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 Abschlussdokumentation des realisierten Moduls zur Internationalisierung von Web-Anwendungen in der Programmiersprache PHP 5 dar. Die Referenz-Implementierung ist für den Einsatz im Kontext des Tele-Teaching-Portals `http://www.tele-task.de` erstellt und an die spezifischen Bedürfnisse angepasst. Dabei wird das Konzept zweier voneinander getrennter Wörterbücher umgesetzt, 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 final draft for the database independent internationalization 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-

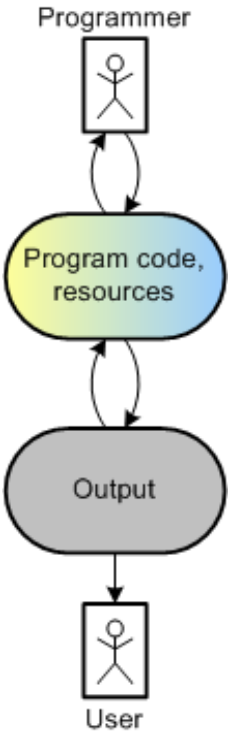

Abbildung 1: Projekte ohne Internationalisierung

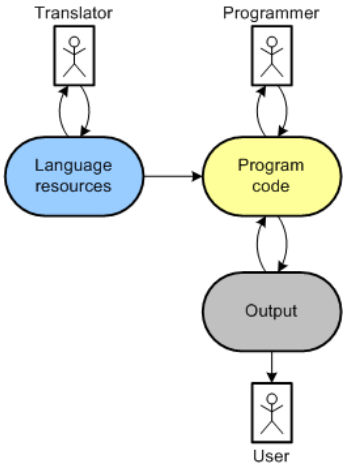

Abbildung 2: Projekte mit Internationalisierung

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, wie in Listing 2 zu erkennen.

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.

4 Aufbau des Systems

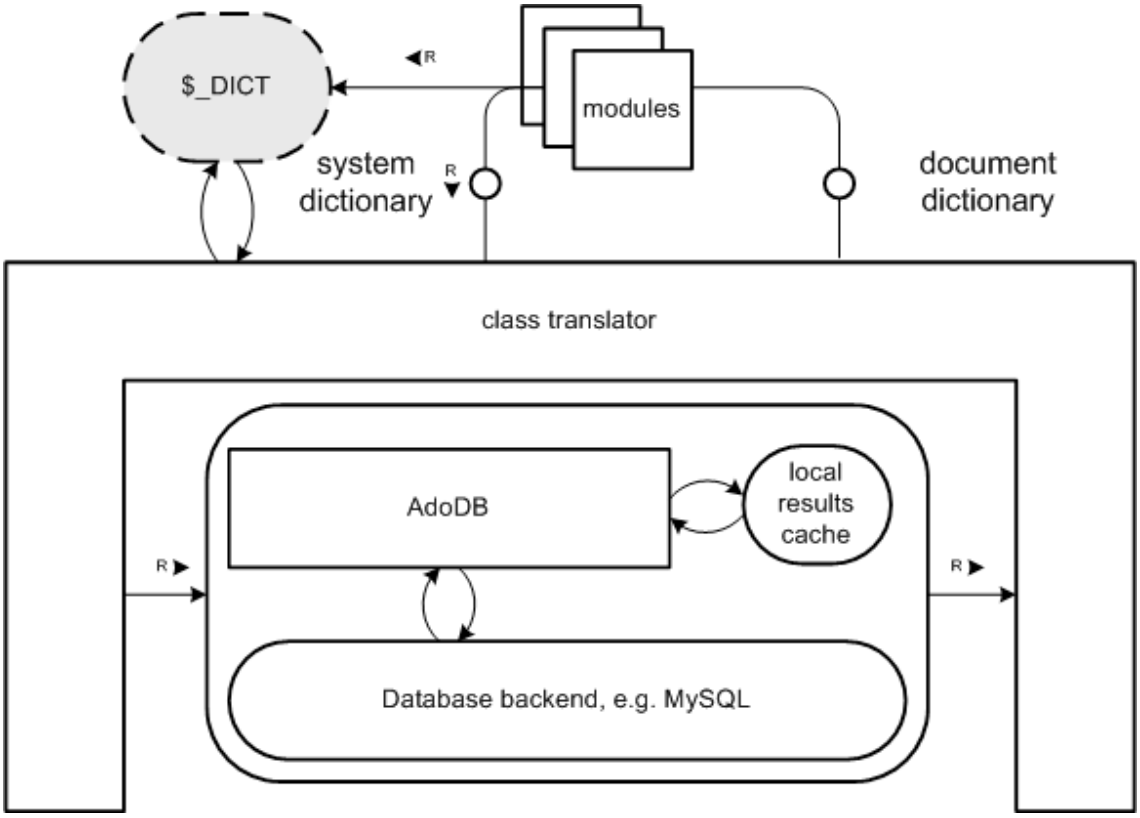

Abbildung 3: Schichtendiagramm – Konzeptioneller Aufbau des Systems

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* 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

