Marcus Straßer

Know-how ist blau.

PHP quick & dirty

12 Praxis-Workshops für schnelles Programmieren

> Tipps und Tricks für die effizente Entwicklung in PHP

> Code wiederverwenden – aber mit System

> Strategien zur effektiven Fehlersuche

FRANZIS

Inhaltsverzeichnis

1 Einleitung

Fast überall im Leben gibt es einen Unterschied zwischen süßem Traum und bitterer Realität – so auch beim Programmieren. Der Traum eines Programmierers ist es, in Ruhe und mit nahezu unbegrenzter Zeit arbeiten zu können, genügend Zeit und Geld zu haben, um ein Projekt sorgfältig planen und es mit all dem so mühsam erlernten Wissen und Können perfekt umsetzen zu können.

Leider sieht die Realität für viele Programmierer anders aus. Zeit und Geld sind knapp. Der Kunde ist ungeduldig und nur schwer davon zu überzeugen, doch bitte sehr zwei Tage Programmierarbeit zu bezahlen, damit erst mal ein schickes Klassendiagramm erstellt werden kann. Mit viel größerer Wahrscheinlichkeit wird man den Satz hören: »Hey, Hauptsache, es ist schnell fertig und es läuft irgendwie!« Hier soll dieses Buch helfen. Es geht nicht darum, schön zu programmieren, sondern »quick & dirty«, also frei übersetzt: schnell und schmutzig. Unser Leitsatz soll dabei lauten: *So »quick« wie möglich und so »dirty« wie nötig.*

Hierbei soll »dirty« aber auf keinen Fall »chaotisch« bedeuten. Es geht niemals darum, eine Programmierung einfach schnell hinzurotzen. Ein heilloses Durcheinander, sei es nur im Quellcode oder aber bereits in der Projektplanung, mag zwar »dirty« sein, aber es ist auch völlig unübersichtlich und nervenaufreibend und damit alles andere als »quick«. Sie werden letztlich mehr Zeit damit verbringen, später noch einmal nachzuvollziehen, was Sie in Ihrer Programmierung eigentlich gemeint oder mit Ihrem Kunden vereinbart hatten, als es gekostet hätte, den Quellcode durchgängig ordentlich zu formatieren oder das Projekt von Beginn an strukturiert zu planen.

Warum aber nun gerade PHP? PHP eignet sich in geradezu idealer Weise für einen »Quick & Dirty«-Programmierstil: Es ist weit verbreitet, fast überall verfügbar, schnell zu installieren und benötigt als Entwicklungsumgebung notfalls nicht mehr als einen kleinen Texteditor. Darüber hinaus ist PHP, im Gegensatz zu schwergewichtigen Programmiersprachen wie Java oder C#, sehr genügsam und zwingt dem Programmierer nicht sonderlich viele Regeln und Gesetze auf. Das kann im Chaos enden. Es ermöglicht aber eben auch, sehr schnell an sein Ziel zu gelangen, wenn man nur versucht, effizient und nicht unbedingt immer perfekt zu programmieren.

- Dieses Buch zeigt Ihnen NICHT, wie man »schön« programmiert. Es geht hier nicht um Programme, mit denen Sie stolz beim nächsten PHP-Entwicklertreffen protzen können.
- Dieses Buch zeigt Ihnen NICHT, wie man PHP lernt. Es fängt nicht bei Null an, sondern setzt ein gewisses Grundwissen voraus.
- Dieses Buch zeigt Ihnen, wie Sie den Alltag als selbstständiger Einzelkämpfer, notorischer Chaot oder Berufseinsteiger meistern können – einen Alltag, in dem es keine Preise für eingehaltene Programmierrichtlinien oder Design-Patterns, sondern nur wütende oder zufriedene Kunden gibt.

Im Gegensatz zu einem normalen Lehrbuch gibt es hier oft kein eindeutiges Richtig oder Falsch. Es geht darum, dass Sie möglichst effizient arbeiten. Daher ist alles in diesem Buch als Vorschlag zu verstehen. Es soll die Richtung zeigen, nicht unbedingt den exakten Weg. Wenn Sie eine bessere Art und Weise für sich finden sollten, schnell und gut zu arbeiten, dann behalten Sie diese bei.

Kurz oder lang?

An dieser Stelle ein Hinweis zur Schreibweise der PHP-Quelltexte in diesem Buch: Es wird inzwischen allgemein empfohlen, die sogenannten »short open tags« für PHP zur Einleitung eines PHP-Programmteils nicht mehr zu nutzen:

```
<? echo "Hallo Welt" ?>
```

Vorzuziehen ist die »lange« Variante:

```
<?php echo "Hallo Welt"?>
```

Ich werde jedoch in sämtlichen Listings die kurze Form wählen – einfach, weil das die Quelltexte an vielen Stellen deutlich lesbarer macht. In diesem Buch geht es weniger um konkrete Quelltexte als darum, die Ideen dahinter zu verstehen. Von der kurzen Variante wird insbesondere dann abgeraten, wenn Sie innerhalb Ihrer Programme auch mit XML-Daten arbeiten. Diese verwenden Zeilen wie diese:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Und so etwas würde innerhalb einer PHP-Datei zu Fehlern führen, wenn eine solche XML-Zeile als PHP-Programmierung interpretiert würde.

