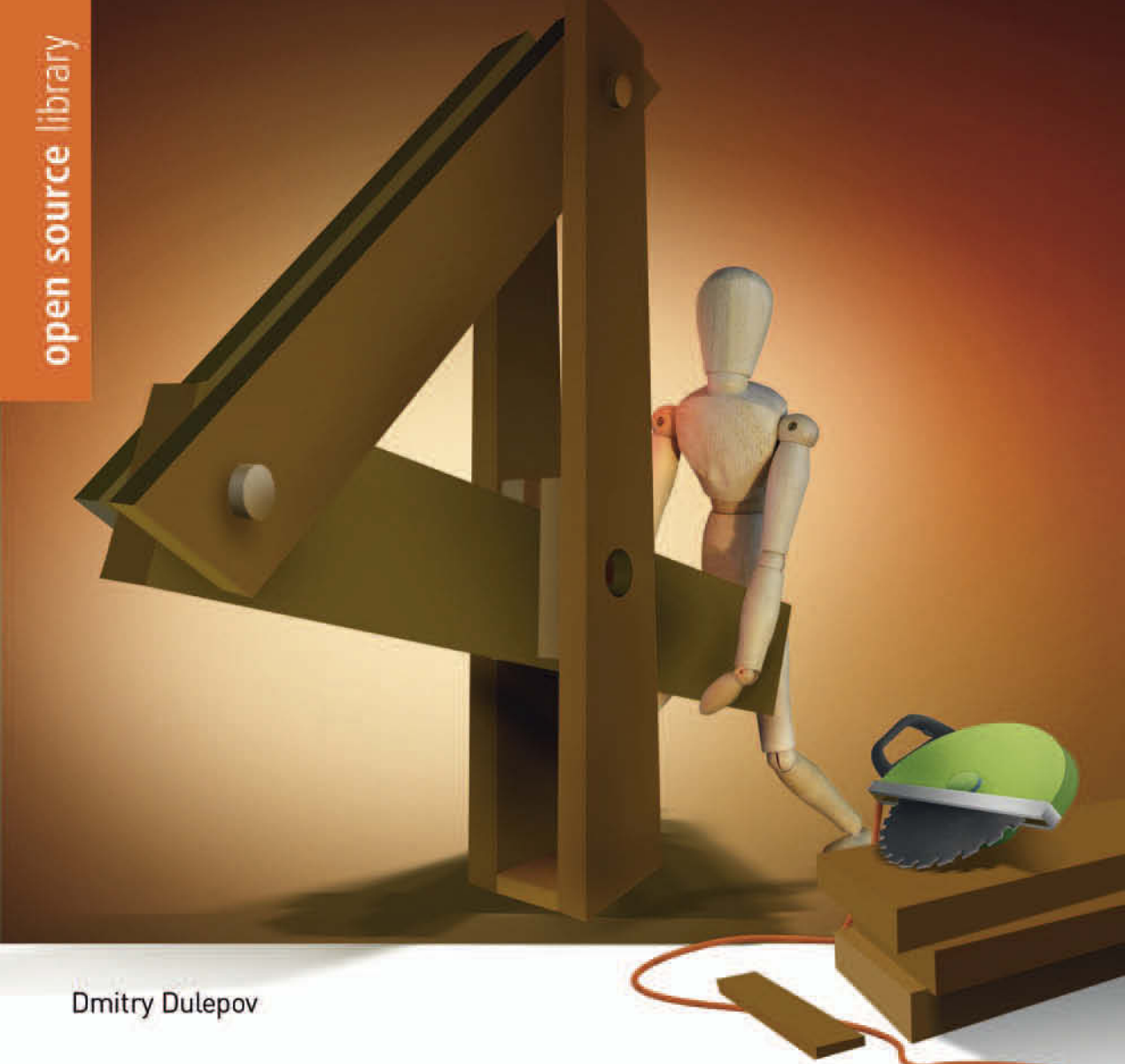
open source library



Dmitry Dulepov

TYPO3-Extensions entwickeln

Der Entwicklerleitfaden für Extensions mit der TYPO3-API

ADDISON-WESLEY

3 Extensions planen

In diesem Kapitel werden wir erläutern, warum die Planung einer Extension wichtig ist und wie man dabei vorgeht. Viele Fachbücher gehen ausführlich auf Aspekte der Planung ein, die sich auf die Webentwicklung beziehen. Hier behandeln wir die Planung nur für TYPO3-Extensions. Gegen Ende des Kapitels werden wir unsere eigene Extension planen, die wir im Laufe dieses Buchs entwickeln.