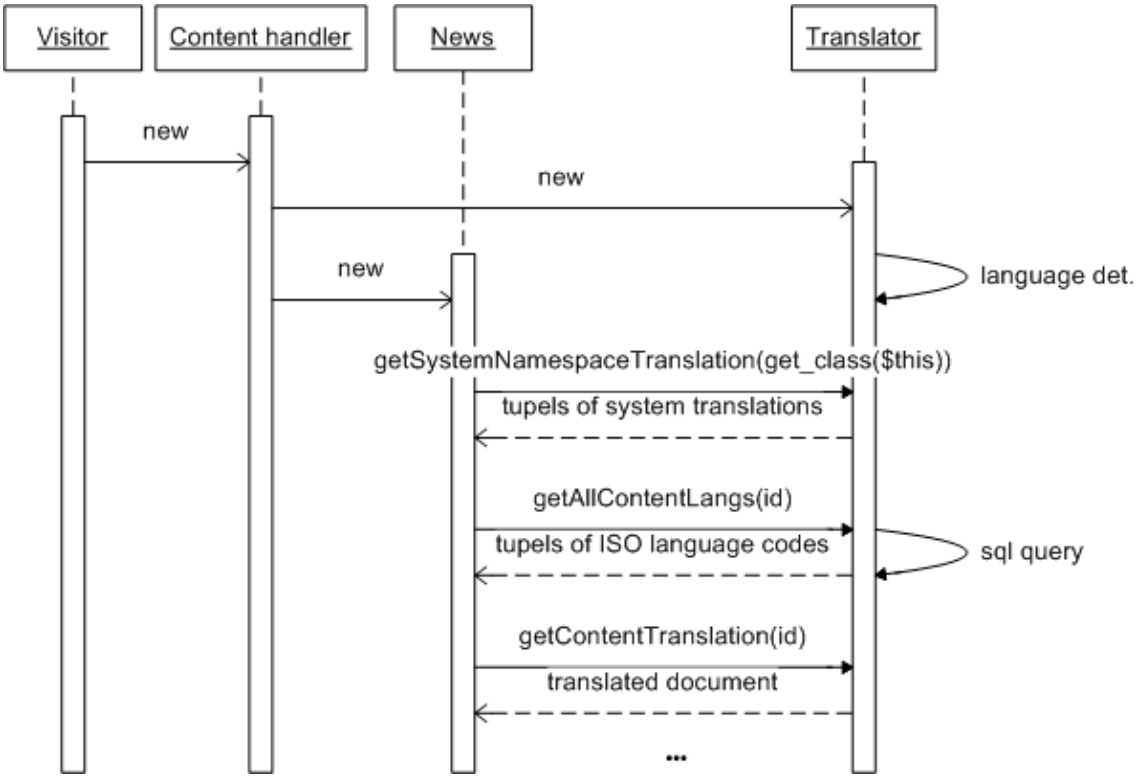

Abbildung 4: Sequenzdiagramm – Auslesen von Übersetzungen eines Moduls

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

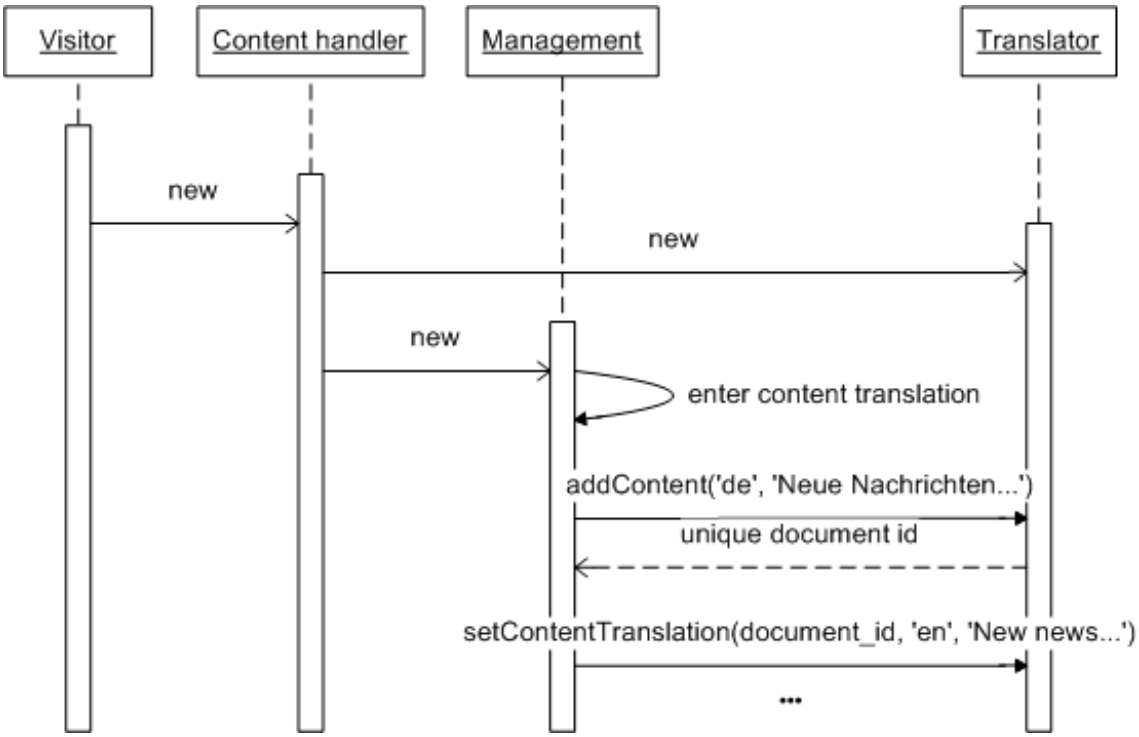

Abbildung 5: Sequenzdiagramm – Erstellen neuer Artikel und deren Übersetzungen

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-Definition

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 4 auf Seite 23.

- `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
  Die Standard-Sprache stellt die Sprache dar, die als Referenz benutzt wird, um zu ermitteln, ob eine Sprache vollständig ist oder nicht. Üblicherweise handelt es sich hierbei um Englisch mit dem Sprachkürzel *en*. Dies stellt sich, dass mindestens für diese 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 Datenbankstruktur

Im Folgenden werden die verwendeten Datenbank-Relationen und deren Entwurfsentscheidungen erläutert.

8.1 Sprachenverzeichnis

Spalte	Typ	Attribute
language	varchar(2)	not null
description	varchar(255)	not null
primary key ("language")		

Tabelle 2: Datenbankstruktur des Sprachenverzeichnisses

Diese Relation wurde erst im Zuge der Entwicklung eingeführt, um eine Liste verfügbarer Sprachkürzel nach ISO-639-1 [ISOI05] und deren Bezeichnung speichern zu können.

8.2 System-Wörterbuch

Spalte	**Typ**	**Attribute**
namespace	varchar(50)	not null
language	varchar(2)	not null
token	varchar(255)	not null
translation	text	not null
changed	timestamp	not null
primary key ("namespace","token", "language")		
index ("namespace", "language")		

Tabelle 3: 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 `JOIN`s) 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 `JOIN`s erzielt.

Im Vergleich zur ursprünglichen Planung waren hier nur recht wenig Änderungen notwendig. Anstelle eines `enum`-Datentyps mit über 100 möglichen Werten für die Spalte `language` wird jetzt ein einfaches `varchar(2)` verwendet. Auf diese Weise muss man die Datentypdefinition der Spalte nicht mit dem Sprachenverzeichnis abgleichen. Zusätzlich wurde die Spalte `changed` eingeführt, welche es ermöglicht festzustellen, wann eine Übersetzung das letzte Mal geändert wurde. Dies kann u.a. dazu verwendet werden, herauszufinden, welche Übersetzungen in Bezug zur Referenzsprache veraltet sind und unter Umständen aktualisiert werden müssen.

8.3 Dokumenten-Wörterbuch

Spalte	Typ	Attribute
id	serial	not null
language	varchar(2)	not null
content	longtext	not null
changed	timestamp	not null
primary key ("id","lang")		
fulltext key ("content")		

Tabelle 4: 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, sind `id` und `language`. Da zu beiden in diesem Kontext aber keine weiteren Daten benötigt werden, sind hier Fremdschlüssel auch eher hinderlich, weshalb darauf verzichtet wurde.

In der ursprünglichen Version war eine zusätzliche Spalte `source` vorgesehen, welches den Eintrag markiert sollte, welcher als erstes mit einer bestimmten ID `id` hinzugefügt wurde. Damit sollte die Quellsprache dargestellt werden, also die Sprache, in welcher der Eintrag ursprünglich verfasst wurde, um später unnötige Übersetzungen über Drittsprachen zu verhindern. Beispiel: ein deutschsprachiger Eintrag wurde ins Englische übersetzt. Ein weiterer Übersetzer will diesen Eintrag ins Französische übersetzen. Ohne die Kenntnis der Quellsprache, könnte er nun gewillt sein die englischsprachige Übersetzung weiter zu übersetzen, anstatt des deutschsprachigen Originaldokuments. Da sich bei jeder Übersetzung Fehler einschleichen können bzw. der Sinn verfälscht werden kann ist es daher offensichtlich erstrebenswert, die Akkumulation und evtl. Multiplikation dieser Fehler/Ungenauigkeiten zu minimieren, indem möglichst die Quellsprache als Ausgangssprache für eine neue Übersetzung gewählt wird. Mit der Einführung der Spalte `source` sollte dies auf Datenbankebene unterstützt werden.

Da diese Attribut bei allen Übersetzungen eines Dokuments identisch ist und nur beim Originaldokument abweicht wird dies wie folgt kodiert. Die Information die vorher in dieser Spalte gespeichert wurde wird nun in der Groß-/Kleinschreibung der Spalte `language` kodiert. Das heißt, falls bei einer konkreten Übersetzung das Sprachkürzel `language` vollständig groß geschrieben ist, so handelt es sich bei dieser Übersetzung um die Originalversion. Ist das Sprachkürzel hingegen vollständig klein geschrieben, so ist dies nicht der Fall. Eine Mischung von Groß-/Kleinschreibung ist unzulässig - das Verhalten der Methoden ist in diesem Falle nicht definiert.

Der Volltext-Index über `content` wurde hinzugefügt, um eine performante Volltext-Suche zu ermöglichen. Da dieser eine MyISAM-Datenbank erfordert und MyISAM im Gegensatz zu InnoDB keine Fremdschlüsselbeziehungen (Foreign Keys) unterstützt, wurde für diese Anwendung auch eine MyISAM-Datenbank verwendet. Die Foreign Keys stellen sich lediglich bei der Zuordnung zwischen Dokumenten-/System-Wörterbuch und Sprachenverzeichnis als sinnvoll heraus, wobei dies nicht mit einer mangelnden Volltextsuche aufzuwägen ist.

9 Schnittstelle

9.1 Umsetzung des Singleton-Patterns

Im Zusammenhang mit großen PHP-Projekten kann man davon ausgehen, dass eine zunehmende Zahl von PHP-Modulen internationalisierte Ausgaben anbieten wird. Dies erfordert eine gesamtheitliche Objekt-Instanz, die die jeweils benötigten Übersetzungsmethoden bereitstellt.

Um Einstellungen nur einmalig tätigen zu müssen, die Anzahl der Exemplare des Objekts Translator möglichst gering zu halten und einhergehend den Speicherverbrauch zu senken, wurde das *Singleton-Pattern* umgesetzt. Dabei handelt es sich um eine Möglichkeit des Zwischenspeicherns, die ausschließlich beim ersten Versuch einer Objekterzeugung ein neues Objekt der betreffenden Klasse erstellt und bei jedem weiteren Versuch lediglich das erstmalig generierte Objekt zurückliefert.

In Listing 1 ist die konkrete Umsetzung dargestellt. Zeile 22 zeigt, dass die übliche Sichtbarkeit des Konstruktors `public` durch `private` überschrieben wurde. Dadurch ist es nicht möglich ein Objekt der Klasse zu erzeugen durch Verwendung des Operators `new`. Um Objekte dieser Klasse verwenden zu können muss der Nutzer die bereitgestellte statische Methode `getInstance()` in Zeile 10 verwenden. Da die Methode zur selben Klasse gehört, kann beim erstmaligen Aufruf ein Objekt der Klasse erzeugt werden und wird im statischen Klassenattribut `$singleton` gespeichert.

Bei jedem weiteren Aufruf der Methode `getInstance()` wird das zwischengespeicherte Objekt zurückgeliefert.