Aber ganz ehrlich: Solange Sie nicht tatsächlich mit XML oder auf einem Server arbeiten, der die Verwendung der kurzen Form deaktiviert hat (was selten der Fall ist),

können Sie auch ohne schlechtes Gewissen die kurze Form benutzen. Sie ist schneller geschrieben und macht Ihren Quelltext leichter lesbar. Insbesondere können Sie auch die kurze Form für den `echo`-Befehl wählen:

```
<?="Hallo Welt" ?>
```

2 Plan & Concept

Ein »Quick & Dirty«-Projekt planen und konzipieren

Sie werden selten ein Projekt »quick & dirty« programmieren, weil Sie es wollen, sondern meist, weil Sie es müssen. Zwar kann durchaus auch eine gewisse Befriedigung daraus entspringen, ein komplettes Anmeldeformular in einer halben Stunde zum Laufen zu bekommen; wenn Sie jedoch für sich allein oder an einem großen Projekt arbeiten, Ihnen also genügend Zeit zur Verfügung steht, dann sollten Sie diesen Luxus auch unbedingt nutzen (und genießen). Für viele Programmierer ist aber gerade das nicht immer möglich: Sie arbeiten allein (als Freiberufler oder aber allein verantwortlich in einer Firma) und haben einen Kunden, der alles möglichst gestern haben will und dem es egal ist, ob schön programmiert wird, solange er nur möglichst bald etwas zum Anklicken hat.

Auf den ersten Blick erscheint es da ein wenig paradox zu behaupten, Sie sollten sich gerade deshalb genügend Zeit nehmen, um das Projekt und den Umgang mit dem Kunden zu planen. Sollten Sie nicht alle Zeit in die Programmierung stecken? Sollten Sie nicht sofort den nächsten Text-Editor öffnen und lostippen? Nein, das sollten Sie nicht.

Tipp
Unterschätzen Sie niemals die Zeit, die Sie mit Besprechungen und Abstimmungen mit dem Kunden und der Planung des Projekts verbringen werden. Gehen Sie daher bei diesen immer gezielt und strukturiert vor.

»Quick & Dirty« bedeutet nicht, vollkommen planlos irgendwie drauflos zu programmieren. Denn ein solches Vorgehen kostet auf lange Sicht *immer* Zeit – und ist damit alles andere als »quick«. Es bringt überhaupt nichts, ein Anmeldeformular in einer halben Stunde fertig zu bekommen, wenn Ihnen der Kunde später erklärt, dass eigentlich alles ganz anders sein soll und Sie noch mal ganz von vorn beginnen müssen.

Versuchen Sie daher, die Kommunikation mit dem Kunden und die Planung des Projekts durchaus straff und effizient zu gestalten, aber widerstehen Sie der (nachvollziehbaren) Versuchung, diesen Teil Ihrer Arbeit zu ignorieren oder als unwichtig anzusehen.

Das ist nicht immer leicht, denn die meisten Programmierer hassen diesen Teil ihrer Arbeit aus tiefstem Herzen, und bei jeder Minute, die in einem »Meeting« verrinnt,

denken sie nur: »Was hätte ich in dieser Zeit doch alles Schönes programmieren können?!« Leider gibt es nur wenige Programmierer, die über den Luxus verfügen, immer das tun zu können, was sie selbst wollen, und die dann auch noch jemanden haben, der sie tatsächlich dafür bezahlt. Alle anderen müssen einfach akzeptieren, dass der Umgang mit den Kunden und die Planung der anstehenden Aufgaben unverzichtbare Teile eines jeden Projekts sind.

Daher werden Sie in diesem gesamten Kapitel keine einzige Zeile PHP-Code finden, sondern es geht ausschließlich um die Planung und die Konzeption Ihres Quick & Dirty-Projekts. Damit all das nicht zu abstrakt (und damit langweilig) wird, soll es im Großteil dieses Buches um ein fiktives, aber konkretes Projekt gehen: die Entwicklung eines Weblogs, eines Blogs. An ihm sollen alle Schritte eines Quick & Dirty-Projekts gezeigt werden. Halten Sie sich aber bitte immer wieder vor Augen, dass dies kein echtes Projekt ist. Es geht nicht in erster Linie darum, die Programmierung eines Weblogs aufzuzeigen, sondern darum, die Konzepte eines effizienten Quick & Dirty-Projekts vorzustellen.

2.1 Planung mit dem Kunden

2.1.1 Bevor Sie anfangen zu programmieren

Bevor Sie auch nur ans Programmieren denken können, muss der Kunde Ihnen erst einmal klar machen, was er möchte. Zumindest in der Theorie. In der Praxis ist es meist so, dass Sie zusammen herauszufinden versuchen, was der Kunde eigentlich will. Denn oft ist ihm das, zumindest im Detail, nicht wirklich klar. Sicher, er weiß, dass er eine Shop-Lösung oder einen Blog möchte. Was dies aber genau bedeutet, was dabei schon zu Beginn des Projekts zu berücksichtigen ist, darüber hat er sich oftmals noch keine Gedanken gemacht (oder auch nicht machen können, weil ihm mit hoher Wahrscheinlichkeit das notwendige Hintergrundwissen fehlt).

Beim ersten Gespräch über das kommende Projekt sollte es daher immer darum gehen, gemeinsam zu formulieren, was die »Vision« des Projekts ist (sei es per Telefon oder bei einem persönlichen Treffen). Dieser Begriff mag etwas hochtrabend und bei einem simplen Blog etwas überdimensioniert wirken, aber es bezeichnet sehr gut, was gemeint ist: Was ist die Kernfunktionalität dessen, was entwickelt werden soll? Was ist das Ziel? Über diese Punkte hat sich der Kunde in der Regel bereits ausgiebige Gedanken gemacht, daher sollten Sie erst einmal zuhören und zu verstehen versuchen, was er meint.