3.1 Planung ist wichtig

Die meisten Open-Source-Entwickler sehen in der Planung eine langweilige Aufgabe. Warum planen, wenn man einfach loslegen und programmieren kann? Die Antwort ist so einfach wie die Frage. Dieser »Hopplahopp«-Ansatz ermöglicht Ihnen nicht, wirklich optimalen Programmcode zu entwickeln. Teile des Codes sind auszutauschen, wenn neue geschrieben werden. Das führt häufig zu redundantem Code oder nicht initialisierten Variablen, nur in Teilen erfüllten Bedingungen und falschen Rückgabewerten. Es kommt dazu, dass es »Finger weg vom Programmcode« heißt, da jede Änderung die Stabilität des gesamten Projekts gefährdet. Oft funktioniert der Code, das Projekt ist aber dennoch eher gescheitert als geglückt, da es nicht erweitert oder wiederverwendet werden kann.

Ein anderer Grund für Planung ist die Erleichterung der Fehlerbehebung und die damit einhergehende Kostensenkung. Open-Source-Entwickler denken darüber oft nicht nach, bis sie anfangen, ihre Dienste bzw. Arbeit an kommerzielle Unternehmen zu verkaufen.

Neueren Studien zufolge wachsen die Kosten der Fehlerbehebung gegen Projektende stark an. Die Kosten sind minimal, wenn die Entwicklung noch nicht angelaufen ist und der zuständige Mitarbeiter erst die Anforderungen zusammenstellt. Wenn das Pflichtenheft erstellt ist und ein Programmierer (oder ein Team von Programmierern) anfängt, über die Umsetzung dieser Anforderungen nachzudenken, kostet eine Änderung oder Korrektur der Aufgaben immer noch nicht viel. Es kann für Entwickler aber schon schwierig sein, wenn sie sich nach Prüfung der Anforderungen schon für einen bestimmten Implementierungsansatz entschieden haben. Noch problematischer wird es in der Entwicklungsphase. Stellen Sie sich vor, dass der gewählte Ansatz falsch war und sich dies aber erst kurz vor Ende der Entwicklung herausstellt. Eine Menge Zeit ist verloren, und vielleicht muss noch einmal von vorne angefangen werden. Stellen Sie sich weiter vor, was passiert, wenn das Projekt für den Kunden freigegeben wird und dieser feststellt, dass das Produkt nicht wie erwartet funktioniert (dass etwas anders als erwartet oder gar nicht implementiert wurde). Die Kosten der Behebung sind wahrscheinlich hoch und übersteigen womöglich das Budget. Stellen Sie sich dann noch vor, was passieren würde, wenn sich Probleme nach der Veröffentlichung des Projekts ergäben.

Jetzt könnten manche Entwickler fragen, wie die Situation bei nichtkommerzieller Entwicklung aussieht, da fälschlicherweise häufig angenommen wird, dass dabei keine Kosten anfallen (zumindest keine direkten). Es gibt aber Kosten! Und häufig sind sie viel heikler als finanzielle Kosten – sie liegen im Ansehen. Wenn das Produkt eines Entwicklers nicht richtig oder gar nicht funktioniert oder offensichtliche Mängel aufweist, kann der allgemeine Ruf des Entwicklers leiden (»auf seinen Code ist kein Verlass«). Entwickler werden auch bei der Ausbesserung Probleme bekommen, weil sie oft nicht verstehen, was schief gelaufen ist. Aber die Lösung liegt nahe: Überstürzen Sie nichts! Planen Sie vernünftig! Sie können künftigen Code erst durchdenken und mit dem Programmieren dann anfangen, wenn Sie ein klares Bild haben.

Planung ist ein wichtiger Teil der Softwareentwicklung. Während freie Mitarbeiter ihre Zeit normalerweise frei zwischen Planung und Umsetzung einteilen können, haben angestellte Entwickler in vielen Fällen nicht diese Freiheit. Zudem sehen viele Vorgesetzte Planung immer noch nicht als notwendigen Schritt in der Softwareentwicklung an, was noch schlimmer ist. Dieser Sachverhalt ist gut in »Die Parabel von den zwei Programmierern« dargestellt. Wir ermutigen Sie, diese ganz zu lesen.

Bei TYPO3 ist Planung noch wichtiger als bei einer durchschnittlichen Anwendung. TYPO3 ist sehr komplex und seine Implementierung auch. Ohne Planung müssen Programmierer mit großer Wahrscheinlichkeit Änderungen am bereits geschriebenen Code vornehmen, um unvorhergesehene Probleme zu beheben. Daher ist eine gute Planung für TYPO3-Extensions extrem wichtig.

3.2 Wie wird geplant?

In der Planung gibt es verschiedene Phasen. Jede davon arbeitet normalerweise eine oder mehrere wichtige Fragen zur Entwicklung ab. TYPO3-Entwickler sollten zumindest über drei verschiedene Phasen nachdenken:

- Anforderungen sammeln
- Die Implementierung planen
- Die Dokumentation planen

Natürlich ist jedes Projekt einzigartig und hat andere Phasen. Aber diese drei Phasen gibt es im Allgemeinen in jedem Projekt.

