Sprach-

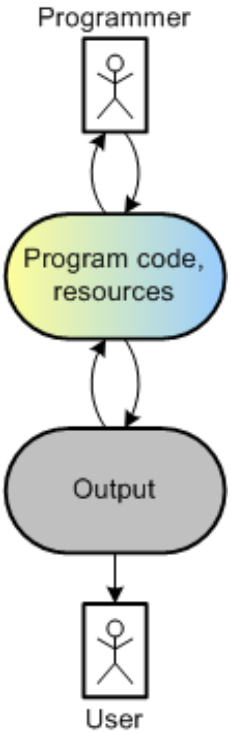

Abbildung 1: Projekte ohne Internationalisierung

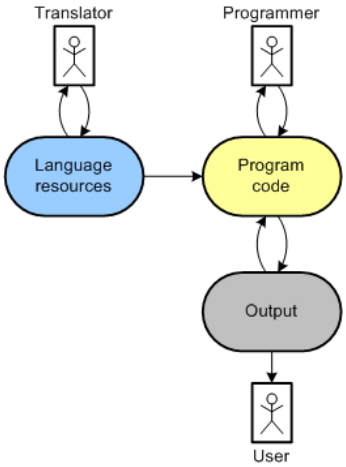

Abbildung 2: Projekte mit Internationalisierung

daten. Wird beispielsweise ein Fehler in der Ursprungssprache entdeckt, müssen möglicherweise alle Übersetzungen entsprechend mit geändert werden. Eine Wartung solcher mehrfacher Ressourcen kann immensen Aufwand verursachen.

Darüber hinaus ist der einwandfreie Einsatz der Software nicht gewährleistet, wenn z.B. kein Zugriff auf die Sprachressourcen möglich ist. Dies kann verschiedene Ursachen haben, z.B. Netzwerkprobleme, wenn die Ressourcen auf einem entfernten Computer abgelegt sind, oder die Daten versehentlich gelöscht wurden. In diesem Fall hätte eine Vermischung von Programmcode und Daten den Vorteil, dass die Software entweder fehlerfrei oder gar nicht mehr laufen würde, weil die Löschung sowohl Sprach- als auch Programmdaten betreffen würde. Solche Probleme können mit entsprechenden Analysen und Anforderungen an die Software im Vorfeld vermieden werden (Qualitätsmanagement).

In jedem Fall gilt es, die Vor- und Nachteile der Internationalisierung abzuwägen und sich für die individuell vorteilhafteste Lösung zu entscheiden.

Im vorliegenden Fall ist die Internationalisierung unbedingt erforderlich, weil zur Entwurfszeit die Menge der im TeleTASK-Projekt verwendeten Sprachen nicht bekannt ist. Das spätere Hinzufügen von Sprachen soll nur noch von Personen ohne Programmierkenntnisse vorgenommen werden (z. B. Dozenten, Übersetzer).

2 Konzepte der Realisierung

Die vorliegende PHP-Anwendung begünstigt den Einsatz zweier Wörterbücher:

- Das System-Wörterbuch beinhaltet die allgemeinen Übersetzungen für die Benutzerschnittstelle sowie Navigation und Hilfe.
- Das Dokumenten-Wörterbuch enthält alle Übersetzungen der so genannten *Content-Objekte*

Alle Übersetzungs-Anfragen erfolgen über eine gemeinsame Schnittstelle, die in Abschnitt 4 im Detail vorgestellt wird.

2.1 System-Wörterbuch

Das System-Wörterbuch wird in logische Bereiche unterteilt, die als *Namensräume* bezeichnet werden. Solche Namensräume werden von den Komponenten des Projektes genutzt, die eine bestimmte Funktionalität kapseln (Module). Beispiele für Module sind Navigation, Suche, Chat usw. Jedes Modul verwendet einen eigenen eindeutigen Namensraum, der sich aus dem ebenso eindeutigen Klassennamen ergeben sollte. Jeder Namensraum enthält wiederum eindeutige textuelle Kürzel (Tokens) für jede enthaltene Übersetzung. Diese Kürzel werden in englischer Sprache verfasst, dienen der Referenzierung im Quellcode und sollen gegenüber numerischen Kürzeln seine Wartbarkeit unterstützen. Ein Token für den Wochentag „Montag“ ist beispielsweise `"monday"`.

Der Namensraum `common` ist reserviert für allgemeine Anfragen, die modulübergreifend sind. Er enthält Übersetzungen von Wochentagen, Monaten und Formatvorlagen für Datum und Uhrzeit.

Um einen fehlerfreien Einsatz der Anwendung zu gewährleisten, muss jedes Modul mindestens **eine vollständige Übersetzung** seiner Sprachdaten bereitstellen, weil die Vollständigkeit späterer Übersetzungen am Umfang der mitgelieferten ersten Übersetzung gemessen wird.

2.2 Dokumenten-Wörterbuch

Inhalte werden als Dokument-Objekte abstrahiert, die über eindeutige Indentifikatoren (ID) referenziert werden. Jedes dieser Objekte kann mehrere Übersetzungen aufweisen. Auf verschiedene Übersetzungen desselben Dokuments wird transparent über die gleiche ID zugegriffen, so dass Anfragen für eine bestimmte Übersetzung muss stets aus Sprache und ID bestehen müssen.