Ein Hauptproblem bei der Kommunikation zwischen Kunde und Programmierer ist, dass sie in zwei verschiedenen Welten leben und daher zwei völlig unterschiedliche Sprachen sprechen, wobei beide jeweils überzeugt sind, dass sie doch eigentlich jeder

verstehen müsse. Es kann daher durchaus sein, dass der Kunde Ihnen seine Vision erklärt, ohne dabei das Wort »Blog« überhaupt in den Mund zu nehmen. Das könnte beispielsweise so klingen:

»Also, wir benötigen etwas, damit wir vor und während der Messe in zehn Tagen auf unserer Webseite den Leuten zeigen können, was so passiert. Eher ziemlich locker. So wie ein Tagebuch, verstehen Sie? Man schreibt etwas rein mit Titel und Text und so, und dann erscheint das mit Datum auf der Webseite. Sie kennen das doch, oder? Ach ja: Ein Bild sollte man noch hochladen können. Und dann wäre es natürlich klasse, wenn man das Ganze irgendwie nach Themen einteilen könnte. Und suchen soll man natürlich können. Und, ach ja, später soll der Nutzer auf unserer Webseite noch Kommentare dazu schreiben können, dann kriegen wir auch ein Feedback, und vielleicht können wir das Ganze noch als Newsletter verschicken, ginge das?«

Bewahren Sie Ruhe. Fast immer haben Kunden viele Ideen, sehr viele Ideen. Warum auch nicht? Ideen kosten nichts. Versuchen Sie daher zunächst, mit dem Kunden die eigentliche Vision zu formulieren, die hinter den Ideen steckt. Was ist die Kernfunktionalität? In diesem Fall identifizieren Sie den Wunsch des Kunden als »Blog«. Das klingt trivial, aber es ist ungemein wichtig, dass Sie und Ihr Kunde über die gleiche Sache reden! Das gilt vor allem dann, wenn es um das grundlegende Wesen der zu erstellenden Anwendung geht. Es gibt kaum etwas Schlimmeres, als wenn beide Partner von Beginn an zwei unterschiedliche Dinge meinen und fatalerweise der Überzeugung sind, dass der andere doch auch genau wissen müsse, worum es geht. Vergewissern Sie sich also, dass Sie verstehen, was der Kunde meint, und umgekehrt. Stellen Sie beispielsweise sicher, dass er einen Blog und kein Forum möchte.

In den seltensten Fällen werden Sie aus einem ersten Gespräch herausgehen und direkt losprogrammieren können. Das ist in der Regel auch gut so, denn in so einem Treffen sollten Sie sich noch nicht in kleinsten Details verlieren. Dazu reichen die Zeit und die Konzentration aller Beteiligten nicht aus. Das führt dazu, dass am Ende des Gesprächs die Details von Punkt 1.1.a geklärt sind, aber Punkt 3 noch nicht einmal angesprochen wurde.

Versuchen Sie daher als Nächstes, eine grobe Feature-Liste des Projekts zusammenzustellen. Gehen Sie dabei so vor, als ob Sie ein Buch schreiben müssten: Den Titel haben Sie bereits (»Der Blog«), jetzt klären Sie die Kapitelüberschriften. Wenn Sie diese haben, fangen Sie an, Unterkapitel zu formulieren usw. Bei unserem Blog-Projekt könnte eine erste Feature-Liste so aussehen:

Blog-Feature-Liste:

- Darstellung von Tagebuch-/Blog-Einträgen in der Webseite (sortiert nach Datum oder Thema)

- Einpflegen neuer Einträge über die Administration
- Einpflegen neuer Themen (für die Einträge) über die Administration (z. B. »Neuigkeiten«, »Wissenswertes« etc.)
- Möglichkeit für die Webnutzer, Einträge zu kommentieren (das setzt eventuell ein Anmeldeformular für die Kommentarfunktion voraus)
- Newsletter-Funktion (Inhalte des Blogs werden als Mail verschickt, dafür werden jeweils An- und Abmeldeformulare für den Newsletter benötigt)

Diese Liste muss keineswegs perfekt sein. Seien Sie sich zu jedem Zeitpunkt bewusst, dass sie sich im Laufe des Projekts noch ständig ändern wird. Einige Punkte werden größer und wichtiger, andere vielleicht ganz wegfallen. Um es noch einmal zu sagen: Entscheidend ist, dass Sie und der Kunde die gleiche Sprache sprechen. Zudem müssen Sie sich absolut sicher sein können, dass Sie tatsächlich das tun, was der Kunde am Ende haben möchte, wenn Sie mit der eigentlichen Programmierarbeit beginnen.

Bis zu diesem Zeitpunkt wird der Kunde ganz auf Ihrer Seite und froh sein, dass Sie ihn verstanden haben. Jetzt aber kommen wir zu einem Punkt, der weder Ihnen noch dem Kunden wirklich gefällt: die Zeit. Denn die Antwort auf die Frage »Wann soll all das fertig sein?« wird sehr oft lauten »So bald wie möglich« (oder, für Freunde unnötiger Abkürzungen und Anglizismen, ASAP – »as soon as possible«). Insbesondere wenn Sie als Einzelkämpfer und Freiberufler unterwegs sind, können Sie hier den mit Abstand größten Fehler begehen, indem Sie nicht versuchen, den Aufwand des Projekts realistisch einzuschätzen und dies dem Kunden auch so mitzuteilen, sondern der Angst verfallen, der Kunde könnte sich einen anderen Programmierer suchen, wenn Sie nicht jeder Deadline bedingungslos zustimmen.