Listing 1: Das Singleton-Pattern in PHP

```
1  /**
2   * @var Translator contains the singleton of this class
3   */
4  private static $singleton;
5
6  /**
7   * method implementing the singleton pattern
8   * ...
9   */
10 public static function getInstance() {
11   if (!isset(self::$singleton)) {
12     $object = __CLASS__;
13     self::$singleton = new $object();
14   }
15   return self::$singleton;
16 }
17
18 /**
19  * constructor of the class
20  * ...
21  */
22 private function __construct() {
23   /* ... */
24 }
```

9.2 Zugriff auf das System-Wörterbuch

Tabelle 5 zeigt bereitgestellte öffentliche Methoden, die dem Zugriff auf das System-Wörterbuch dienen.

Methode	Kurzbeschreibung	Abschnitt
registerSystem-Namespace	importiert Übersetzungen unter Angabe eines Namensraums in das Systemwörterbuch.	9.2.1, S. 25
deleteSystem-Namespace	löscht einen Namensraum teilweise oder vollständig aus dem Systemwörterbuch.	9.2.2, S. 26
getSystemComplete-Langs	liefert die Liste vollständig übersetzter Sprachen zurück	9.2.7, S. 31
getSystemAllLangs	liefert eine Liste, die zusätzlich zum Ergebnis von `getSystemCompleteLangs` teilweise übersetzte Sprachen enthält	9.2.6, S. 31
getSystemNamespace-Translation	liefert einen vollständig übersetzten Namensraum zurück	9.2.8, S. 31
getSystemTranslation	liest einzelne Übersetzung in mehreren Sprachen	9.2.4, S. 28
setSystemTranslation	setzt die Übersetzung für ein bestimmtes Kürzel (Token) in einem Namensraum	9.2.5, S. 29
deleteSystemTranslation	löscht die ausgewählte Übersetzung für ein bestimmtes Kürzel (Token) in einem Namensraum	9.2.3, S. 27

Tabelle 5: Zugriffsmethoden für das System-Wörterbuch

9.2.1 `registerSystemNamespace`

Tabelle 6 zeigt die Methodensignaturen der Methode `registerNamespace`, die zum Einbringen mehrerer Übersetzungen eines Namensraums dient. Dabei ist es angedacht, dass neue Module beim erstmaligen Integrieren in eine Web-Anwendung eine Installationsroutine durchlaufen. Dieser Installationsvorgang soll sicherstellen, das mitgelieferte Sprachdaten in das Systemwörterbuch eingetragen werden.

Listing 2: registerSystemNamespace - Importieren von Übersetzungen

```
1  $translations = array();
2  $translations[0] = array(
3    $SYS_DICT['language'] => "de",
4    $SYS_DICT['token'] => "owner",
5    $SYS_DICT['translation'] => "Besitzer",
6    $SYS_DICT['changed'] => "999"
7  );
8  $translations[1] = array(
9    $SYS_DICT['language'] => "en",
10   $SYS_DICT['token'] => "owner",
11   $SYS_DICT['translation'] => "Owner",
12 );
13 $translator->registerSystemNamespace(get_class($this), $translations);
```

Es ist darauf zu achten, dass Sprachdaten je Übersetzung eindeutig durch den Zeitstempel ihrer letzten Bearbeitung versioniert sind. Dadurch wird garantiert, dass nachträglich durchgeführte Sprachmodifikationen, die aktueller als die mit dem Modul bereitgestellten Sprachdaten sind, nicht bei erneutem Aufrufen der Methode `registerSystemNamespace` durch ältere Versionen ersetzt werden. Dennoch ist dieser Parameter optional und wird bei Nichtangabe durch den Wert 0 ersetzt. Dadurch besitzen Übersetzungen ohne korrekte Versionierung den ältesten Zeitstempel und können von jüngeren Übersetzungen jederzeit überschrieben werden.

Um das Überschreiben der Sprachdaten zu erzwingen, muss ein expliziter Aufruf der Methode `deleteSystemNamespace` erfolgen, vgl. Abschnitt 9.2.2.

```
public function registerSystemNamespace($namespace, $translations =
    null)
```

Parameter	Typ	Beschreibung
$namespace	String	Eindeutiger Bezeichner für den zu importierenden Sprachnamensraum. Ein einfacher Weg diesen eindeutig zu halten ist die Verwendung des Klassenbezeichners der bereitstellenden Klasse.
$translations	Array	Übersetzungen werden durch zweidimensionale Arrays definiert, wobei der erste Index zur Trennung einzelner Übersetzungen dient. Im zweiten Index müssen die folgenden Attribute überschrieben werden. • "**language**" Sprachkürzel, das die Sprache der jeweiligen Übersetzung kennzeichnet. • "**token**" Token, das zum Auffinden der assoziierten Übersetzung dient. • "**translation**" die eigentliche Übersetzung. • ["**changed**"] optionaler Unix-Zeitstempel, der die letzte Modifikation kennzeichnet.

Tabelle 6: Methodensignatur – `registerSystemNamespace`

9.2.2 `deleteSystemNamespace`

Listing 3: deleteSystemNamespace - Löschen ausgewählter Übersetzungen

```
1 $inserts = array();
2 $inserts[0] = array(
3   $SYS_DICT['language'] => "de",
4   $SYS_DICT['token'] => "owner",
5   $SYS_DICT['translation'] => "Besitzer",
6   $SYS_DICT['changed'] => "999"
7 );
8 // entfernt die deutschsprachige Übersetzung von "owner" aus dem
    Systemwörterbuch
9 $translator->deleteSystemNamespace("a testing", array($inserts[0]));
```

Um einen gesamten Sprachnamensraum aus dem Systemwörterbuch zu entfernen, wird die Methode `deleteSystemNamespace` nur mit dem obligatorischen Parameter `$namespace` verwendet.

Ist lediglich das Löschen einzelner Übersetzungen nötig, geschieht dies durch Verwendung eines zweidimensionalen Arrays für den Parameter `$translations`, vgl. Listing 3. Die Attribute `language`, `token` und `translation` müssen angegeben werden, um explizit eine Übersetzung zu identifizieren. Der Parameter `changed` hingegen ist optional und definiert einen Unix-Zeitstempel. Sollte der Eintrag jünger als der angegebene Zeitstempel sein, so verbleibt die Übersetzung im Systemwörterbuch, andernfalls erfolgt eine Löschung.

Die Methode `deleteSystemTranslation` aus Abschnitt 9.2.3 entfernt genau eine Übersetzung eines Namensraum.

```
public function deleteSystemNamespace($namespace, $translations = null)
```

Parameter	Typ	Beschreibung
$namespace	String	Eindeutiger Bezeichner des bestehenden Sprachnamensraum auf dem agiert werden soll. Meist handelt es sich dabei um den Klassennamen der bereitstellenden Klasse.
[$translations]	Array	Übersetzungen werden durch zweidimensionale Arrays definiert, wobei der erste Index zur Trennung einzelner Übersetzungen dient. Im zweiten Index müssen die folgenden Attribute überschrieben werden. 1. "**language**" Sprachkürzel, das die Sprache der jeweiligen Übersetzung kennzeichnet. 2. "**token**" Token, das zum Auffinden der assoziierten Übersetzung dient. 3. ["**changed**"] optionaler Unix-Zeitstempel, der die letzte Modifikation kennzeichnet, so dass nur bei Übereinstimmen der Zeitstempel gelöscht wird.

Tabelle 7: Methodensignatur – `deleteSystemNamespace`

9.2.3 `deleteSystemTranslation`

Tabelle 8 zeigt die Methode `deleteSystemTranslation`, die zum Löschen genau einer Übersetzung des System-Wörterbuchs dient.

Diese Methode kapselt einen Aufruf der Methode `deleteSystemNamespace` aus Abschnitt 9.2.2.

Listing 4: deleteSystemTranslation - Löschen einer Übersetzungen

```
1 $translator->deleteSystemTranslation("en", get_class($this), "owner");
2 // entfernt (falls vorhanden) die englischsprachige Übersetzung zum
      Token "owner"
3
4 $translator->deleteSystemTranslation("de", get_class($this), "owner",
      1136070000);
5 // entfernt die deutsprachige Übersetzung nur dann, falls diese vor dem
       01. Januar 2006 zuletzt modifiziert wurde
```

```
public function deleteSystemTranslation($language, $namespace, $token,
    $changed = null)
```

Parameter	Typ	Beschreibung
$language	String	ISO-konformes Sprachkürzel für die zu löschende Übersetzung.
$namespace	String	Eindeutiger Bezeichner des Sprachnamensraum auf dem agiert werden soll. Meist handelt es sich dabei um den Klassennamen der bereitstellenden Klasse.
$token	String	Für das Auffinden im Programmcode verwendetes Token der zu entfernen Übersetzung.
[$changed]	Integer	Optionaler Unix-Zeitstempel, der das Löschen derart eingrenzt, so dass ausschließlich Übersetzungen, die vor dem angegebenen Datum modifiziert wurden, gelöscht werden. Standardmäßig wird dieser Wert nicht berücksichtigt.

Tabelle 8: Methodensignatur – `deleteSystemTranslation`

9.2.4 `getSystemTranslation`

Die Methode `getSystemTranslation` dient dazu einzelne Übersetzungen in einer oder mehreren Sprachen zu ermitteln. Dabei verwendet die Methode die in Tabelle 9 genannten Parameter.