2.3 Wichtige sonstige Vereinbarungen

1. Module, die internationalisierte Zeichenfolgen bereitstellen, müssen über eine Installationsroutine verfügen, die u. a. mitgelieferte Übersetzungen unter Verwendung eines eindeutigen Namensraums im System-Wörterbuch registrieren. Das initiale Einlesen von Übersetzungen in die Datenbank soll durch den Methodenaufruf `registerNamespace()` erfolgen.
2. Sprachen werden über die nach ISO 639-1 festgelegenten zweibuchstabigen Kürzel referenziert.

3 System- und Dokumenten-Wörterbuch

Die gegebenen Anforderungen legten es nah, eine Unterteilung in System- und Dokumenten-Wörterbuch vorzunehmen, da diese sich funktional nicht unerheblich unterscheiden. Dabei wurde die Gelegenheit genutzt, diese Funktionen auf unterschiedlichen Datenstrukturen operieren zu lassen, um für die Entwickler die Benutzung zu vereinfachen. Dazu seien die in Tabelle 1 dargestellten konzeptionellen Unterscheidungen herangezogen, welche im Folgenden ausgeführt werden.

System-Wörterbuch	**Dokumenten-Wörterbuch**
Nicht-Inhalte	Inhalte
statisch im Code referenziert	nicht statisch im Code referenziert
ändert sich zur Laufzeit nicht/selten	ändert sich relativ häufig zur Laufzeit
muss komplett in eine Sprache übersetzt sein, damit diese auswählbar ist	Inhalt wird automatisch in der am besten passenden Sprache angezeigt

Tabelle 1: Wörterbücher: konzeptionelle Unterscheidung

3.1 Inhalt und Erstellung

Wie aus den Bezeichnungen ersichtlich, soll das Dokumenten-Wörterbuch tatsächliche Inhalte enthalten, während das System-Wörterbuch für Nicht-Inhalte zuständig ist. Konkret sind Inhalte diejenigen Daten, welche zur Laufzeit des Systems in Folge von Benutzerinteraktion generiert werden, wohingegen Nicht-Inhalte zumindest teilweise zur Entwicklungszeit erzeugt werden. Teilweise nur deshalb, da es im laufenden Betrieb durchaus möglich sein soll, für bestehende Einträge im System-Wörterbuch Übersetzungen hinzufügen zu können; neue Einträge sollen auf diese Art hingegen nicht entstehen.

3.2 Referenzierung

Die dynamische Natur der Einträge im Dokumenten-Wörterbuch macht es nötig, dass dessen Einträge auch über automatisch generierte Identifikatoren (im Folgenden ID bzw. IDs) angesprochen werden. Im Kontext des Dokumenten-Wörterbuches werden Einträge daher über (numerische) IDs referenziert, welche nicht eine konkrete Übersetzung eines Inhaltes referenzieren, sondern eine Menge davon, wobei diese Inhalte die gleiche Bedeutung in verschiedenen Sprachen repräsentieren sollen. Mit einer konkreten ID kann also ein bestimmter Inhalt referenziert werden, ohne sich dabei auf eine Sprache festlegen zu müssen.

Die Einträge im System-Wörterbuch werden statisch vom Code des jeweiligen Moduls referenziert, so dass deren Anzahl zur Entwicklungszeit bereits bekannt ist. Daher werden an dieser Stelle keine dynamisch generierte IDs verwendet. Stattdessen werden im System-Wörterbuch Tokens verwendet. Ein Token sei hierbei eine relativ kurze Zeichenkette, welche vom Entwickler nahezu beliebig ausgewählt werden kann und nur innerhalb eines zum Modul gehörigen Namensraumes eindeutig sein muss. Dies vereinfacht die Referenzierung im Quellcode, da der Identifikator hiermit vom Entwickler selbst festgelegt werden kann. In der Funktion der Referenzierung eines sprachunabhängigen Textes ist das Token äquivalent zu den IDs des Dokumenten-Wörterbuches.

3.3 Namensräume und der gemeinsame Namensraum `common`

Namensräume sind eindeutige Bezeichner, die zur Gruppierung von Sprachen dienen. Im aktuellen Kontext ist damit insbesondere die Gruppierung von Übersetzungen eines Moduls im des System-Wörterbuch gemeint.

Durch das Namensraum-Konzept wird die Isolation von Sprachdaten verschiedener voneinander unabhängiger Module sichergestellt. Dadurch steht die Pflege und das Einbringen neuer Namenräume mit ähnlichen Übersetzungen miteinander nicht im Konflikt.

Damit trotzdessen verschiedene Module auf identische verwendete Übersetzungen gemeinsam zugreifen können, wurde der gemeinsame Namensraum `common` eingeführt. Dieser ist ein Namensraum wie jeder andere auch, mit dem kleinen Unterschied, dass dessen Übersetzungen implizit zu jedem geladenen Namensraum hinzugefügt werden - wobei allerdings die Übersetzungen Vorrang haben, die im zu ladenden Namensraum vetreten sind.

Um die Eindeutigkeit der gewählten Namensräume sicherzustellen kann man meinst auf den Bezeichner der Klasse durch die Aufruf der Methode `get_class($this)` zurückgreifen.