Wenn Sie wissen, dass die Zeit nicht reichen wird, dann ignorieren Sie unbedingt die kleine Stimme in Ihrem Kopf, die Ihnen zuruft: »Ach komm, irgendwie kriegen wir das schon hin!« Das werden Sie nicht. Niemals. Solche Projekte enden immer im heillosen Chaos, durchwachten Nächten und (zu Recht) erzürnten Kunden, die stinksauer sind, weil es in der Programmierung nur so vor Fehlern wimmelt und das Projekt dann eben doch nicht rechtzeitig – wie versprochen – fertig ist.

Unterschätzen Sie Ihren Kunden nicht. Auch wenn Sie vielleicht manchmal das Gegenteil annehmen: Kunden sind nicht dumm. Natürlich möchten sie alles so schnell wie möglich haben (wer will das nicht?), aber sie wissen auch, dass das nicht immer geht. Ihr Kunde wird sich niemand anderen suchen wollen, wenn er den Eindruck gewinnt, dass Sie realistisch den benötigten Aufwand abschätzen können, und wenn Sie ihm Vorschläge unterbreiten, wie bis zu seiner angesetzten Deadline zumindest ein Teil des Pro-

jekts fertiggestellt werden kann, oder Alternativen nennen, die es Ihnen erlauben würden, das komplette Projekt rechtzeitig zu beenden.

Schauen Sie sich also die Feature-Liste einmal an und überschlagen Sie ganz grob den Aufwand für die einzelnen Punkte. Versuchen Sie einzuschätzen, ob all das bis zur angestrebten Deadline geschafft werden kann. In unserem Fall würden Sie sicher feststellen, dass dies nicht der Fall ist. Die Messe startet in zehn Tagen, und der Blog soll ja bereits einige Zeit davor online gehen. Sicher werden Sie im Gespräch mit dem Kunden bemerkt haben, an welchen Funktionen sein Herz hängt, und Sie werden auch einschätzen können, welche Programmierung für alle weiteren Funktionen vorhanden sein und daher als Erstes gemacht werden muss. Hier könnten Sie dem Kunden vorschlagen, dass Sie in den nächsten Tagen die Kernfunktionen des Blogs fertigstellen werden:

PHASE 1 (fertig bis eine Woche vor der Messe)

- Darstellung von Tagebuch-/Blog-Einträgen in der Webseite (sortiert nach Datum oder Thema)
- Einpflegen neuer Einträge über die Administration
- Einpflegen neuer Themen (für die Einträge) über die Administration (z. B. »Neuigkeiten«, »Wissenswertes« etc.)

Der Kunde könnte also bereits beginnen, Blog-Inhalte einzutragen und diese auf seiner Webseite zu präsentieren. Alle weiteren Anforderungen könnten Sie dann ohne Probleme in »Phase 2« fertigstellen, beispielsweise bis zum tatsächlichen Beginn der Messe.

PHASE 2 (fertig bis zur Messe)

- Möglichkeit für die Webnutzer, Einträge zu kommentieren (benötigt evtl. ein Anmeldeformular für die Kommentarfunktion)
- Newsletter-Funktion (Inhalte des Blogs werden als Mail verschickt, dafür werden jeweils An- und Abmeldeformulare für den Newsletter benötigt)

Der Kunde wird diesen Vorschlag sicher akzeptieren, wenn er nicht den Eindruck hat, dass Sie einfach zu faul und/oder unfähig sind, die Entwicklung rechtzeitig fertig zu bekommen. Selbstverständlich wird es Fälle geben, in denen der Kunde durchaus versteht, dass das alles nicht bis zum festgelegten Termin zu schaffen ist, er selbst aber keine Wahl hat. Es muss einfach bis zu einem bestimmten Datum fertig sein. Auch in diesem Fall sollten Sie ihm nicht das Blaue vom Himmel versprechen, sondern

gemeinsam überlegen, wie Sie es schaffen können, dass trotzdem alles bis zum gewünschten Termin abgeschlossen ist. Sie könnten beispielsweise noch einen zweiten Programmierer hinzuziehen.

Ganz genauso sollten Sie reagieren, wenn der Kunde etwas völlig Unrealistisches verlangt. Das ist meist gar keine böse Absicht. Vielen Menschen, die nicht in der Technikwelt zu Hause sind, ist es schlichtweg nicht klar, warum etwas, das bei Google doch ganz einfach geht, nicht auch bei ihrem eigenen Programm funktionieren sollte. Wenn Sie dem Kunden jetzt nur ein »Geht nicht« vorsetzen, wird er zu Recht stur auf seiner Position beharren. Wenn Sie ihm aber klarmachen können, dass sein Wunsch nicht realistisch umzusetzen ist, und ihm eine Alternative aufzeigen, dann wird er sich sehr schnell von seiner »Ich will das wie Google«-Idee abbringen lassen. Ein wirklich professionell arbeitender Kunde will keinen unkritischen Ja-Sager als Programmierer an seiner Seite haben, sondern jemandem, mit dem er realistisch und konstruktiv an seinem Projekt arbeiten kann.