3.2.1 Anforderungen sammeln

Zuerst muss ein Entwickler wissen, was seine Extension tun soll. Das mag recht offensichtlich klingen, aber nicht viele Autoren von Extensions wissen genau, welche Funktionalität ihr Produkt am Ende haben wird. Sie entwickelt sich mit der Zeit, und oft unterscheidet sich die ursprüngliche Idee vollständig von der letztendlichen Umsetzung. Wie vorherzusehen ist, wird dabei weder die ursprüngliche noch die endgültige gut ausgeführt.

Im anderen Fall, wenn die Anforderungen an die Extension gesammelt, gründlich geplant und planmäßig implementiert werden, ergeben sie zumeist ein harmonisches Gesamtbild.

Das Erste, was Sie beim Erstellen einer Extension tun sollten, ist daher herauszufinden, wozu sie dienen soll. Das nennt sich »Anforderungen sammeln«.

Bei nichtkommerziellen Extensions bedeutet das Sammeln von Anforderungen einfach aufzuschreiben, was jede Extension tun sollte. Bei einer News-Extension könnte dies beispielsweise wie folgt aussehen:

- News nach Datum sortiert anzeigen
- Neueste News anzeigen
- Newsarchiv anzeigen
- Nur wenig Text in einer Auflistung von News anzeigen

Wie Sie gesehen haben, sieht das Sammeln von Anforderungen leichter aus, als es eigentlich ist. Es kann jedoch noch komplexer werden, wenn eine Extension für einen externen Kunden entwickelt wird.

Alan Cooper zeigt in seinem berühmten Buch *About Face* auf, wie Benutzer, Softwarearchitekten und Entwickler das gleiche Produkt sehen. Für den Benutzer sieht es wie ein vollkommener Kreis aus. Ein Softwarearchitekt sieht eher etwas wie ein Achteck. Ein Entwickler erstellt etwas, das wie ein Polygon mit vielen Segmenten aussieht, die in verschiedenen Winkeln angebracht sind. Diese Unterschiede gibt es immer, und jede beteiligte Partei ist daran interessiert, sie zu minimieren. Ein Entwickler darf keine Angst haben, Fragen zu stellen. Je genauer das Bild ist, das er hat, desto besser wird er die Bedürfnisse des Kunden verstehen.

3.2.2 Die Implementierung planen

Wenn die Anforderungen gesammelt sind, muss darüber nachgedacht werden, welche Blöcke eine Extension enthalten soll. Es kann Blöcke für den Datenabruf, für die Darstellung, für Konvertierungen usw. geben. Bei der Umsetzung einer TYPO3-Extension sollte die Planung in einer Auflistung von Frontend-Plugins, Backend-Modulen und unabhängigen Klassen enden. Die Aufgaben jedes Plugins, jedes Moduls und jeder Klasse müssen klar sein.

Wenn es um Frontend-Plugins geht, sind Aspekte des Cachings zu berücksichtigen. Während der Großteil der Ausgabe zur Leistungssteigerung von TYPO3 gecacht werden kann, sollte die Verarbeitung von Formularen ausgenommen werden. Einige Erweiterungen verhindern das Cachen der Seite bei der Verarbeitung von Formularen gänzlich. Es gibt aber einen besseren Ansatz, nämlich ein gesondertes Frontend-Plugin der nicht gecachten Ausgabe.

Bei Backend-Modulen müssen Sie auf die Bedienerfreundlichkeit achten. Standardmäßige Backend-Navigation ist nicht sehr flexibel, und das ist bei der Planung von Backend-Modulen zu berücksichtigen.

Bestimmte Funktionsbereiche können in gesonderte Klassen verschoben werden. Das umfasst allgemeine Funktionen und alle öffentlichen APIs, die Extensions für andere Extensions anbieten. Hooks oder »user functions« werden normalerweise je nach Funktionsbereich bzw. gehookter Klasse abgelegt.

3.2.3 Die Dokumentation planen

Eine gute Extension geht immer mit Dokumentation einher. Auch sie sollte geplant werden. Normalerweise wird die Dokumentation einer Extension mithilfe einer Vorlage erstellt, in der bestimmte Abschnitte vorgegeben sind. Auch wenn dies den Entwicklern das Schreiben der Dokumentation erleichtert, müssen sie immer noch selbst planen, was sie unter diese Abschnitte fassen wollen.

3.3 Planung für TYPO3

Es gibt mehrere TYPO3-spezifische Aspekte bei der Planung. Entwickler müssen sich vor der eigentlichen Entwicklung um diese Besonderheiten kümmern.

3.3.1 Extension Keys

Jede Extension muss einen eindeutigen Schlüssel haben. Die Extension Keys können alphanumerische Zeichen und Unterstriche enthalten. Sie dürfen nicht mit einer Ziffer, dem Buchstaben `u` oder dem Präfix `test_` anfangen. Nicht jede Kombination von Zeichen ergibt einen guten Extension Key.