3.4 Abgrenzung vom dynamischen Inhaltsobjekt

Inhaltsobjekte sind im Wesentlichen Text-Objekte, die übersetzt werden können, wobei jede weitere Strukturierung von Modul zu Modul verschieden sein kann. Es ist nicht sinnvoll eine Abbildung der Struktur des Inhaltsobjekts im Internationalisierungs-Modul vorzunehmen, da diese beliebig komplex sein kann. Durch Einführung einer solchen Abbildung ist es zwar möglich ein Inhaltsobjekt über eine einzelne ID zu erfragen, jedoch wird es im Gegenzug notwendig die potentiell sehr verschiedenen Datenstrukturen zu standardisieren. Daher enthält das Dokumenten-Wörterbuch keine inhaltsübergreifenden Strukturen, sondern ausschließlich einfache Zeichenketten.

3.5 Datenbankstruktur

3.5.1 System-Wörterbuch

Spalte	**Typ**	**Attribute**
namespace	varchar(50)	not null
language	enum	not null
token	varchar(255)	not null
translation	text	not null
primary key ("namespace","token", "lang")		
index ("namespace", "lang")		

Tabelle 2: Datenbankstruktur des System-Wörterbuches

Das System-Wörterbuch beschränkt sich aus Performancegründen auf die wesentlichen Informationen und wird wahrscheinlich als einzelne Tabelle realisiert werden, um sonst evtl. nötige Verknüpfungen zwischen mehreren Tabellen (SQL JOINs) zu vermeiden, was der Performance abträglich wäre. Damit muss man zwar explizit zu jedem Eintrag eine Sprache angeben, was aber weniger oder zumindest nicht mehr Speicher als eine Referenz einnimmt. Speichereinbußen hat man durch diese Vorgehensweise beim Speichern der Namensraumbezeichner - diese sind potentiell etwas länger. Da ca. 40 Byte Overhead pro Eintrag in Relation zur Gesamtgröße eines Eintrages

von über 500 Byte relativ gering sind, überwiegen hier eindeutig die Vorteile, die man durch die vermiedenen JOINs erzielt.

3.5.2 Dokumenten-Wörterbuch

Spalte	Typ	Attribute
id	serial	not null
language	enum	not null
source	bool	not null
content	longtext	not null
primary key ("id","lang")		
index ("id", "source")		

Tabelle 3: Datenbankstruktur des Dokumenten-Wörterbuches

Auch das Dokumenten-Wörterbuch kommt als eigenständige Tabelle daher. Die einzigen Werte, die hier mehr als einmal pro Eintrag vorkommen können, wären `id` und `language`. Da zu beiden in diesem Kontext aber keine weiteren Daten benötigt werden, wären hier Fremdschlüssel auch eher hinderlich, weshalb darauf verzichtet wurde. Erwähnenswert ist das Feld `source`, welches den Eintrag markiert (mit true), welcher als erstes mit einer bestimmten ID `id` hinzugefügt wurde. Dies soll die Quellsprache darstellen, also die Sprache, in welcher der Eintrag ursprünglich verfasst wurde. Dies soll es ermöglichen, dass später einfach festgestellt werden kann welches die Quellsprache eines Eintrages ist, um zu vermeiden, dass unnötige Übersetzungen über Drittsprachen geschehen. Beispiel: ein Eintrag wurde in Sprache A verfaßt und nach Sprache B übersetzt. Ein Übersetzer kommt nun später und will diesen Eintrag in Sprache C übersetzen. Ohne die Kenntnis, welches die Quellsprache ist, könnte er nun vlt. die Version in Sprache B nach C übersetzen. Da sich bei jeder Übersetzung Fehler einschleichen können bzw. der Sinn verfälscht werden kann ist es daher offensichtlich erstrebeneswert, die Akkumulation und evtl. Multiplikation dieser Fehler/Ungenauigkeiten zu minimieren, indem, sofern möglich, immer die Quellsprache als Ausgangssprache für eine neue Übersetzung gewählt wird. Mit der Einführung der Spalte `source` soll dies auf Datenbankebene ermöglicht werden.