Auf Grund der typenunabhängigen Variablen in PHP ist der Typ des Rückgabewerts parameterbedingt unterschiedlich. Das heißt, wird die Methode nur mit den obligatorischen zwei Parametern aufgerufen, so ist der Rückgabewert genau eine Übersetzung in Form einer Zeichenkette oder bei fehlender Übersetzung false.

Listing 5: getSystemTranslation - Abfrage einer Sprache

```
1 // Liefert _wahr_ zurück, da das globale Array initial leer ist
2 echo is_null($_DICT["Translator"]["Version"]);
3
4 // Liefert "Übersetzer Version 1.1" zurück
5 echo $translator->getSystemTranslation("Translator", "Version");
6
7 // Liefert nun ebenfalls "Übersetzer Version 1.1" zurück
8 echo $_DICT["Translator"]["Version"];
```

Listing 5 zeigt die Verwendung der Methode mit den obligatorischen Parametern in Zeile 5. Im Vorhinein wir das globale Array `$_DICT` auf dessen Inhalt überprüft. Als Seiteneffekt des Methodenaufrufs wird das speicherinvariante Array an der angegeben Stelle gefüllt. Dadurch wird eine weitere lesende Zugriffsmöglichkeit, sowie ein weiterer Mechanismus zum Zwischenspeichern von Abfragen im Kontext der nutzenden Module bereitgestellt (vgl. Abschnitt 4.1).

Der optionale Parameter `$changed` hat Einfluss auf den Typ des Rückgabewerts. Listing 6 Zeile 1 zeigt die Abfrage des Begriffs *Version* in der derzeit eingestellten Präferenzsprache. Das zurückgelieferte eindimensionale Array enthält zusätzlich zu der Übersetzung unter anderem den Zeitstempel der letzten Änderungen an dieser Übersetzung.

Listing 7 Zeile 1 zeigt die Abfrage der deutschen und englischen Übersetzung des Begriffs *Version*. Das zurückgelieferte zweidimensionale Array enthält einen numerischen Index sowie einen Index

bestehend aus ISO 639-1 konformen Sprachkürzeln. Jedes Array enthält außerdem den Zeitstempel der letzten Änderungen der betreffenden Übersetzung.

Listing 6: getSystemTranslation - Abfrage einer Sprache und Zeitstempel

```
$array = $translator->getSystemTranslation("Translator", "Version",
    null, true);
print_r($array);
// liefert ein eindimensionales Array zurück
Array
(
    [0] => Übersetzer Version 1.1
    [translation] => Übersetzer Version 1.1
    [1] => de
    [language] => de
    [2] => 1133561524
    [changed] => 1133561524
)
```

Durch Verwendung des Parameters `$changed` ist es möglich Übersetzungen eindeutig zu versionieren, so dass einhergehend das unbeabsichtigte Überschreiben von neueren Übersetzungen durch ältere Versionen verhindert wird.

```
public function getSystemTranslation($namespace, $text, $languages =
    null, $changed = false, $caching = true)
```

Parameter	Typ	Beschreibung
$namespace	String	Name des zu durchsuchenden Namensraums, beispielsweise der Klassenname der aufrufenden Klasse.
$text	String	Der zu übersetzende Text in der Domain-Sprache (Englisch).
[$language]	String, Array	Optionale Zielsprache(n), standardmäßig ist dieser Parameter `null` und die internen Spracheinstellungen werden verwendet. Es kann ein einzelnes Sprachkürzel oder ein Array bestehend aus ISO 639-1 konformen Sprachkürzeln übergeben werden. Eine Überprüfung auf korrekte Sprachkürzel wird durchgeführt.
[$changed]	boolean	Optionaler Parameter, der dazu dient, im Ergebnis auch den Zeitstempel der letzten Modifikation einzuschließen. Standardmäßig ist dieser Parameter deaktiviert.
[$caching]	boolean	Optionaler Parameter, der dazu dient, Ergebnisse von Datenbanktransaktionen durch ADOdb zwischenspeichern zu lassen. Standardmäßig ist dieser Parameter aktiviert.

Tabelle 9: Methodensignatur – `getSystemTranslation`

9.2.5 `setSystemTranslation`

Mit der Methode `setSystemTranslation` wird genau eine Übersetzung dem Systemwörterbuch hinzugefügt.

Diese Methode kapselt einen Aufruf der Methode `registerSystemNamespace`, vgl. 9.2.1.

Listing 7: getSystemTranslation - Abfrage mit mehreren Sprachen und Zeitstempel

```
$array = $translator->getSystemTranslation("Translator", "Version",
    array ("de", "en"), true);
print_r($array);
// liefert ein zwei dimensionales Array zurück
Array
(
    [0] => Array
        (
            [0] => Übersetzer Version 1.1
            [translation] => Übersetzer Version 1.1
            [1] => de
            [language] => de
            [2] => 1133561524
            [changed] => 1133561524
        )

    [1] => Array
        (
            [0] => Translator Version 1.1
            [translation] => Translator Version 1.1
            [1] => en
            [language] => en
            [2] => 1133095696
            [changed] => 1133095696
        )

    [de] => Array
        (
            [0] => Übersetzer Version 1.1
            [translation] => Übersetzer Version 1.1
            [1] => de
            [language] => de
            [2] => 1133561524
            [changed] => 1133561524
        )

    [en] => Array
        (
            [0] => Translator Version 1.1
            [translation] => Translator Version 1.1
            [1] => en
            [language] => en
            [2] => 1133095696
            [changed] => 1133095696
        )

)
```

```
public function setSystemTranslation($language, $namespace, $token,
    $translation, $changed = null)
```

Parameter	Typ	Beschreibung
$language	String	ISO-konformes Sprachkürzel für die zu setzende Übersetzung.
$namespace	String	Eindeutiger Bezeichner des Sprachnamensraum auf dem agiert werden soll. Meist handelt es sich dabei um den Klassennamen der bereitstellenden Klasse.
$token	String	Für das Auffinden im Programmcode verwendetes Token der hinzuzufügenden Übersetzung.
$translation	String	Die eigentliche Übersetzung des unter Token definierten Begriffs.
[$changed]	Integer	Optionaler Unix-Zeitstempel, der dazu dient die angegebene Übersetzung eindeutig zu versionieren. Standardmäßig wird dieser Wert durch den ältest möglichsten Zeitstempel (0) ersetzt.

Tabelle 10: Methodensignatur – `setSystemTranslation`

9.2.6 `getSystemAllLangs`

Die Methode `getSystemAllLangs` liefert die Liste aller Sprachen zurück, für welche mindestens eine Übersetzung im Systemwörterbuch vorliegt.

Nutzbar ist dies z.B. um zu ermitteln, welche Sprachen teilweise übersetzt wurden. Dazu werden die Ausgaben von `getSystemAllLangs` und `getSystemCompleteLangs` vergleichen. Diejenigen Sprachen, die im Resultat von `getSystemAllLangs`, nicht aber im Resultat von `getSystemCompleteLangs` auftauchen entsprechen der Menge der unvollständig übersetzten Sprachen.

```
public function getSystemAllLangs()
```

Tabelle 11: Methodensignatur – `getSystemAllLangs`

9.2.7 `getSystemCompleteLangs`

Die Methode `getSystemCompleteLangs` liefert die Liste aller Sprachen zurück, welche in Bezug auf die Referenzsprache vollständig übersetzt sind.

```
public function getSystemCompleteLangs()
```

Tabelle 12: Methodensignatur – `getSystemCompleteLangs`

9.2.8 `getSystemNamespaceTranslation`

Die Methode `getSystemNamespaceTranslation` übersetzt einen kompletten Namensraum in eine gegebene oder eine vom Benutzer bevorzugte Sprache.

```
public function getSystemNamespaceTranslation($namespace, $language,
    $caching)
```

Parameter	Typ	Beschreibung
$namespace	String	Bezeichner des zu übersetzenden Namensraums.
[$language]	String	Optionaler Parameter: angeforderte Sprache; wenn diese weggelassen wird, so wird automatisch eine vom Benutzer bevorzugte Sprache verwendet.
[$caching]	bool	Optionaler Parameter: `true`, wenn zwischengespeicherte Anfragen verwendet werden sollen, `false`, falls dies nicht erwünscht ist. Standardmäßig ist dieser Parameter aktiviert.

Tabelle 13: Methodensignatur – `getSystemNamespaceTranslation`

9.3 Zugriff auf das Dokumenten-Wörterbuch

Tabelle 14 zeigt bereitgestellte öffentliche Methoden, die dem Zugriff auf das Dokumenten-Wörterbuch dienen.