Ein Extension Key muss beschreibend, darf aber nicht zu lang sein. Persönliche oder unternehmensbezogene Präfixe sind nicht verboten, werden aber nicht empfohlen. Unterstriche sollten vermieden werden. Abkürzungen sind ebenfalls zu vermeiden, weil sie für andere Benutzer oft keinen Sinn ergeben.

Folgende Beispiele zeigen gute Extension Keys:

- `news`
- `comments`
- `usertracker`
- `loginbox`

Hier einige Beispiele für schlechte Extension Keys:

- `news_extension`
- `mycorp_ustr`
- `myverygoodextensionthatdoesalotofthings`
- `mvgetdalot`

- john_ext
- div2007

3.3.2 Die Datenbankstruktur

Die meisten TYPO3-Extensions verwenden die Datenbank, um Daten zu laden oder zu speichern. Eine Änderung der Datenstruktur während der Anwendungsentwicklung kann diese stark verlangsamen oder sogar Daten beschädigen, wenn bereits Eingaben in das System erfolgt sind. Daher ist es außerordentlich wichtig, die Datenstruktur einer Extension mit Weitsicht zu durchdenken. Das erfordert Kenntnisse darüber, wie Datenbanktabellen in TYPO3 aufgebaut sind.

Tabellen in TYPO3-Datenbanken müssen bestimmte Strukturen aufweisen, damit TYPO3 sie ordnungsgemäß verwalten kann. Wenn eine Tabelle die Anforderungen von TYPO3 nicht erfüllt, sehen Benutzer eventuell Fehlermeldungen im Backend (insbesondere im Modul `Web > List`) und die Daten können beschädigt werden.

Jeder Eintrag in jeder TYPO3-Tabelle gehört zu einer bestimmten Seite in TYPO3. TYPO3 ist in der Lage, diese Zuordnung für jeden Eintrag herauszufinden.

Feldnamen

TYPO3 verlangt, dass jede Tabelle mindestens zwei Felder mit vordefinierten Namen hat:

- `uid`

  Hierbei handelt es sich um eine eindeutige Kennung (ID). Es muss ein Feld sein, das automatisch ganzzahlig hochzählt (Autoincrement).

- `pid`

  Dieses Feld bestimmt, zu welcher Seite der Eintrag gehört. Wenn dieses Feld 0 ist, zeigt das an, dass die Seite zum »root level« gehört (was im TYPO3-Seitenbaum durch ein Globus-Symbol dargestellt wird).

Es gibt weitere (optionale) Felder, deren Namen sich in der Tabellenkonfiguration ändern lassen, im Normalfall aber in jeder Tabelle unverändert bleiben:

- `crdate`

  Hält Datum und Uhrzeit der Erstellung eines Eintrags als Unix-Timestamp fest.

- `tstamp`

  Hält Datum und Uhrzeit der letzten Änderung eines Eintrags als Unix-Timestamp fest.

- `deleted`

  Wird dieses Feld auf einen von null verschiedenen Wert gesetzt, wird der Eintrag als gelöscht angesehen (und damit weder im Backend noch im Frontend angezeigt).

Gelöschte Einträge verbleiben in der Datenbank und können von bestimmten Extensions wiederhergestellt werden. Ist dieses Feld nicht definiert, werden Einträge wirklich aus der Datenbank gelöscht.

- `hidden`

  Wird dieses Feld auf einen von null verschiedenen Wert gesetzt, wird der Eintrag verborgen (also im Frontend nicht angezeigt).

- `starttime`

  Gibt Datum und Uhrzeit des Zeitpunkts, an dem ein Eintrag verfügbar (also im Frontend angezeigt) wird in Form eines Unix-Timestamp an.

- `endtime`

  Gibt Datum und Uhrzeit des Zeitpunkts, an dem ein Eintrag aufhört, verfügbar zu sein (also im Frontend angezeigt zu werden), in der Form eines Unix-Timestamp an.

- `cruser_id`

  Der `uid`-Wert des Backend-Benutzers, der diesen Eintrag erstellt hat. Normalerweise hat er den Wert 0, wenn der Eintrag über das Frontend-Plugin erfolgte.

- `fe_group`

  Ein Frontend-Benutzer kann auf den Eintrag nur dann zugreifen, wenn er zu einer oder mehreren Gruppen dieser Liste gehört. Was »zugreifen« bedeutet, hängt von der Anwendung ab, aber gewöhnlich ist der Eintrag für Benutzer außerhalb dieser Gruppen vollständig unzugänglich.

- `l18n_parent`

  Bezieht sich auf die Lokalisierung von Einträgen.

- `l18n_diffsource`

  Bezieht sich auf die Lokalisierung von Einträgen.