Das gilt insbesondere, wenn Sie als Programmierer einmal anderer Meinung sein sollten. Wenn Sie ernsthaft denken, dass der Kunde mit einer Idee einen großen Fehler begeht, so weisen Sie ihn unbedingt darauf hin. Begründen Sie, warum Sie das Problem anders angehen würden und wo die Vorteile Ihrer Variante liegen. Bleibt Ihr Gegenüber jedoch auch dann noch bei seiner Lösung, so kämpfen Sie nicht weiter. Verschwenden Sie keine Energie damit, innerlich zu toben und zu wüten, weil Sie den Kunden für einen Vollidioten halten. Auch hier gilt: Der Kunde ist König! Das klingt abgegriffen, ist aber trotzdem wahr. Es geht hier nicht um Ihr Projekt, sondern um das des Kunden! Er bezahlt dafür und bestimmt somit, was zu tun ist. Sie ersparen sich eine Menge Frustration und Wut, wenn Sie diese Wahrheit einfach akzeptieren.

2.1.2 Während der Entwicklung

Der Anfang einer Programmierung ist sicher für die meisten Entwickler der Abschnitt eines Projektes, der ihnen am meisten Spaß bereitet. Sie fangen etwas Neues an und können erst einmal ohne jede Kritik die Programmierung so umsetzen, wie sie es selbst für richtig halten. Der Stress beginnt meist dann, wenn dem Kunden die erste Version der Anwendung gezeigt wird, er aber in einigen Punkten doch ganz anderer Meinung ist und einen Teil der stolz entworfenen Programmierung über den sprichwörtlichen Haufen wirft. Selbst wenn Sie sich vor dem Projekt sehr gut und detailliert abgestimmt haben, wird es immer wieder Kleinigkeiten geben, die der Kunde anders haben möchte, sei es eine Checkbox, die etwas weiter oben erscheinen soll, oder die Auswahl der Blog-Themen, die er sich dann doch anders vorgestellt hatte.

Natürlich kommen solche Änderungswünsche immer zur falschen Zeit (nämlich wenn Sie gerade mitten in der Programmierung Ihrer dreifach rekursiven Funktion stecken)

und fast immer auf dem falschen Weg, nämlich per Telefon. Versuchen Sie unbedingt, sich selbst und auch den Kunden dazu zu »erziehen«, andere Wege der Kommunikation zu finden. Gespräche am Telefon sind persönlicher als eine Mail, aber auch deutlich missverständlicher und fehleranfälliger, da Sie sie nicht noch einmal abrufen können. Das führt dazu, dass Sie sich grobe Notizen während des Gesprächs machen, dann erst einmal weiter an der Stelle arbeiten, an der Sie gerade unterbrochen wurden, einige Stunden später den Zettel wieder aufnehmen und sich fragen: »Was wollte er noch mal genau?« Das kostet Zeit und vor allem Nerven.

Besser als ein Telefongespräch ist daher eine Mail. Diese können Sie lesen, wenn Sie Zeit und Ruhe haben, und Sie haben etwas in der Hand, wenn der Kunde später doch ganz anderer Meinung ist und sich selbst nicht mehr an seine Anweisung erinnern kann (das ist meistens keine böse Absicht, aber es kommt vor). Leider leben wir in einer Welt, in der wir jeden Tag, jede Minute unzählige Mails bekommen. Und auch von Ihrem Kunden werden Sie reichlich elektronische Post erhalten, die unter Umständen gar keine konkreten Änderungen an dem Projekt betreffen. Solange Sie Ihre Mails nicht sehr sauber nach »wichtig« und »unwichtig« sortieren, werden Sie bald die entscheidenden Mails mit den Korrekturwünschen genauso verzweifelt in Ihrem Postfach suchen wie einen einzelnen Zettel in einem Papierhaufen.

Aber auch hierfür gibt es eine Lösung: ein To-do-System oder Bug-Tracking-Programm. Die Rede ist von einer Webprogrammierung, die es Ihnen und Ihrem Kunden erlaubt, neue Anforderungen und beobachtete Fehler einzustellen, zu sammeln und zu kommentieren. Das klingt alles erst einmal nach einem Werkzeug für große, komplexe Projekte und so gar nicht nach Quick & Dirty. Aber unterschätzen Sie nie die Zeit, die Sie aufbringen werden, um mit dem Kunden zu kommunizieren – in welcher Form auch immer. Ein Programm, das Fehler, Absprachen und Vorgehensweisen sammelt und zuordnet, hilft daher sehr wohl, Ihr Projekt möglichst »quick« zu machen.

Versuchen Sie es – auch wenn das Projekt noch so klein ist. In der Regel sind Kunden begeistert, wenn sie das Gefühl haben, Ideen und Anmerkungen zentral eingeben und verwalten zu können. Und Sie müssen sich an keine Telefongespräche mehr erinnern oder E-Mails suchen. Als Beispiel für eine solche »To-do-Verwaltung« soll hier kurz das Programm Flyspray vorgestellt werden: *http://flyspray.org/*

Flyspray ist eine in PHP geschriebene und leicht zu installierende Webanwendung, die MySQL als Datenbank nutzt. Es soll hier nicht näher auf die Installation und Verwendung von Flyspray eingegangen werden. Auch wenn das Programm viele Möglichkeiten bietet, so ist die Handhabung doch relativ einfach: Sie legen User an (erst einmal nur sich und Ihren Kunden), und jeder dieser Nutzer kann »To-dos« verfassen, diese mit einer Priorität und einem Status versehen und einem anderen Nutzer zuweisen.