Methode	Kurzbeschreibung	Abschnitt
addContent	trägt ein Dokument erstmalig in das Wörterbuch ein und fügt anschließend seine Übersetzungen hinzu	9.3.1, S. 32
addContentTranslations	Alias für `addContent`	9.3.1, S. 32
getContentTranslation	liefert die erste passende Dokumenten-Übersetzung unter Verwendung der voreingestellten Sprachpräferenzen.	9.3.6, S. 34
getContentTranslationInLang	liefert die Übersetzung eines Dokuments des Dokumenten-Wörterbuchs in explizit angegebenen Sprachen.	9.3.6, S. 34
deleteContent	löscht Einträge aus dem Dokumenten-Wörterbuch	9.3.3, S. 33
getAllContentLangs	liefert die Sprachen in denen die gegebenen IDs übersetzt sind	9.3.4, S. 33
getObjectSourceLang	liefert die Quellsprache eines Eintrages	9.3.5, S. 34
fullTextSearch	liefert ein zweidimensionales Array bestehend u.a. aus Dokumenten-ID, Sprache und Ranking	9.3.7, S. 37

Tabelle 14: Zugriffsmethoden für das Dokumenten-Wörterbuch

9.3.1 `addContent`

Die Methode `addContent` fügt dem Dokumenten-Wörterbuch einen Eintrag samt Übersetzungen hinzu.

```
public function addContent($translations)
```

Parameter	Typ	Beschreibung
$translations	Array	Assoziatives Array mit Sprache als Schlüssel und der Übersetzung als Wert. Die erste Sprache wird dabei als Original angesehen.

Tabelle 15: Methodensignatur – `addContent`

9.3.2 `addContentTranslations`

Die Methode `addContentTranslations` fügt dem Dokumenten-Wörterbuch mehrere Eintrag samt Übersetzungen hinzu.

Diese Methode kapselt einen Aufruf der Methode `addContent`.

```
public function addContent($translations)
```

Parameter	Typ	Beschreibung
$translations	Array	Assoziatives Array mit Sprache als Schlüssel und der Übersetzung als Wert. Die erste Sprache wird dabei als Original angesehen.

Tabelle 16: Methodensignatur – `addContentTranslations`

9.3.3 `deleteContent`

Die Methode `deleteContent` dient dem Löschen einzelner oder mehrerer Einträge im Dokumentenverzeichnis.

```
public function deleteContent($ids)
```

Parameter	Typ	Beschreibung
$ids	Integer, Array	ID oder IDs (Array of Integer) der zu löschenden Einträge im Dokumentenverzeichnis. Wenn eine oder mehrere der übergebenen IDs nicht existierten, so verursacht dies keinen Fehler.

Tabelle 17: Methodensignatur – `deleteContent`

9.3.4 `getAllContentLangs`

Die Methode `getAllContentLangs` liefert zu per `ID` gegebenen Einträgen im Dokumentenwörterbuch alle Sprachen, in welchen Übersetzungen für diese Einträge vorliegen. Dabei kann ausgewählt werden, ob die Schnittmenge (`$union` = `false`) oder die Vereinigung (`$union` = `true`) der Sprachen zurückgeliefert werden soll.

Liegt ein Dokument 1 in den Sprachen A und B vor, das Dokument 2 hingegen in den Sprachen B und C, so enthielte die Schnittmenge - die Menge der Sprachen, für die alle gegebenen Doku-

mente übersetzt sind - lediglich die Sprache B. Die Vereinigung - die Menge der Sprachen, für die mindestens ein Dokument übersetzt ist - hingegen enthielte die Sprachen A, B und C.

Wenn die Schnittmenge angefordert wurde und es existiert mindestens eines der angeforderten Dokumente nicht, so ist das Ergebnis auf jeden Fall leer. Im Falle der Vereinigung ist das Ergebnis genau dann leer, wenn keines der angeforderten Dokumente existiert.

```
public function getAllContentLangs($ids, $union)
```

Parameter	Typ	Beschreibung
$id	Integer, Array	ID (int) oder IDs (Array of Integer) der abzufragenden Einträge im Dokumentenverzeichnis. Wenn eine oder mehrere der übergebenen IDs nicht existierten, so verursacht dies keinen Fehler.
[$union]	boolean	Optionaler Parameter: `false`, wenn die Schnittmenge der Sprachen abgefragt werden soll; `true` wenn die Vereinigung abgefragt werden soll; standardmäßig wird die Schnittmenge ermittelt.

Tabelle 18: Methodensignatur – `getAllContentLangs`

9.3.5 `getObjectSourceLang`

Die Methode `getObjectSourceLang` liefert zu einem Eintrag `ID` diejenige Sprache zurück, welche in der Datenbank als Quellsprache markiert ist - also diejenige Sprache, in der das Dokument ursprünglich verfasst wurde.

Wenn kein Eintrag mit der übergebenen `ID` existiert, so wird `false` zurückgeliefert. Sollte einmal zu einem gegebenen Eintrag keine Quellsprache verzeichnet sein, so wird ebenfalls `false` zurückgeliefert. Letzteres stellt allerdings einen inkonsistenten Datenbankzustand dar und sollte mit den Methoden dieser Schnittstelle nicht erreichbar sein.

In allen anderen Fällen liefert die Methode das Sprachkürzel der Quellsprache des Eintrages zurück.

```
public function getObjectSourceLang($id)
```

Parameter	Typ	Beschreibung
$id	int	ID (int) des Eintrages, dessen Quellsprache abgefragt werden soll

Tabelle 19: Methodensignatur – `getObjectSourceLang`

9.3.6 `getContentTranslationInLang`

In Tabelle 20 sind die beiden Methodensignaturen der Methoden `getContentTranslation` und `getContentTranslationInLang` dargestellt. Dabei stellt `getContentTranslation` auf Grund der geringeren Parameter einen vereinfachten Zugriff auf die Übersetzung eines ausgewählten Dokuments unter Verwendung der klasseninternen Sprachkonfiguration bereit. Es wird ein Aufruf der Methode `getContentTranslation` gekapselt.

Listing 9 zeigt den üblichen Aufruf der Methode `getContentTranslation`, der als Ergebnis eine übersetzte Zeichenfolge des spezifizierten Dokuments oder in Ermangelung des Dokuments `false`

Listing 8: getContentTranslationInLang - mehrere Übersetzung eines Dokuments mit mehreren explizit angegebenen Spracheinstellungen

```
$array = $translator->getContentTranslationInLang(1, array('de', 'en'),
    true, false);
if ($translation === false) echo
   $GLOBALS['_DICT']['common']['missing_translation'];
else print_r($translation);

// liefert ein zweidimensionales Array oder im Fehlerfall false zurück
Array (
    [0] => Array (
                    [0] => Das ist der Probeartikel mit
                       Dokumentenschlüssel 1 in UTF-8.
                [content] => Das ist der Probeartikel mit
                   Dokumentenschlüssel 1 in UTF-8.
            [1] => DE
            [language] => DE
            [2] => 1134055715
            [changed] => 1134055715
        )
    [1] => Array (
            [0] => That is the test article #1.
            [content] => That is the test article #1.
            [1] => en
            [language] => en
            [2] => 1134923125
            [changed] => 1134923125
        )
    [DE] => Array (
                    [0] => Das ist der Probeartikel mit
                       Dokumentenschlüssel 1 in UTF-8.
                    [content] => Das ist der Probeartikel mit
                       Dokumentenschlüssel 1 in UTF-8.
            [1] => DE
            [language] => DE
            [2] => 1134055715
            [changed] => 1134055715
        )
    [en] => Array (
            [0] => That is the test article #1.
            [content] => That is the test article #1.
            [1] => en
            [language] => en
            [2] => 1134923125
            [changed] => 1134923125
        )
)
```

Listing 9: getContentTranslation - Übersetzung eines Dokuments anhand der klasseninternen Spracheinstellungen

```
$translation = $translator->getContentTranslation(1, 'de');
if ($translation === false)
  echo $GLOBALS['_DICT']['common']['missing_translation'];
else
  echo $translation;
// liefert einen String mit der entsprechenden Übersetzung oder false
   zurück:
// "Das ist der Probeartikel mit Dokumentenschlüssel 1 in UTF-8."
```

Listing 10: getContentTranslationInLang - Übersetzung eines Dokuments mit explizit angegebenen Spracheinstellungen

```
$array = $translator->getContentTranslationInLang(1, 'de', true,
   false);
if ($translation === false)
  echo $GLOBALS['_DICT']['common']['missing_translation'];
else
  print_r($translation);
// liefert ein eindimensionales Array oder im Fehlerfall false zurück
Array
(
    [0] => Das ist der Probeartikel mit Dokumentenschlüssel 1 in UTF-8.
    [content] => Das ist der Probeartikel mit Dokumentenschlüssel 1 in
       UTF-8.
    [1] => DE
    [language] => DE
    [2] => 1134055715
    [changed] => 1134055715
)
```

zurückliefert. Dabei werden die klasseninternen Spracheinstellungen verwendet. Wird auch hier der optionale Parameter `$changed` gesetzt, so ist bei vorhandener Übersetzung der Rückgabewert ein Array, wie in Listing 10 dargestellt.