- `sys_language_uid`

  Bezieht sich auf die Lokalisierung von Einträgen.

Mehrere andere Felder sind optional, ihre Namen gibt TYPO3 jedoch vor. Sie sind für Workspaces und Versionierung relevant:

- `t3ver_oid`
- `t3ver_id`
- `t3ver_wsid`
- `t3ver_label`
- `t3ver_state`
- `t3ver_stage`

- t3ver_count
- t3ver_tstamp
- t3ver_move_id
- t3_origuid

Andere Feldnamen sind frei verwendbar.

Je nach ihrem Zweck und ihrem Datentyp in TYPO3 gehören Felder normalerweise zu einem der folgenden Datenbanktypen:

TYPO3-Datentyp	Datenbanktyp
Eingabefeld	varchar
	tinytext
	int
Textbereich (einschließlich RTE)	Text
Checkbox	Int
Radio Button	Int
Selectbox (einfache Werte)	
Selectbox (Datenbankrelation)	varchar
Datenbankrelation	varchar
	int (in manchen Fällen möglich, aber nicht zu empfehlen)
	text
Schreibgeschützt	varchar
Benutzerdefiniertes Feld	int
	text

Tabelle 3.1: Felder in TYPO3 und ihre Datentypen

Eine oft von Extension-Entwicklern gestellte Frage lautet, warum davon abgeraten wird, einfache int-Felder für Datenbankrelationen zu verwenden. Die Antwort liegt in der Art und Weise, wie TYPO3 Referenzen auf Einträge speichert. Normalerweise ist das der uid-Wert eines anderen Eintrags, aber ihm kann auch ein Tabellenname vorangestellt sein (zum Beispiel tt_content_10). TYPO3 versteht beide Formate, wobei das zweite es ermöglicht, in einem Feld auf Einträge in verschiedenen Tabellen zu verweisen. Es leuchtet ein, dass das int-Feld all diese Informationen nicht aufnehmen kann.

Bei der Generierung einer Extension wird eine Datenbanktabellenstruktur zusammen mit anderen Dateien entwickelt. Wenn die Aufgaben einer Extension ordentlich geplant werden, sind während der Entwicklung nur geringfügige Änderungen erforderlich.

Indizes

Indizes sind die gelungensten Aspekte von Datenbanken, aber auch die verzwicktesten. Sie helfen, Daten schneller auszuwählen. Dieses Thema ist sehr umfangreich, weshalb wir es nur auf einem ganz elementaren Niveau behandeln können.

Standardmäßig erstellt TYPO3 zwei Indizes: einen für `uid`- und einen weiteren für `pid`-Spalten. Der erste wählt einen Eintrag nach seiner eindeutigen ID aus, während der zweite Abfragen für das Modul `Web > List` beschleunigt.

Entwickler sollten Abfragen hinzufügen, die ihren Extensions helfen, Daten schneller abzurufen. Die folgenden Tipps helfen dabei, bessere Indizes zu erstellen.

Eine Vielzahl von Indizes auf voneinander getrennten Feldern ist nicht sinnvoll. MySQL verwendet immer nur einen Index nach dem anderen. Wenn eine Abfrage daher aus vielen Feldern besteht, sollten dem Index ein oder mehrere Felder hinzugefügt werden. Ist der Index allerdings zu groß, kann es passieren, dass MySQL ihn ignoriert und die gesamte Tabelle nach den Einträgen absucht. Aus diesem Grund sollte ein Index nicht aus mehr als drei oder vier Feldern bestehen.

Felder sollten im Index in der gleichen Reihenfolge wie in der Abfrage aufgeführt werden. Das hilft MySQL dabei, den richtigen Index zu wählen.

Wenn eine Abfrage die Sortierung nutzt, sollte das Sortierungsfeld als letztes Feld des Indexes eingefügt sein. Befindet es sich nicht am Ende des Indexes, ignoriert MySQL den Index wahrscheinlich.

Jedes `text`- und `varchar`-Feld sollte eine Längenangabe im Index enthalten, um die Länge des Indexes zu minimieren.

Die MySQL-Anweisung `EXPLAIN` hilft einem Entwickler zu erkennen, wie Indizes verwendet werden. Sie sollte mit einem echten Satz von Daten verwendet werden, da sie die Abfrage mithilfe von Daten auswertet.