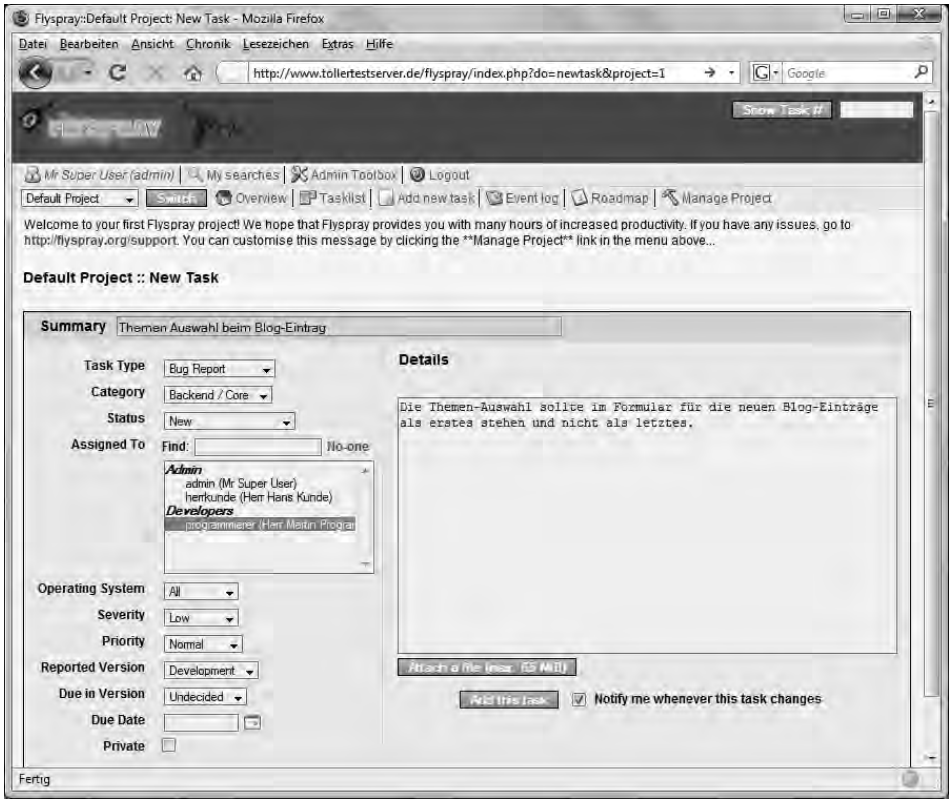


Bild 2.1: Eine Aufgabe in Flyspray verfassen

Lassen Sie sich und vor allem Ihren Kunden nicht von den zahlreichen Einstellungsmöglichkeiten verwirren. Die meisten werden Sie bei einem Quick & Dirty-Projekt nicht unbedingt brauchen. So sollten Sie beim Eintragen einer neuen Aufgabe lediglich die Felder »Summary«, »Assigned To«, »Details« und »Attach a file« ausfüllen. (Letzteres auch nur, wenn Sie beispielsweise einen Screenshot hinzufügen wollen.) Den Rest der Einstellungsmöglichkeiten können Sie einfach höflich ignorieren. Unter »Assign To« geben Sie an, für welche Nutzer diese Aufgabe bestimmt ist. Diese werden auch automatisch eine E-Mail hierzu erhalten. Sollen mehrere Nutzer benachrichtigt werden, so können Sie diese mit gedrückter STRG-Taste auswählen. Der Zuständige kann sich die Aufgabe dann jederzeit in Flyspray selbst ansehen und kommentieren:

Bild 2.2: Kommentare in Flyspray – so bleibt alles für die Nachwelt erhalten.

Auf diesen Kommentar kann der Kunde dann seinerseits antworten usw. Der Vorteil liegt auf der Hand: Sie haben nicht eine endlose Kette von E-Mails oder gar Telefonaten, sondern alle Anmerkungen und Kommentare zu einer Aufgabe gebündelt vorliegen und können diese auch zu jedem späteren Zeitpunkt wieder einsehen. Das ist nicht nur hilfreich für Sie als Programmierer, sondern auch für den Kunden, da er so stets einen

kompletten Überblick über die noch ausstehenden Aufgaben hat. Auch wenn dies bei einem größeren Projekt dann schon einmal so aussehen kann:

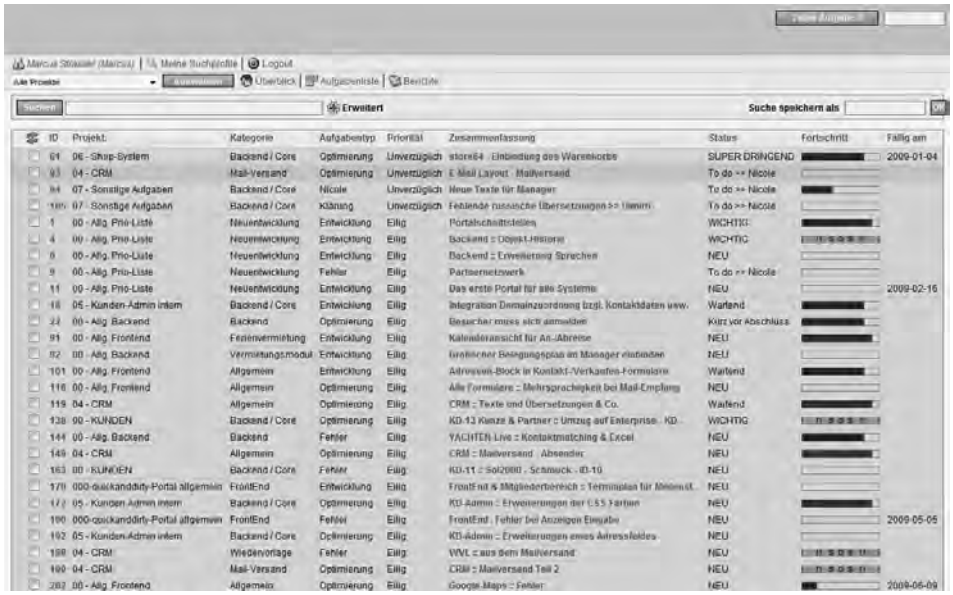


Bild 2.3: Es gibt noch eine Menge zu tun!