Listing 10 zeigt die funktional mächtigere Methode `getContentTranslation`. Dabei wird im gezeigten Fall die deutschsprachige Übersetzung des Dokuments 1 angefragt und mit Zeitstempel der letzten Änderung zurückgeliefert. Dabei soll die Anfrage, falls möglich ohne zwischengespeicherte Ergebnise befriedigt werden. Das Ergebnis ist ein eindimensionales Array mit sowohl textuellem als auch numerischem Index.

Listing 8 erfragt simultan mehrere Übersetzungen eines Dokuments. Als Rückgabe wird ein zweidimensionales Array zurückgeliefert, wobei der erste Index sowohl in textuelle Form der ISO-Sprachkürzel als auch in numerischer Form die Übersetzung eindeutig definiert.

```
public function getContentTranslation($id, $changed = false)
public function getContentTranslationInLang($id, $languages, $changed =
    false, $caching = true)
```

Parameter	Typ	Beschreibung
$id	Integer	Dokumenten-ID zur eindeutigen Kennzeichnung eines Dokuments im Dokument-ID unabhängig von der eigentlichen Übersetzung.
[$language]	String, Array	Optionale Zielsprache(n); standardmäßig ist dieser Parameter `null` und die internen Spracheinstellungen werden verwendet. Es kann ein einzelnes Sprachkürzel oder ein Array bestehend aus ISO 639-1 konformen Sprachkürzeln übergeben werden. Eine Überprüfung auf korrekte Sprachkürzel wird durchgeführt.
[$changed]	boolean	Optionaler Parameter, der dazu dient, im Ergebnis auch den Zeitstempel der letzten Modifikation einzuschließen. Standardmäßig ist dieser Parameter deaktiviert.
[$caching]	boolean	Optionaler Parameter, der dazu dient, um Ergebnisse von Datenbanktransaktionen durch ADOdb zwischenspeichern zu lassen. Standardmäßig ist dieser Parameter aktiviert.

Tabelle 20: Methodensignatur – `getContentTranslationInLang` und `getContentTranslation`

9.3.7 `fullTextSearch`

Da das Dokumenten-Wörterbuch für eine sehr große Anzahl von Elementen konzipiert ist, wurde bereits frühzeitig eine Designentscheidung hinsichtlich des Auffindens von Dokumenten getroffen. Als Datenbanktyp der MySQL-Referenzdatenbank wurde *MyISAM* gewählt. Einerseits handelt es sich dabei um einen abwärtskompatibler Tabellentyp, der bereits mit MySQL in den Version 4 existierte. Andererseits ermöglicht dieser Tabellentyp das Anlegen eines Volltext-Indexes (`FULLTEXT`), der zum schnellen Auffinden vom Textpassagen geeignet ist.

Die von MySQL bereitgestellte Volltext-Suche kann in den in Tabelle 21 zusammengefassten Varianten durchgeführt werden. Das verwendete Verfahren kann explizit beim Aufruf der Methode durch den Parameter `$mode` gesteuert werden.

Tabelle 22 zeigt die von der Methode `fullTextSearch` verwendeten Parameter.
Die Methode liefert stets ein nach Relevanz sortiertes zweidimensionales Array zurück, welches die Dokument-ID, das Sprachkürzel des Dokuments, sowie das jeweilige Such-Ranking zurück. Durch

$mode	Suche	Beschreibung
0	normal	Ergebnis darf max. 50% des Tabelleninhalts umfassen und Stop-Wörter werden ignoriert (vgl. [MyS05b, 12.7]).
1	`IN BOOLEAN MODE`	Suchergebnis ist nicht nach Relevanz sortiert. Relevant ist ein boolescher Wert: 1 (Begriff(e) enthalten) oder 0 (Begriff(e) nicht enthalten) (vgl. [MyS05b, 12.7.1]).
2	`WITH QUERY EXPANSION`	Ein zweiter Suchvorgang versucht Synonyme zu finden und liefert ein deutlich größeres Suchergebnis bei sehr kurzen Suchbegriffen (vgl. [MyS05b, 12.7.2]).

Tabelle 21: Einstellungen für die Volltextsuche

die Dokumenten-ID und das Sprachkürzel kann die gewählte Übersetzung eindeutig identifiziert werden.

Listing 11: fullTextSearch - Suche mit klasseninternen Spracheneinstellungen

```
$array = fullTextSearch("Wasser*");
print_r($array);
// liefert ein zwei dimensionales Array zurück
Array
(
    [0] => Array
        (
            [0] => 1
            [id] => 1
            [1] => DE
            [language] => DE
            [2] => 1
            [ranking] => 1
        )

)
```

Listing 11 veranschaulicht die einfachste Verwendung der Methode. Dabei werden die klasseninternen Spracheinstellungen verwendet. Des Weiteren wird der Suchmodus 1 standardmäßig verwendet, so dass das Ergebnis-Ranking stets 1 ist. Das Sprachkürzel ist hierbei großgeschrieben, da es sich bei der deutschsprachigen Übersetzung des Artikels um die Ursprungssprache handelt (vgl. Abschnitt 8).

Listing 12 verwendet eine komplexere Anfrage, denn der Suchbegriff muss nach UTF-8 enkodiert werden. Des Weiteren soll der Zeitstempel der letzten Dokument-Modifikation in das Suchergebnis inkludiert werden und das Zwischenspeichern der Datenbankergebnisse deaktiviert werden. Das zurückgelieferte Array ist mit dem aus Listing 11 bis auf den zusätzlichen Eintrag für den Zeitstempel identisch.

Listing 13 verwendet im Gegensatz zum Listing 12 den Suchmodus 2. Dadurch liefert die Synonym-Suche außerdem verwandte Ergebnisse, die im Suchmodus 1 nicht auffindbar sind.

Listing 12: fullTextSearch - Suche mit ausgewählter Sprache und Suchmodus 1

```
$array = fullTextSearch("Wasserfälle", "de", 1, true, false);
print_r($array);
// liefert ein zwei dimensionales Array zurück
Array
(
    [0] => Array
        (
            [0] => 1
            [id] => 1
            [1] => DE
            [language] => DE
            [2] => 1
            [ranking] => 1
            [3] => 1134055715
            [changed] => 1134055715
        )
)
```

Listing 13: fullTextSearch - Suche mit ausgewählter Sprache und Suchmodus 2

```
$array = fullTextSearch("Wasserfälle", "de", 2, true, false);
print_r($array);
// liefert ein zwei dimensionales Array zurück
Array
(
    [0] => Array
        (
            [0] => 7
            [id] => 7
            [1] => de
            [language] => de
            [2] => 4.7197593775259
            [ranking] => 4.7197593775259
            [3] => 1134056625
            [changed] => 1134056625
        )
    [1] => Array
        (
            [0] => 1
            [id] => 1
            [1] => DE
            [language] => DE
            [2] => 2.2197775775001
            [ranking] => 2.2197775775001
            [3] => 1134055715
            [changed] => 1134055715
        )
)
```

```
public function fullTextSearch($searchString, $languages = null, $mode
    = 1, $encode_UTF8 = false, $changed = false, $caching = true)
```

Parameter	Typ	Beschreibung
$search-String	String	Suchanfrage mit zu suchenden Begriffen. Unterstützt werden Wildcard-Anfragen, wie `'Tele-Task*'` oder verschiedene ein- und ausschließende Suchanfragen, wie `'+Vorlesung -2004'`, vgl. [MyS05b].
[$language]	String, Array	Optionale Zielsprache(n); standardmäßig ist dieser Parameter `null` und die internen Spracheinstellungen werden verwendet. Es kann ein einzelnes Sprachkürzel oder ein Array bestehend aus ISO 639-1 konformen Sprachkürzeln übergeben werden. Eine Überprüfung auf korrekte Sprachkürzel wird durchgeführt.
[$mode]	Integer	Optionaler Parameter, kennzeichnet den zu verwendenden Suchmodus, vgl. Tabelle 21. Standardmäßig wird der Suchmodus 1 (`IN BOOLEAN MODE` verwendet, um die Ergebnissortierung zu deaktivieren und das größtmögliche Suchergebnis zurückzuliefern.
[$encode_UTF8]	boolean	Optionaler Parameter, falls er gesetzt ist, so wird die gegebene Suchanfrage zusätzlich in UTF8 umgewandelt. Standardmäßig ist dieser Parameter deaktiviert.
[$changed]	boolean	Optionaler Parameter, der dazu dient, im Ergebnis auch den Zeitstempel der letzten Modifikation einzuschließen. Standardmäßig ist dieser Parameter deaktiviert.
[$caching]	boolean	Optionaler Parameter, der dazu dient, um Ergebnisse von Datenbanktransaktionen durch ADOdb zwischenspeichern zu lassen. Standardmäßig ist dieser Parameter aktiviert.