4 Aufbau des Systems

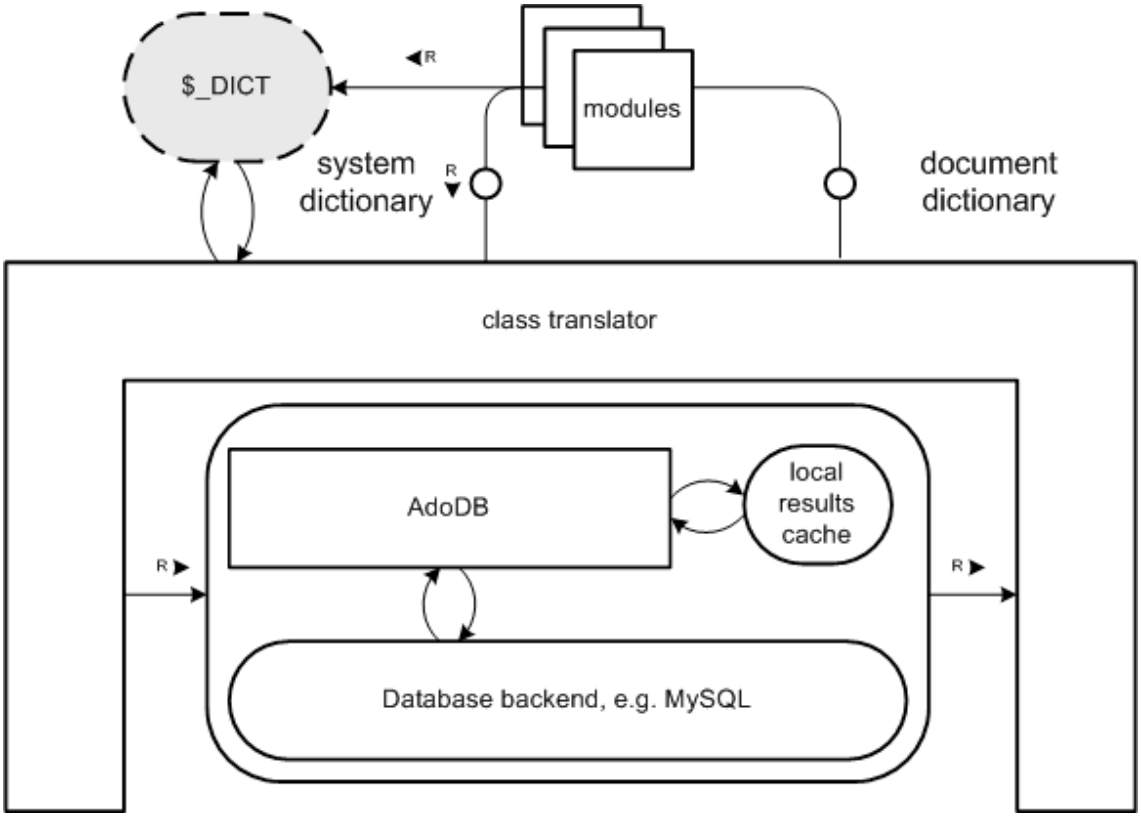

Abbildung 3: Schichtendiagramm - Konzeptioneller Aufbau des Systems

4.1 Modulinterne Realisierung

Abbildung 3 zeigt den konzeptionellen Aufbau des zu realisierenden Übersetzer-Moduls. Dabei kapselt die Klasse den Zugriff auf das zugrunde liegende Persistenz-Modell vollständig, so dass sich die Speicherung von Daten vollständig transparent darstellt.

Zum einen wird dadurch eine Komplexitätsminderung des zugrunde liegenden Systems erreicht. Zum anderen erhöht sich dadurch die Systemintegrität, da ein eigenmächtiger Zugriff auf gespeicherte Daten vermieden wird und versehentliche Datenmodifikationen nicht durchgeführt werden können.

Für die Datenhaltung wird ein Datenbankmanagement-System (DBMS) eingesetzt, welches durch das Drittanbieter-Modul ADOdb abstrahierend benutzt wird. Das heißt, auch innerhalb des Übersetzer-Moduls wird vom eingesetzten DBMS abstrahiert, um die Vielfalt der einsetzbaren DBMS-Lösungen nicht einzuschränken. ADOdb dient dabei als *Facade-Pattern*, das MySQL-Anfragen in Anfragen für eine Vielzahl gängiger DBMS umgestaltet. Damit eröffnet die Verwendung von ADOdb eine Standardisierung hinsichtlich Datenbankzugriffsmethoden, die für mehrere Systeme geeignet ist. ADOdb wird verwendet, da eine derartige Standardisierung auch in PHP 5 bislang nicht integriert ist.

Zusätzlich eröffnet die Verwendung von ADOdb eine Entlastung des verwendeten DBMS, denn durch die Bereitstellung eines eigenen Ergebnis-Caches ist es möglich gleichartige Übersetzungsanfragen ohne explizites Abfragen der Datenbank zu befriedigen.

4.2 Schnittstellen zu anderen Modulen

Aufbauend auf der im Kapitel 3 beschriebenen Zweiteilung hinsichtlich System- und Dokumenten-Wörterbuch ergeben sich unterschiedliche Zugriffsmöglichkeiten. Beiden Wörterbüchern ist die Eigenschaft gemein, dass sie ein *Singleton-Pattern* nach [TPG05, 19-24.] darstellen. Das heißt, zur Laufzeit ist jeweils genau ein repräsentierendes Objekt verfügbar. Diese Entwurfsentscheidung reduziert den Ressourcenverbrauch, der bei mehrfacher Bildung von neuen Objekten durch den Operator *new* notwendig wird und trägt so zur Performanz des Gesamtsystems bei.

4.2.1 Schnittstellen für den Zugriff auf das System-Wörterbuch

Das System-Wörterbuch eröffnet einerseits einen direkten Zugriff auf Übersetzungen durch verschiedene Methoden. Dabei können die Übersetzungen sowohl einzelner als auch mehrerer Wörter simultan abgefragt werden. Dies erfordert im Programmcode des nutzenden Moduls jedoch mehrmals explizite Methodenaufrufe, die den Programmieraufwand zusätzlich belasten können.