Eine derart gebündelte Kommunikation ist aber natürlich keine Garantie für einen jederzeit verständlichen Austausch mit Ihrem Kunden. Nicht nur zu Beginn, sondern auch während eines Projekts werden Sie Anforderungen und Ideen des Kunden immer wieder neu diskutieren und abstimmen müssen.

Wie gehen Sie aber vor, wenn es sich nicht um Fehler oder simple Änderungen handelt, sondern der Kunde plötzlich ganz neue Anforderungen an Sie stellt? Wenn es nur vereinzelte Kleinigkeiten sind, dann sollten Sie nicht weiter darüber nachdenken. Machen Sie es einfach. Sie werden sonst mehr Zeit damit verbringen, mit dem Kunden darüber zu diskutieren, ob dies Teil der ursprünglichen besprochenen Funktionen war, als es einfach schnell zu programmieren.

Aber was machen Sie, wenn es keine Kleinigkeit ist? Bevor Sie auch nur eine Zeile programmieren, weisen Sie den Kunden darauf hin, dass dies eine neue Anforderung ist und was sie für die Entwicklung des Programms bedeutet. Vielleicht ist ihm gar nicht bewusst, dass sein Wunsch so aufwendig ist. Fragen Sie, ob andere Funktionen dafür wegfallen oder an Priorität verlieren und somit in eine spätere Phase verschoben werden sollen. (Wenn Sie als Freiberufler arbeiten, müssen Sie auch darauf hinweisen, dass sich die Kosten erhöhen werden.) Wie gesagt: Kunden sind nicht dumm. Natürlich werden

sie immer wieder versuchen, kostenlos ein paar zusätzliche »Gimmicks« zu bekommen, aber kein vernünftig und professionell arbeitender Mensch wird von Ihnen verlangen, aufwendige neue Funktionen zu entwickeln, ohne dass dafür andere warten müssten.

Auch hier gilt alles, was oben gesagt wurde: Wenn ein Wunsch nicht realistisch umzusetzen ist, zeigen Sie Alternativen auf. Beginnen Sie mit der Programmierung erst, wenn Sie sich sicher sind, dass beide Seiten wirklich die gleiche Sprache sprechen und genau wissen, was gemeint ist. Wenn Sie die Idee (oder einen Teil davon) für einen Fehler halten, dann weisen Sie darauf hin.

2.1.3 Nach der Entwicklung

Die wenigsten Projekte sind wirklich beendet, wenn sie online gehen. Entweder ist noch nicht alles komplett fertiggestellt, oder es folgt direkt eine Phase 2, die an das bestehende Projekt anschließt. In jedem Fall wird es mit großer Wahrscheinlichkeit auch nach der offiziellen Deadline noch Fehlermeldungen und Änderungswünsche des Kunden geben. Wie gehen Sie damit um?

Spätestens in dieser Phase des Projekts lohnt es sich, über einen Testserver nachzudenken, also ein System, auf dem Sie dem Kunden neu implementierte Änderungen zeigen können, ohne das aktuelle Livesystem zu stören. Dieses Testsystem kann auf dem Server des Kunden liegen, oder Sie besorgen sich selbst etwas Webspace, auf dem Sie Ihrem Kunden seine Webseite präsentieren können. (In diesem Fall sollten Sie diese unbedingt durch ein Passwort schützen, denn keinem Kunden wird es gefallen, wenn er weiß, dass seine Webseite bei Ihnen einfach so zu erreichen ist.)

Aber auch so ein Testsystem wird Sie nicht davor bewahren, dass Fehler in der aktuellen Webseite auftreten: Fehler, die dem Kunden nicht gefallen und für die Sie verantwortlich sind. Niemand mag diese Situationen (der Kunde auch nicht), aber es wird zwangsläufig etwa so passieren: Ihr Kontakt mit dem Kunden war bisher herzlich, fast freundschaftlich, gemeinsam haben Sie an dem Projekt gebastelt und waren beide zu Recht stolz, als dieses schließlich online geschaltet wurde. Doch plötzlich haben Sie einen Menschen am Telefon, den Sie kaum wiedererkennen, der wütend auf Sie ist, weil plötzlich nur noch eine kryptische Fehlermeldung erscheint, wo gestern noch der Blog zu bewundern war. Alle Freundlichkeit ist dahin, und der Kunde will wissen, was passiert und wer schuld ist.

Geben Sie in diesem Fall nicht Ihrem ersten Instinkt nach: Sie leugnen alles oder – noch schlimmer – schieben alles auf jemand anderen. Es mag sein, dass Ihnen spontan eine plausible Ausrede einfällt, aber in der Regel wird der Kunde merken, dass Sie nur versuchen, sich aus Ihrer Verantwortung zu stehlen, und verständlicherweise noch wütender werden. Stehen Sie zu Ihrem Fehler. Fehler kommen vor, in jedem Bereich. Niemand kann sie völlig ausschließen. Das bedeutet nicht, dass man sich keine Gedanken darüber

machen und einfach alles tolerieren sollte, was kaputtgeht, aber niemand ist ein schlechter Programmierer, nur weil ihm ein Fehler unterlaufen ist. Sie sind dann ein schlechter Programmierer, wenn Sie das Problem von sich wegschieben.