Tabelle 22: Methodensignatur – `fullTextSearch`

9.3.8 `setContentTranslation`

Tabelle 23 zeigt die Methodensignaturen der Methode `setContentTranslation`, die zum Hinzufügen genau einer weiteren Übersetzungen zu einem bestehenden Dokument des Dokumenten-Wörterbuchs dient.

```
public function setContentTranslation($id, $language, $translation,
    $encode_UTF8 = false)
```

Parameter	Typ	Beschreibung
$id	Integer	Eindeutige ID des Originaldokuments, die beim Aufruf von `addContent`, vgl. Abschnitt 9.3.1 zurückgeliefert wurde.
$language	String	Sprachkürzel der hinzufügenden Übersetzung.
$translation	String	Die Übersetzung des Dokuments.
[$encode_UTF8]	boolean	Optionaler Parameter, der vor dem hinzuzufügen in die Datenbank die gegebene Übersetzung implizit nach UTF-8 umwandelt. Standardmäßig ist dieser Parameter deaktiviert.

Tabelle 23: Methodensignatur – `setContentTranslation`

9.3.9 `modifyContent`

Tabelle 24 zeigt die Methodensignaturen der Methode `modifyContent`, die zum Modifizieren eines existierenden Originaldokuments im Dokumenten-Wörterbuch dient.

```
public function modifyContent($id, $language, $content, $encode_UTF8 =
    false)
```

Parameter	Typ	Beschreibung
$id	Integer	Eindeutige ID des Originaldokuments, die beim Aufruf von `addContent`, vgl. Abschnitt 9.3.1 zurückgeliefert wurde.
$language	String	Sprachkürzel der hinzufügenden Übersetzung.
$content	String	Die geänderte Inhalt des Dokuments.
[$encode_UTF8]	boolean	Optionaler Parameter, der vor dem hinzuzufügen in die Datenbank die gegebene Übersetzung implizit nach UTF-8 umwandelt. Standardmäßig ist dieser Parameter deaktiviert.

Tabelle 24: Methodensignatur – `modifyContent`

9.4 Weitere Zugriffsmethoden

9.4.1 Methoden zum Auslesen der parametrisierten Datenbankstruktur

Um die Datenbanklogik möglichst flexibel vom Programmcode zu trennen, wurde der erstellte Code hinsichtlich der Tabellennamen und -werte parametrisiert. Dabei wurden die in Listing 14 Methoden bereitgestellt, um den Zugriff auf gelieferte Datenbankresultate zu vereinfachen. Weitere Informationen zum Aufbau der Datenbankstruktur sind im Abschnitt 8 zu finden.

Listing 14: Methoden zum Zugriff auf die Datenbankparametrisierung

```
print_r($translator->getSysDictMappings());
Array
(
    [namespace] => namespace
    [language] => language
    [token] => token
    [translation] => translation
    [changed] => changed
    [ranking] => ranking
)
print_r($translator->getDocDictMappings());

Array
(
    [id] => id
    [language] => language
    [original] => original
    [content] => content
    [changed] => changed
)
print_r($translator->getDictLangMappings());

Array
(
    [language] => language
    [description] => description
)
print_r($translator->getCommonNamespaceIdentifier());
common
```

```
public final function getSysDictMappings()
public final function getDocDictMappings()
public final function getDictLangMappings()
public final function getCommonNamespaceIdentifier()
```

Methode	Kurzbeschreibung	Abschnitt
getAllISOLangs	Liefert ein eindimensionales Array bestehend aus Attribut-Wert-Paaren zurück. Dabei entspricht der Array-Index dem ISO-639-1-konformen Sprachenkürzel und der assoziierte Wert entspricht der textuellen Beschreibung der Sprache.	9.4.4, S. 44
getDate	Liefert eine formatierte Datums-Zeichenfolge.	9.4.5, S. 44
getTime	Liefert eine formatierte Uhrzeit-Zeichenfolge.	9.4.5, S. 44
getDateAndTime	Liefert eine formatierte Zeichenfolge bestehend aus Datum und Uhrzeit.	9.4.5, S. 44
getSysDictMappings	Liefert ein eindimensionales Array mit textuellem Index zurück. Dabei stellen die Indizes feste datenbankunabhängige Feldbezeichner des System-Wörterbuchs dar. Die assoziierten Werte stellen den tatsächlichen Feldbezeichner im System-Wörterbuch dar, um eine bezeichnerunabhängige Realisierung zu gewährleisten.	-
getDocDictMappings	Liefert ein eindimensionales Array mit textuellem Index zurück. Dabei stellen die Indizes feste datenbankunabhängige Feldbezeichner des Dokumenten-Wörterbuchs dar. Die assoziierten Werte stellen den tatsächlichen Feldbezeichner im Dokumenten-Wörterbuch dar, um eine bezeichnerunabhängige Realisierung zu gewährleisten.	-
getDictLangMappings	Liefert ein eindimensionales Array mit textuellem Index zurück. Dabei stellen die Indizes feste datenbankunabhängige Feldbezeichner der ISO-639-1-Tabelle dar. Die assoziierten Werte stellen den tatsächlichen Feldbezeichner in der ISO-639-1-Tabelle dar, um eine bezeichnerunabhängige Realisierung zu gewährleisten.	-
getCommon-NamespaceIdentifier	Liefert den Bezeichner des Standard-Namensraums zurück, der bei gewissen Methodenaufrufen als Seiteneffekt implizit geladen wird (vgl. Abschnitt 9.2.8).	-
setDefaultLanguage	setzt die klasseninternen Sprachpräferenzen neu.	9.4.2, S. 43

Tabelle 25: Methoden zum Auslesen der parametrisierten Datenbankstruktur

9.4.2 `setDefaultLanguage`

Tabelle 26 zeigt die Methodensignaturen der Methode `setDefaultLanguage`, die dazu dient, die Sprachpräferenzen des Nutzers zu ändern.

```
public function setDefaultLanguage($languages)
```

Parameter	Typ	Beschreibung
$language	String, Array	einzelnes Sprachkürzel oder Listen von Sprachkürzeln, die durch die Array-Indexierung priorisiert werden. Das heißt, falls ein Array gegeben wird, so stellt der erste Array-Eintrag die bevorzugteste Sprache dar. Eine Überprüfung auf korrekte Sprachkürzel wird durchgeführt, so dass fehlerhafte Sprachkürzel entfernt werden.

Tabelle 26: Methodensignatur – `setDefaultLanguage`

9.4.3 `getLanguages`

Die Methode `getLanguages` liefert ein eindimensionales Array zurück, das die aktuellen Spracheinstellungen des Nutzers repräsentiert. Die Spracheinstellungen können durch einen Aufruf der Methode `setDefaultLanguage` konfiguriert werden. Die Reihenfolge der Array-Einträge definiert die Präferenz der Sprachen, wobei der erste Eintrag die bevorzugteste Sprache und der letzte Eintrag stets die fest definierte Standard-Sprache Englisch darstellt.

9.4.4 `getAllISOLangs`

Um die Sprachen für mögliche Übersetzungen zu definieren, enthält die Datenbank alle nach ISO-639-1 konformen zweibuchstabigen Sprachkürzel und deren englische Landesbezeichnung aus [ISOI05].

Listing 15: getAllISOLangs

```
1 $array = getAllISOLangs();
2 print_r($array);
3 // liefert ein eindimensionales Array zurück
4 Array
5 (
6     [aa] => Afar
7     /* ... */
8     [zu] => Zulu
9 )
```

Dabei handelt es sich bei der Sprachliste um ein Singleton-Objekt, da lediglich beim erstmaligen Aufruf der Methode der ADOdb-eigene Ergebnis-Cache befragt wird. Alle weiteren Methodenaufrufe liefern jeweils ein als klasseneigenes zwischengespeichertes Attribut zurück. Listing 27 zeigt das zurückgelieferte eindimensionales Array als Resultat des Methodenaufrufs.

```
public function getAllISOLangs()
```

Tabelle 27: Methodensignatur – `getAllISOLangs`

9.4.5 Formatierte Datums- und Uhrzeitausgaben

```
public function getDate($timestamp = null, $language = null, $GMT =
    false)
public function getTime($timestamp = null, $language = null, $GMT =
    false)
public function getDateAndTime($timestamp = null, $language = null,
    $GMT = false)
```

Parameter	Typ	Beschreibung
[$timestamp]	Integer	Optionaler Parameter, der einen Unix-Zeitstempel abweichend vom aktuellen Datum definiert. Standardmäßig ist dieser Parameter `null`, so dass ein Zeitstempel entsprechend der aktuellen Systemzeit verwendet wird.
[$language]	String, Array	Optionale Zielsprache(n); standardmäßig ist dieser Parameter `null` und die internen Spracheinstellungen werden verwendet. Es kann ein einzelnes Sprachkürzel oder ein Array bestehend aus ISO 639-1 konformen Sprachkürzeln übergeben werden. Eine Überprüfung auf korrekte Sprachkürzel wird durchgeführt.
[$GMT]	boolean	Optionaler Parameter, kennzeichnet die Zeitzone des Zeitstempel als Greenwich Mean Time (GMT). Standardmäßig ist dieser Parameter nicht gesetzt, so dass die Zeitzone des lokalen Systems verwendet wird.