Datenbankrelationen

In TYPO3 gibt es drei Arten von Datenbankrelationen:

- Herkömmliche Relationen
- m:n-Relationen
- Inline-Bearbeitung relationaler Datensätze (Inline Relational Record Editing, IR-RE)

Herkömmliche Relationen gab es in TYPO3 als Erstes. Sie können einseitige (nur von einer Tabelle auf eine andere) und gegenseitige Referenzen enthalten. Relationen mit einer anderen Tabelle werden hierbei in der verweisenden Tabelle als Zahl (die `uid` des Eintrags in einer anderen Tabelle) oder als Tabellenname mit einem Unterstrich und dem `uid`-Wert gespeichert. Die letztgenannte Syntax ermöglicht Referenzen auf jeden Eintrag im System.

m:n-Relationen verwenden eine gesonderte Tabelle, um Relationen zwischen zwei Tabellen zu speichern. Derartige Relationen sind immer gegenseitig. Ein besonderes Feld ist auch hier in beiden referenzierten Tabellen vorhanden, aber es enthält die Anzahl der Relationen, um schnell ermitteln zu können, ob es Referenzen gibt. Obwohl TYPO3 diese Methode uneingeschränkt unterstützt, wird sie nur selten angewendet. Sie erfordert zusätzliche Abfragen und Mehrarbeit dabei, die Referenzen zu erhalten und sicherzustellen, dass keine toten Enden auftreten. Der andere Nachteil von m:n-Relationen ist die Schwierigkeit, die Referenzen mit einem kurzen Blick auf die Daten zu verstehen. Das erfordert eine gewisse Konzentration auf mehrere Datenbankfelder in m:n-Tabellen, was es schwieriger macht, Fehler bei den Relationen zu beheben. Andererseits sind m:n-Relationen besser als herkömmliche. Die Informatik zieht diesen Relationstyp allen anderen vor.

Die Inline-Bearbeitung relationaler Datensätze (*Inline Relational Record Editing*, IRRE) ist am neuesten. Sie ermöglicht es, zueinander in Beziehung stehende Einträge als Teil des Haupteintrags zu bearbeiten. Bei der Verwendung von IRRE gibt es viele Möglichkeiten, Daten zu speichern. IRRE ist ein enorm breites und komplexes Thema, das nicht in dieses Buch gehört. Wir empfehlen jedem, der sich für IRRE interessiert, sich auf der Website `typo3.org` die entsprechende Dokumentation zu suchen.

In der Praxis nutzen die Entwickler von Extensions normalerweise herkömmliche Relationen. Sie sind die schnellsten und einfachsten und haben sich über Jahre in erfolgreicher Arbeit bewährt.

3.4 Die Planung von Extensions

In diesem Abschnitt planen wir die Extension, die wir in diesem Buch entwickeln, nämlich eine Extension zum Thema »Auflistung und Statistik der Frontend-Benutzer«. Betrachten wir nun, was sie tun soll.

3.4.1 Anforderungen

Beginnen wir mit der Definition einiger Anforderungen für unsere Extension.

Funktionalitäten

Diese Extension muss die folgenden Aufgaben erfüllen:

- Eine Liste der Frontend-Benutzer im Frontend anzeigen
  - Der Benutzer der Extension muss die Liste der Felder anpassen können. Folgende Felder sollen standardmäßig angezeigt werden:
    - Anmeldename des Benutzers
    - Tatsächlicher Name des Benutzers
    - Datum der Registrierung
    - Letzte Anmeldung

  - Seiten bei einer langen Liste nummerieren
  - Die Einträge der Listenansicht auf eine Einzelansicht der Benutzer verlinken
- Informationen zu einzelnen Benutzern mit anpassbaren Feldern anzeigen
- Liste der Frontend-Benutzer im Backend anzeigen
  - Einfaches Filtern nach Benutzernamen ermöglichen
  - Die Bearbeitung der Einträge ermöglichen
- Benutzerstatistiken im Backend anzeigen:
  - Wie oft hat sich ein Benutzer angemeldet?
  - Wann hat sich ein Benutzer zuletzt angemeldet?
  - Wie lange war der Benutzer angemeldet?
  - Welche Seiten hat der Benutzer besucht und wie oft?

Die Statistikaufgaben werden mithilfe der bestehenden Extension `loginusertrack` modelliert, aber wir werden sie ganz neu schreiben, weil diese Extension zu alt ist und nicht die neue TYPO3-API verwendet.

Usability und Erweiterbarkeit

- Website-Entwickler müssen das Erscheinungsbild des Frontends leicht anpassen können (sowohl in CSS als auch in HTML).
- Die Extension sollte es ermöglichen, in der Zukunft auf einfache Weise neue Funktionen hinzuzufügen.
- Die Extension sollte sowohl die TypoScript- als auch die benutzerfreundliche FlexForm-Konfiguration verwenden.
- Die Extension muss vollständig lokalisierungsfähig sein.

Technische Anforderungen

- Die Extension muss richtig mit dem Cache umgehen.
- Die Extension muss vom Entwickler dokumentiert werden und Kommentare im Code enthalten.
- Die Extension wird beim Schreiben der Seitenstatistiken keinen Abfragestring berücksichtigen. Nur die Seiten-ID wird erfasst.