Daher gibt es eine weitere lesende Zugriffsmöglichkeit auf Übersetzungen einer Sprache, die durch einen einmaligen Methodenaufruf von `getSystemNamespaceTranslation` initialisiert wird. Im Anschluss daran können Übersetzungen des angegebenen Namensraums auch durch die Verwendung der globalen PHP-Variable `$GLOBALS[_DICT]` bzw. `$_DICT` befriedigt werden. Dadurch wird ein vom Modul unabhängiger Zugriff auf bereits abgefragte Namensräume möglich, so dass auf diesem Weg ebenfalls Ergebnisse zwischengespeichert werden und die Performanz gesteigert wird. Der Zusammenhang ist in Abbildung 3 oben links dargestellt. Einerseits wird ein direkter Zugriff über einen Kommunikationskanal ermöglicht. Andererseits wird der Zugriff über den gestrichelt gekennzeichneten strukturvarianten Speicher `$_DICT` eröffnet.

4.2.2 Schnittstellen für den Zugriff auf das Dokumenten-Wörterbuch

Das Dokumenten-Wörterbuch kann im Gegensatz zum System-Wörterbuch ausschließlich über direkte Zugriffsmethoden abgefragt werden. Dabei ist es möglich unter Verwendung der intern festgelegten Sprachen multimodale Übersetzungen bereitzustellen. Dies eröffnet dem Nutzer die Möglichkeit unter Verwendung derselben Dokumenten-Identifikation ohne Rücksicht auf verschiedene Übersetzungen stets die korrekte Übersetzung implizit auswählen bzw. setzen zu lassen.

Ein anschauliches Beispiel stellt hierfür die Methode `getContentTranslation` dar, die zu einer gegebenen Dokumenten-Identifikation die entsprechende Übersetzung gemäß der Spracheinstellungen des Nutzers zurückliefert. Dabei wird lediglich die ID des zu übersetzenden Dokuments der Methode als Parameter übergeben.

5 Anwendungsszenarien

Zur Verdeutlichung der Anwendung des zu realisierenden Moduls wird an dieser Stelle ein exemplarischer Ablauf innerhalb einer rein fiktiven Anwendung dargestellt.

5.1 Verwenden vorhandener Übersetzungen

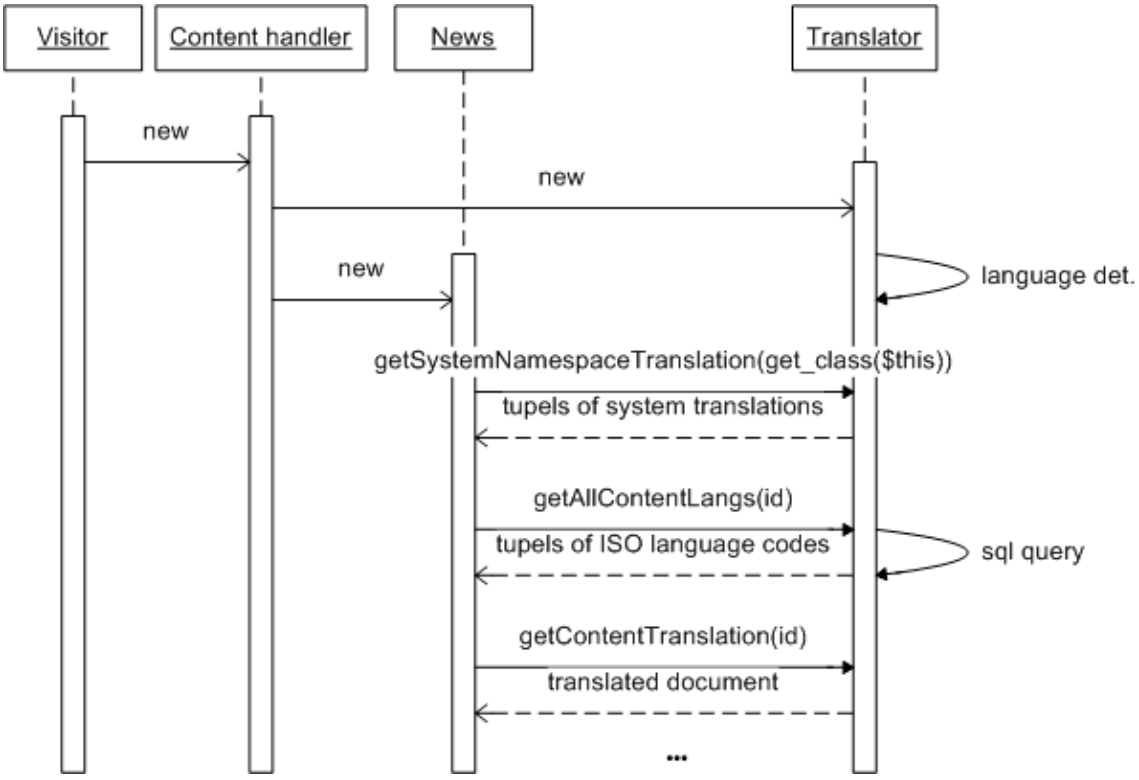

Abbildung 4: Sequenzdiagramm für das Auslesen von Übersetzungen eines Moduls

Abbildung 4 zeigt ein UML-Sequenzdiagramm, das Exemplare der Klassen Visitor, Content Handler, News und Translator im zeitlichen Ablauf darstellt, um den Bereich News unter Verwendung des Translator-Moduls mit Inhalt zu füllen.