Tabelle 28: Datums- und Uhrzeit-Methoden

10 Ausblick

10.1 Authentifizierter Zugriff

Die Internationalisierung erfordert je nach Anzahl unterstützter Sprachen eine Vielzahl von Übersetzungen. Dabei ist es erforderlich, dass dieser Dienst jederzeit verfügbar und manipulationsfrei arbeitet.

Der eigentliche Datenbestand eines System- oder Dokumenten-Wörterbuchs stellt nicht zwangsläufig geheime Daten dar, dennoch ist der Schutz der selbigen offenbar erforderlich.

Um den Zugriff auf den Datenbestand nur ausgewähltem Personal zu gestatten ist die Einführung einer abstrakten Sicherheitsschicht erforderlich. Durch dieses Access-Proxy-Pattern (vgl. [DB99] ist eine sehr generische Realisierung möglich, die alle im Translator-Interface angegebenen Methoden (vgl. Abschnitt 6) vollständig überschreibt und die Zugriffsmethoden so im Vererbungsbaum finalisiert.

Einerseits verhindert dies weiteres manipulatives Überschreiben bereitgestellter Methoden. Andererseits ist das Proxy-Objekt im Folgenden in der Lage über den Zugriff auf das gekapselte Klassenobjekt Translator zu entscheiden.

Der prinzipielle Ablauf ändert sich für den Nutzer nur wenig. Statt direkt ein Objekt der Klasse Translator anzufordern wird nun ein Objekt der Klasse Translator-Proxy benötigt. Je nach Zugriffsrechten des Nutzers wird das gewünschte Objekt oder ein Null-Objekt zurückgeliefert.

10.2 Optimierung des Ressourcenbedarfs

An vielen Stellen innerhalb von PHP-Projekten findet Internationalisierung ihren Einsatz. Damit trägt die Internationalisierung einen entscheidenden Teil zum Ressourcen-Bedarf der Anwendung bei.

Um möglichst gute Ergebnisse für das individuelle Anwendungsszenario am Einsatzort zu erreichen, wurden zwei Ansätze verfolgt. Die Auswahl welcher dieser beiden Ansätze Einsatz im konkreten Szenario findet hängt stark von der Charakteristik des durchschnittlich zu erwartenden Nutzers ab und kann nicht pauschal beantwortet werden. Daher ist es erforderlich, dass vor Einsatz des Übersetzungsmoduls Performanzuntersuchungen beider Implementierungen im konkreten Umfeld durchgeführt und verglichen werden.

Zum Umschalten zwischen beiden Performanzstrategien dient das boolesche Klassenattribut `ALTERNATIVE_IMPL`, das standardmäßig auf wahr steht. Damit wird die unter Abschnitt 10.2.1 beschriebene Taktik für SQL-Anfragen verwendet. Steht dieser Parameter auf unwahr, so wird das unter Abschnitt 10.3 beschriebene Verfahren zum Cache-Warming verwendet.

10.2.1 Entlastung der Datenbank durch Anfragenminimierung

Es gibt die Möglichkeit alle Anfragen an die Datenbank möglichst kompakt zu formulieren und damit genau auf den aktuellen Besucher der Webseite und dessen persönliche Spracheinstellungen zuzuschneiden. Listing 16 zeigt dies am Beispiel einer SQL-Abfrage zum Auffinden der Übersetzung eines Dokuments aus dem Dokumenten-Wörterbuch. Hierbei ist der Teil der WHERE-Bedingung von Bedeutung, der mit Hilfe einer ungeordneten Menge unter Verwendung des Operators `IN` alle präferierten Sprachen des Nutzers in die Abfrage einbezieht. Da diese ungeordnete Menge potentiell je Nutzer individuell ausgeprägt sein kann, können Anfragen auf dasselbe Dokument ebenso individuell aussehen.

Geht man zusätzlich davon aus, dass das betreffende Dokument lediglich in der Sprache Deutsch verfügbar ist, dass beliebig viele Anfragen auf genau nur ein Dokument zutreffen können. Dies stellt bereits gewaltige Anforderungen an das Datenbank-Caching, um nicht mehrmals Anfragen stellen zu müssen, die im Endeffekt stets mit demselben einen Dokument befriedigt werden.

Die in Listing 16 Zeile eins gezeigte Vorgehensweise iteriert nach Erhalt des Ergebnissen durch die Sprachpräferenzen des Nutzers und entscheidet so über die Auswahl des passenden Dokuments. Das heißt, mit zunehmender Länge der Sprachliste je Nutzer erhöht sich das Aufwand für die Nachbearbeitung des Suchergebnisses mitunter drastisch. Dieser Ansatz vermindert jedoch die Anzahl der Anfragen auf eine Datenbank und reduziert dadurch die Anzahl gebundener Ressourcen.

Listing 16: Optimierung des Ressourcenbedarfs

```
1 SELECT `content`, `language`, UNIX_TIMESTAMP(`changed`) AS `changed`
    FROM `document dictionary` WHERE `id` = '1' AND `language` IN
    ('zu', 'nl', 'aa', 'fr', 'de', 'en');
2
3 SELECT `content`, `language` FROM `document dictionary` WHERE `id` =
    '1' AND `language` = 'zu' LIMIT 1;
4 SELECT `content`, `language` FROM `document dictionary` WHERE `id` =
    '1' AND `language` = 'nl' LIMIT 1;
5 SELECT `content`, `language` FROM `document dictionary` WHERE `id` =
    '1' AND `language` = 'aa' LIMIT 1;
6 SELECT `content`, `language` FROM `document dictionary` WHERE `id` =
    '1' AND `language` = 'fr' LIMIT 1;
```

10.2.2 Entlastung der Datenbank durch *Cache-Warming*-Strategie

Durch Verwendung *Cache-Warming* werden geeignete Datenbankanfragen derart umgestaltet, dass die Teilergebnisse für möglichst viele Nutzer repräsentativ sind und wieder verwendet werden können.

Listing 16 zeigt ab Zeile drei und folgende mehrere Datenbankabfragen, die jedoch alle exakt auf dasselbe Dokument abzielen. Im konkreten Fall ist das Dokument in französischer Sprache verfügbar, so dass im Gegensatz zur Abfrage aus Zeile eins nicht alle Sprachen getestet werden müssen. Das heißt, das sukzessive Iterieren durch die Sprachpräferenzen des Nutzer wie in Abschnitt 10.2.1 beschrieben wird hierbei auf die Datenbank verlagert.

Dies führt zwar zu zusätzlichen Datenbank-Anfragen kann aber nach erstmaliger Anfrage deutlich einfacher für weitere Nutzer wieder verwendet werden. Einhergehend mit der Verwendung des ADOdb-Zwischenspeichers kann zusätzlich eine Entlastung der Datenbank stattfinden.

10.3 Performanzsteigerung durch *stored procedures*

Auf Grund der nutzerspezifischen, individuell geordneten Sprachlisten, muss je nach Länge der Listen unterschiedlicher Aufwand betrieben werden, um ein passendes Dokument zu finden. Das ist insbesondere dann der Fall, wenn das zu aufzufindende Dokument nur in einer Sprache vorliegt, der Nutzer diese Sprache aber am Ende seiner Präferenzliste eingetragen hat. Der dadurch entstehende Overhead ist mitunter beträchtlich.

Eine weitere Alternative zur Performanzsteigerung ist die Verwendung von so genannten *stored procedures* oder *user defined functions*. Dabei handelt es sich um Methoden, die direkt auf Datenbankseite ausgeführt werden und dadurch deutlich schneller Ergebnisse liefern können, u.a. durch geringere Kommunikation. Die Erstellung solcher Methoden erfordert genaue Kenntnis des eingesetzten Datenbankmanagementsystem (DBMS) und dessen Vorgehensweise bei bestimmten Anfragen.

Da der Zugriff auf die Datenbank in der vorliegenden Implementierung gekapselt ist, kann keine generische Lösung für solche Methoden gegeben werden. Ist im konkreten Einsatzszenario die Wahl des DBMS nicht variabel, so sollte man die Erstellung solcher Datenbank-Methoden zur Performanzsteigerung in Betracht ziehen, vgl. [MyS05a].

Teil I

Anhang

A 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

[DB99] Dan Becker. Design networked applications in RMI using the Adapter design pattern, May 1999.

[ISOI05] Internation Standards Organisation (ISO). ISO 639-1 code list, December 2005. `http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt`.

[MyS05a] MySQL AB, `http://dev.mysql.com/doc/refman/5.0/en/adding-functions.html`. *Adding New Functions to MySQL*, December 2005.

[MyS05b] MySQL AB, `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`. *Full-Text Search Functions*, December 2005.

[TPG05] The PHP Group. *Pattern in PHP5*. `http://php5.de/manual/de/language.oop5.patterns.php`, October 2005.