3.4.2 Der Extension Key

Wir werden `feuserstat` als Extension Key verwenden. Ein so gewählter Schlüssel bietet mehrere Vorteile:

- Er ist kurz, aber nicht zu kurz.
- Er ähnelt dem Titel einer Extension.
- Er gibt dem Benutzer eine Vorstellung davon, worum es in der Extension geht.

Zur Registrierung des Extension Keys haben wir ein Konto auf der Website *typo3.org* eröffnet und einen neuen Schlüssel im Unterabschnitt EXTENSION KEY des Abschnitts EXTENSIONS der Website registriert. Dieser Schlüssel ist nun vom Autor dieses Buchs angemeldet. Sie sollten nicht versuchen, ihn zu registrieren, da dies fehlschlagen wird, doch Sie können den Schlüssel `user_feuserstat` verwenden, wenn Sie alle Programmierschritte selbst wiederholen möchten. Beachten Sie, dass sich die Tabellen- und Klassennamen entsprechend ändern.

3.4.3 Frontend-Plugins

Unsere Extension wird ein Frontend-Plugin beinhalten, das zwei Funktionalitäten bereitstellt:

- Die Benutzerliste anzeigen
- Informationen zu einzelnen Benutzern anzeigen

Das Plugin ist cachefähig. Es hat klar abgegrenzte, vernünftig dimensionierte Methoden. Wir möchten Codedoppelungen vermeiden und Code so schreiben, dass er von künftigen Funktionen wiederverwendet werden kann.

Eine Frage, die Entwickler hin und wieder lösen müssen, ist die, wie viele Plugins erstellt werden sollten. Hier zum Beispiel werden Auflistung und Einzelansicht in einem Plugin vereint. Dadurch wird die PHP-Klasse größer, was die Verwaltung (allein aufgrund der größeren Datei) erschwert. Wenn Auflistung und Einzelansicht auf separate Plugins aufgeteilt werden, sind sie leichter zu verwalten, aber jede gemeinsame Funktion wird dann auf eine weitere gemeinsame Klasse zugreifen müssen, womit sich die Anzahl der Plugins verdoppelt. Zwei Plugins verlängern die Liste der Plugins in TYPO3 unnötig. Daher entscheiden sich viele Autoren von Extensions für ein einzelnes Plugin mit mehreren Modi.

Eine weitere Alternative zu der Lösung mit vielen Plugins besteht darin, für jede Ansicht eine gesonderte Klasse zu verwenden.

Ziehen Sie diese verschiedenen Möglichkeiten immer in Betracht, wenn Sie Plugins planen.

3.4.4 Das Backend-Modul

Unsere Extension wird ein Backend-Modul haben, das die in den Anforderungen vorgegebenen Funktionen implementiert.

3.4.5 Weitere Klassen

Die Extension benötigt einen Hook, über den sie erfasst, wann die folgenden Ereignisse im System stattfinden:

- Der Benutzer ist angemeldet.
- Der Benutzer ist abgemeldet.
- Der Benutzer besucht eine Seite.

Wir werden recherchieren, um herauszufinden, welche Hooks wir verwenden können.

3.4.6 Die Datenbankstruktur der Extension

Da wir mit Frontend-Benutzern arbeiten werden, verwenden wir die TYPO3-Standardtabelle `fe_users`. Wir werden diese Tabelle in keiner Weise verändern. Unsere gesamten Daten werden in anderen Tabellen gespeichert.

Zunächst müssen wir Sitzungsinformationen zu Benutzern speichern, wie Sitzungsbeginn, Sitzungsende und die Anzahl von Seitenaufrufen während dieser Sitzung. Zusätzlich speichern wir die während dieser Sitzung zuerst und zuletzt besuchte Seite.

Dann erfassen wir für jede Sitzung die Anzahl der Seitenaufrufe. Abfrageparameter für die Seite werden wir nicht berücksichtigen. Es ist wichtig, dass wir dies hier erwähnen, denn das bedeutet, dass wir uns über diese Einschränkung im Planungsstadium Gedanken gemacht haben. Abfrageparameter können den Seiteninhalt verändern, und wir entscheiden uns dafür, sie außer Acht zu lassen.

Einträge aus beiden Tabellen sollten auf derselben Seite gespeichert werden wie die Einträge der Frontend-Benutzer.

D.h., wir werden zwei Tabellen in der Datenbank haben. Wir wollen diese noch genauer planen.

Die erste Tabelle sollte die folgenden Felder haben:

Feldname	Feldtyp	Beschreibung
`uid`	`int`	TYPO3-Standardfeld, das einen Eintrag eindeutig identifiziert.
`pid`	`int`	TYPO3-Standardfeld, das die Seite identifiziert, auf der sich die Einträge befinden.
`fe_user`	`int`	Dies ist die Datenbankrelation zur Tabelle `fe_users`. Wir wählen das Integer-Feld, da wir nur eine Tabelle referenzieren. Wir müssen TYPO3 explizit auffordern, den Tabellennamen *nicht* der Zahl voranzustellen.