Ein durch den Benutzer initiierter Seitenaufruf führt dazu, dass ein Objekt der Klasse *Content Handlers* erstellt wird. Der Content Handler sei hierbei eine PHP-Klasse, die der Verwaltung weiterer PHP-Module dient und den Ablauf koordiniert. Der Content Handler ist nicht Teil des Translator-Moduls.

Im Kontext des Content Handlers wird ein Exemplar der Klasse *Translator* erstellt, um von weiteren Modulen genutzt werden zu können. Diese Objektgenerierung kann auf Grund des angestrebten Singleton Patterns auch durch das News-Modul direkt geschehen. Einhergehend mit der Erstellung des Übersetzer-Moduls wird einmalig die eingebaute Spracherkennung aufgerufen, die die im HTTP-Header übermittelte Benutzereinstellung `HTTP_ACCEPT_LANGUAGE` hinsichtlich der Sprachaushandlung (Content-Negotiation) verwendet.

Im Folgenden wird durch den Content Handler ein Objekt der Klasse *News* erzeugt, die den eigentlichen Nutzer des Übersetzer-Moduls in diesem Szenario darstellt. Initial wird daher die Methode `getSystemNamespaceTranslation(get_class($this))` aufgerufen.

Hierbei definiert `get_class($this)` eindeutig den zu ladenden Namensraum aus dem System-Wörterbuch, der vor allem Sprachelemente für die Bedienung des News-Moduls beinhaltet. Die Methode bedarf keiner expliziten Kennzeichnung der zu wählenden Sprache, diese Entscheidung wird auf Basis der o.g. Spracherkennung gefällt.

Als Ergebnis des Methodenaufrufs wird ein Array mit Systemübersetzungen für das Modul *News* zurückgeliefert und zusätzlich in das global verfügbare Array `$_DICT` kopiert.

Zur Kennzeichnung der Mehrsprachigkeit auf der zu generierenden News-Seite sollen kleine Landesflaggen als graphische Metapher dargestellt werden. Zur Ermittlung der verfügbaren Sprachen wird die Methode `getAllContentLangs($id)` aufgerufen, die ein Array mit Sprachkürzeln für ein bestimmtes Dokument zurückliefert.

Im dargestellten Beispiel werden Einträge für den News-Bereich als Dokumente betrachtet und finden sich folgerichtig im Dokumenten-Wörterbuch wieder.

Durch die Methode `getContentTranslation` wird anhand der aktuellen Spracheinstellungen des Benutzers die Übersetzung des durch die Variable `$id` gegebenen Dokuments geliefert.

5.2 Erstellung neuer Dokumente und deren Übersetzungen

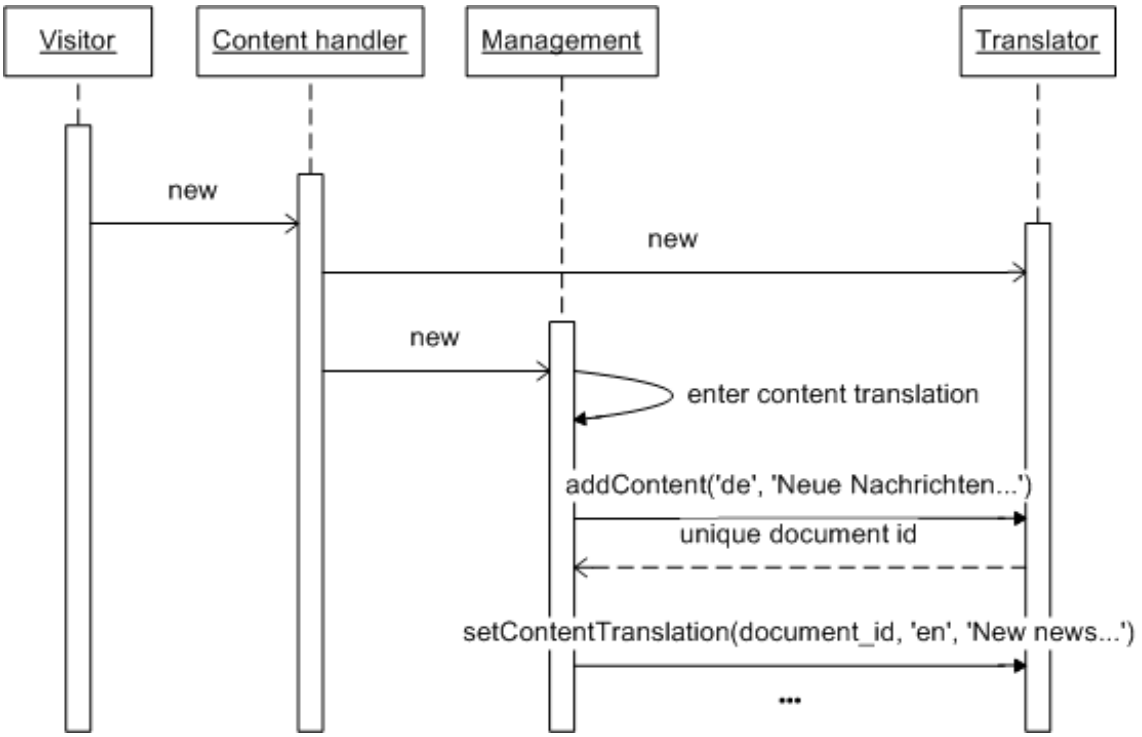

Abbildung 5: Sequenzdiagramm für das Erstellen neuer Artikel und deren Übersetzungen

Abbildung 5 zeigt ein UML-Sequenzdiagramm, das den prinzipiellen Verlauf einer Übersetzungseingabe darstellt.