Tabelle 3.2: Die erste Tabelle unserer Extension und ihre Felder

Feldname	Feldtyp	Beschreibung
session_start	int	Dies ist der Unix-Timestamp für den Zeitpunkt, an dem die Sitzung beginnt.
session_end	int	Dies ist der Unix-Timestamp für den Zeitpunkt, an dem die Sitzung endet. Wir aktualisieren diesen Wert jedes Mal, wenn der Benutzer eine Seite aufruft. Der letzte Wert ist daher automatisch der Zeitpunkt des Sitzungsendes.
hits	int	Die Anzahl von Seitenaufrufen in dieser Sitzung.
first_page	int	Dies ist die Datenbank, die zur pages-Tabelle gehört. Sie zeigt die Seiten-ID, bei der die Sitzung begann (bei Anmeldung des Benutzers).
last_page	int	Dies ist die Datenbank, die zur pages-Tabelle gehört. Sie zeigt die Seiten-ID, bei der die Sitzung endete (bei Abmeldung des Benutzers oder Schließen des Browsers). Sie wird in der gleichen Weise aktualisiert wie das Feld session_end.

Tabelle 3.2: Die erste Tabelle unserer Extension und ihre Felder (Forts.)

Die zweite Tabelle wird folgende Felder haben:

Feldname	Feldtyp	Beschreibung
uid	int	TYPO3-Standardfeld, das einen Eintrag eindeutig identifiziert.
pid	int	TYPO3-Standardfeld, das die Seite identifiziert, auf der sich die Einträge befinden.
crdate	int	Unix-Timestamp, der die Zeit der Erstellung des Eintrags anzeigt.
tstamp	int	Der Unix-Timestamp, der anzeigt, wann ein Eintrag zuletzt aktualisiert wurde. Dies mag übertrieben erscheinen (weil Statistiken normalerweise nicht geändert werden), aber dieses Feld wird uns ermöglichen, eventuelle manuelle Änderungen an den Statistiken zu erkennen.
fe_user	int	Dies ist eine Referenz auf die fe_users-Tabelle.
sesstat_uid	int	Dies ist eine Referenz auf die erste Tabelle. Da es ein int-Feld ist, werden wir TYPO3 dazu auffordern, dem Tabellennamen keine Ziffer voranzustellen. Wir könnten auch die Sitzungs-ID verwenden, aber es ist besser, eine Relation zwischen diesen beiden Feldern zu haben. TYPO3 zeigt Benutzern, wie viele Einträge aus der zweiten Tabelle zu der Sitzung gehören, die in der ersten Tabelle erfasst ist. Wir müssen dazu einen Referenzindex auf die zweite Tabelle pflegen.
page_uid	int	Dies ist eine Relation auf die pages-Tabelle. Sie bezieht sich auf eine besuchte Seite.
hits	int	Dieses Feld zeigt die Anzahl von Aufrufen für diese Seite während einer Sitzung.

Tabelle 3.3: Die zweite Tabelle unserer Extension und ihre Felder

Einige Leser mögen sich fragen, warum wir ein `hits`-Feld in beiden Tabellen haben. Könnten wir es nicht einfach nur in der zweiten Tabelle anlegen und die SQL-Funktion `SUM` dazu nutzen, einen kumulativen Wert zu gewinnen? Ja, das könnten wir. Aber die `SUM`-Funktion in SQL ist aufwändig, wenn es sehr viele Einträge gibt, aus denen auszuwählen ist. Aus diesem Grund verwenden wir ein gesondertes Feld in der ersten Tabelle, um die Anzahl der Besuche je Seite zu zählen.

Ein anderer Aspekt, mit dem wir uns beschäftigen sollten, ist der Schutz dieser Tabellen vor manuellen Änderungen durch Backend-Benutzer.

Uns ist auch bewusst, dass die gewählte Datenbankstruktur uns nicht ermöglicht, die Benutzernavigation über die Site zu verfolgen.

Die Datenbankindizes definieren wir nicht an dieser Stelle, sondern wenn wir Abfragen erstellen. Die Tabellennamen werden definiert, wenn wir die Extension generieren.

3.4.7 Dokumentation

Neben einer normalen Beschreibung der Extension sollte das zugehörige Handbuch Informationen zum Löschen des Seitencaches enthalten, um sicherzustellen, dass die Liste aktualisierte Informationen anzeigt, sobald ihr ein neuer Benutzer-Datensatz hinzugefügt wird.

3.5 Zusammenfassung

In diesem Kapitel haben Sie gesehen, wie wichtig die Planung von Extensions ist. Wir haben grundlegende Prinzipien der Extension-Planung und TYPO3-spezifische Aspekte vorgestellt und die Datenbankstruktur erläutert. Zudem haben wir unsere eigene Extension geplant, die wir in diesem Buch entwickeln werden.