Dabei führt die Anfrage des Anwenders zur Erstellung von Objekten der Klassen Content Handler, Management, sowie Translator. Die Managment-Instanz[1] stellt ein Modul dar, welches die Erfassung von neuen Artikeln und deren Übersetzungen ermöglicht.

[1]Das Management-Modul ist nicht Teil des Translator-Moduls und dient lediglich der Veranschaulichung des Anbieter-Nutzer-Prinzips im Rahmen der bereitgestellten Translator-Schnittstellen.

Nachdem ein neuer Artikel durch den dazu berechtigten Anwender erfasst wurde, wird dieser mit Aufrufen der Methode `addContent` persistent in die Datenbank geschrieben. Dabei werden als Parameter die erfasste Sprache und der hinzuzufügende Artikel-Inhalt übergeben und die zugeteilte eindeutige Dokumenten-ID zurückgeliefert.

Einhergehend mit diesem Vorgehen wird der erfasste Artikel als Artikel in der Ursprungssprache gekennzeichnet, damit weiterführende Übersetzungen stets von der Quellsprache aus stattfinden.

Falls Übersetzungen zu dem eben erfassten Artikel hinzugefügt werden sollen, so wird die Methode `setContentTranslation` aufgerufen. Dabei erhält diese Methode als Parameter die eindeutige interne Dokumenten-ID, das Sprachenkürzel der Übersetzungssprache, sowie den übersetzten Inhalt. Anhand dieser Eingabe kann die Assoziativität zur Originalsprache sichergestellt werden.

6 Interface

An dieser Stelle sei das Interface in Pseudo-Code dargestellt, was eine verkürzte Darstellung der Semantik ermöglichen soll. Die Syntax ist dabei an PHP angelehnt.

6.1 System-Wörterbuch

- `Array(Token => Translation)` **getSystemNamespaceTranslation**`(Namespace)`
  Diese Funktion liefert zu einem gegebenen `Namespace` alle Übersetzungen in einem assoziativen Array zurück. Wenn der Namespace nicht `common` ist, so wird der gemeinsame Namensraum mit in das Ergebnis eingefügt.
- `Translation` **getSystemTranslation** `(Namespace, Token[, Lang])`
  Dies liefert eine Übersetzung zur angegebenen Kombination (`Namespace`, `Token`) in der vom Benutzer bevorzugten Sprache. Wenn optional eine Sprache `Lang` angegeben wird, so wird die Übersetzung in dieser Sprache zurückgeliefert.
- **registerSystemNamespace**`(Namespace,`
  `Array(Lang => Array(Token => Translation)))`
  Registriert eine Menge von Übersetzungen für einen gegebenen `Namespace`. Dabei können die Übersetzungen für mehrere Sprachen angeben werden, weshalb der zweite Parameter ein assoziatives Array enthält, welches wiederum assoziative Arrays enthält.
- **deleteSystemNamespace**`(Namespace)`
  Dies löscht alle Übersetzungen aus dem angegebenen `Namespace`.
- `Array(Lang)` **getSystemCompleteLangs**`()` Diese Funktion liefert eine Liste mit allen Sprachen zurück, welche, in Bezug zu einer Referenzsprache, vollständig sind; also diejenigen, welche mindestens die Tokens beinhalten, welche auch in der Referenzsprache vorhanden sind.
- `Array(Lang)` **getSystemAllLangs**`()`
  Liefert eine Liste mit allen Sprachen zurück, für die im Systemwörterbuch mindestens eine Übersetzung existiert.
- **setSystemTranslation**`(Lang, Namespace, Token, Translation)`
  Diese Funktion setzt für eine bestimmte bestimmtes `Token` im angegeben `Namensraum` die Übersetzung in der Sprache `Lang` auf `Translation`. Wenn vorher keine solche Übersetzung vorhanden war, wird eine angelegt, andernfalls die bisherige überschrieben.

6.2 Dokumenten-Wörterbuch

- `Translation` **getContentTranslation**`(ID)`
  Liefert die Übersetzung eines Eintrages `ID` in einer vom Benutzer bevorzugten Sprache zurück.
- `Translation` **getContentTranslationInLang**`(ID, Lang)`
  Liefert die Übersetzung eines Eintrages `ID` in der angegeben Sprache `Lang` zurück.
- `ID` **addContent**`(Lang, Translation)`
  Fügt einen neuen Eintrag in der angegebenen Sprache `Lang` mit der dazugehörigen Übersetzung `Translation` hinzu und liefert die dabei erzeugte `ID` zurück. Dieser Eintrag wird auch automatisch als `source` markiert, siehe hierzu auch Tabelle 3 auf Seite 12.

- `ID` **addContentTranslation**`(Array(Lang => Translation))`
  Fügt einen neuen Eintrag hinzu und erlaubt es dabei, mehrere Übersetzungen in verschiedenen Sprachen anzugeben. Da in PHP auch assoziative Arrays geordnet sind, kann hier auch zugesichert werden, dass der erste Eintrag im übergebenen Array in der Datenbank als `source` markiert wird.

- **deleteContent**`(ID)`
  Löscht alle Einträge mit der angegebenen `ID` in allen Sprachen.

- `Array(Lang)` **getContentLangs**`(ID)`
  Liefert eine Liste mit allen Sprachen zurück, für die der Eintrag `ID` eine Übersetzung bereithält. Die Reihenfolge ist hier ohne Belang, der erste Eintrag im Ergebnis ist hier also nicht unbedingt die Quellsprache.

- **setContentTranslation**`(ID, Lang, Translation)`
  Fügt eine Übersetzung `Translation` in der Sprache `Lang` zum bestehenden Eintrag `ID` hinzu.

- `Lang` **getContentSourceLang**`(ID)`
  Liefert die Quellsprache des Eintrages `ID`.

7 Sprachen

Um die vom Benutzer bevorzugte Sprache zu ermitteln, werden drei verschiedene Quellen genutzt, welche jeweils eine Liste mit Sprachen in einer bestimmten Reihenfolge liefern. Diese drei Listen werden in der im Folgenden beschriebenen Reihenfolge aneinander gekettet, um alle in Betracht kommenden Sprachen zu ordnen. Nach dem Entfernen von Duplikaten erhält man so eine Liste, in der die vom Benutzer bevorzugte Sprache priorisiert ist. Verwendet man diese Liste zum Auffinden von Übersetzungen, so ist sichergestellt, dass der Benutzer immer Übersetzungen präsentiert bekommt, die mit den den individuelle gewählten Sprachpräferenzen übereinstimmen.

- Benutzer-Einstellungen im CMS
  Diese erhalten absoluten Vorrang, da diese Einstellung am feingranularsten ist. Beispielsweise ist eine Realisierung denkbar, bei der der Benutzer zu einem bestimmten Inhaltsobjekt auf eine graphische Metapher der Landessprache repräsentiert durch die Landesflagge klickt. Dadurch würde das betrachtete Inhaltsobjekt in der so gewählten Sprache dargestellt werden.

- HTTP-Header-Eintrag `HTTP_ACCEPT_LANGUAGE`
  Mittlerweise kann man in vielen Web-Browsern konfigurieren, welche Sprachen des Inhalts zurückgeliefert werden sollen. Diese Anforderung durch einen Eintrag im HTTP-Header in der Variable `HTTP_ACCEPT_LANGUAGE` in einer Liste übermittelt, wobei jedem Eintrag optional ein Gewicht zugeordnet ist. Für unsere Belange ignorieren wir die Gewichte und verwenden lediglich die Reihenfolge dieser Sprachen. Da diese üblicherweise nicht pro Webseite konfiguriert werden kann, ordnen wir dieser Liste eine geringere Priorität zu.

- Standard-Sprache
  Dies ist wahrscheinlich Englisch (en) oder Deutsch (de) und stellt die Sprache dar, welche als Referenz benutzt wird, um zu ermitteln, ob eine Sprache vollständig ist. Dies soll sicherstellen, dass wenigstens für eine Sprache immer eine Übersetzung gegeben ist. Indem wir dieser Sprache die geringste Priorität zuordnen und ans Ende der Liste der vom Benutzer bevorzugten Sprachen stellen, wird garantiert, dass der Benutzer wenigstens Text in einer Sprache erhält, falls er z.B. vergisst die Einstellungen seinen Präferenzen anzupassen oder falls schlichtweg keine Sprache vorhanden ist, die der Benutzer angegeben hat. Dies bedeutet natürlich auch, dass der Benutzer unter Umständen Text in einer Sprache präsentiert bekommt, die er nicht kennt. Statt ihm das gewünschte Dokument vorzubehalten, hat der Benutzer so die Möglichkeit sich selber um eine Übersetzung zu bemühen.

8 Glossar

Cache-Warming verwendet gezielt vereinfachte Datenbankabfragen, um Ergebnisse im Abfrage-Speicher generisch zu halten, damit möglichst viele Benutzer diese wiederverwenden können

Proxy-Pattern (auch Object-Adapter-Pattern) verdeckt eine bestehende Implementierung hinter einem neuen Interface, um neuen Anforderungen gerecht werden zu können.

Inhaltsobjekt Sprachinvarianter komplexer Datentyp, der nicht unterstützt wird; muss zum Zwecke der internationalisierung in eine Menge von Zeichenketten zerlegt werden

Dokumenten-Wörterbuch abstrahiert alle mehrsprachigen Inhalte auf Dokumentenebene und enthält jeweils mindestens das Dokument in der Ursprungssprache sowie eine variierende Anzahl von Übersetzungen des Ausgangsdokuments.

Singleton-Pattern garantiert die Existenz höchstens eines Objekts einer Klasse (Einmaligkeit), vgl. [TPG05].

Sprachnamensräume oder Namensräume sind eindeutige Bezeichner, die zur Aggregierung von Daten dienen. In diesem Kontext sind damit die Übersetzungen eines Moduls oder einer Gruppe von Modulen im System-Wörterbuch gemeint.

System-Wörterbuch dient der Speicherung von kurzen obligatorischen Übersetzungen gruppiert nach Namensräumen.

Literatur

[TPG05] The PHP Group. *Patterns in PHP*. `http://php5.de/manual/de/language.oop5.patterns.php`, October 2005.