Leider gibt es viele Menschen und damit in Ihrem Fall Kunden, denen es in so einem Fall weniger um die Behebung des eigentlichen Problems geht, sondern vor allem darum, den Schuldigen am Pranger zu sehen. Spielen Sie dieses Spiel nicht mit! Schießen Sie nicht zurück. Ducken Sie sich aber auch nicht. Das hört sich leichter an, als es ist. Versuchen Sie immer auf einer sachlichen Ebene zu bleiben oder, um es mit den berühmten Worten von Henry Ford zu sagen, suchen Sie nach Lösungen, nicht nach Problemen! Machen Sie dem Kunden klar, dass Sie sich um den Fehler kümmern und alles unternehmen werden, ihn sofort zu beheben. Sie müssen dabei nicht winselnd auf die Knie fallen und um Gnade flehen.

Wie gesagt: Fehler kommen vor, und – auch wenn Sie das so Ihrem Kunden niemals sagen sollten – nicht nur Sie sind von Ihrem Kunden abhängig, sondern auch umgekehrt. Der Kunde hat eine Webseite online gestellt, eine Webseite, die Sie programmiert haben. Auch wenn Ihr Gegenüber am Telefon Zeter und Mordio brüllt, so wird schon einiges passieren müssen, bevor er Sie vor die Tür setzt und sich einen anderen Programmierer sucht. Das bedeutet natürlich keinesfalls, dass Sie sich alles erlauben können. Es bedeutet lediglich, dass Sie sich weniger Gedanken darum machen sollten, ob Sie morgen noch diesen Job haben (und aus Angst vor dem Verlust desselben anfangen, jeden Fehler abzustreiten), sondern dass Sie sich einzig und allein auf die Lösung des Problems konzentrieren sollten.

Als Beispiel hier eine kleine Szene, wie Sie sie in ähnlicher Form sicher eines Tages erleben werden. Sie kommen gut gelaunt morgens in Ihr Büro, nur um nach wenigen Minuten einen empörten Kunden am Telefon zu haben. Wie gehen Sie damit um? So sollten Sie es NICHT machen:

Kunde: »Verdammt noch mal, was ist da los? Ich sehe auf der Seite nur noch so eine komische Fehlermeldung.«

Programmierer: »Keine Ahnung. Da sollte eigentlich alles gehen.«

Kunde: »Das kann doch wohl nicht wahr sein. Wir sind gestern online gegangen, wir haben den Newsletter rausgeschickt, und die Kunden kommen jetzt sicher alle auf die Seite. So was darf einfach nicht passieren!«

Programmierer: »Also, da muss irgendwer an dem Server rumgespielt haben. Vielleicht liegt das an den Rechten oder der Datenbank. Da habe ich ja leider keinen Einfluss drauf.«

Kunde: »Aber Sie haben das doch programmiert und getestet, oder schieben Sie da einfach Ihre Dateien auf den Server und das war's?«

Programmierer: »Äh, natürlich nicht, ich, äh, als ich das getestet habe, war noch alles okay. Das muss nachher passiert sein.«

Und so weiter. SO könnte es etwas besser ablaufen:

Kunde: »Verdammt noch mal, was ist da los? Ich sehe auf der Seite nur noch so eine komische Fehlermeldung.«

Programmierer: »Moment, ich sehe mir das mal an.«

Kunde: »Das kann doch wohl nicht wahr sein. Wir sind gestern online gegangen, wir haben den Newsletter rausgeschickt, und die Kunden kommen jetzt sicher alle auf die Seite. So was darf einfach nicht passieren!«

Programmierer: »Hm, Sie haben Recht. Also ich habe gestern noch ein Update eingespielt, aber als ich die Seite getestet habe, sah eigentlich alles okay aus. Es kann aber durchaus sein, dass es damit zu tun hat.«

Kunde: »Das hätten Sie doch sehen müssen. Wie kann denn so was passieren?«

Programmierer: »Ich kann Ihnen leider nur sagen, dass gestern noch alles funktionierte. Ich werde jetzt erst noch einmal die alte Version einspielen, damit ging ja alles, und dann schaue ich mir die neuen Dateien noch mal an. Ich melde mich dann sofort bei Ihnen.«

Was hier zum Thema Fehler beschrieben wurde, gilt prinzipiell für den gesamten Umgang mit Kunden: Seien Sie ehrlich. Mag sein, dass Viele das als eine verlorene Tugend ansehen und der Meinung sind, dass man so im harten Geschäftsalltag nicht weiterkommt, aber die Wirklichkeit sieht in der Regel anders aus.

Es ist gut möglich, dass Sie sich mit einer geschickten Ausrede in einem Einzelfall eine Menge Ärger ersparen. Aber gerade wenn Sie mit einem Kunden nicht nur eines, sondern mehrere Projekte umsetzen, wird dieser im Laufe der Zeit ein Gefühl dafür bekommen, was er von Ihren Aussagen zu halten hat. Wenn Sie bei jedem Problem von sich wegweisen, dann wird niemand mehr wissen, wann Sie dies zu Recht tun und wann Sie nur zu feige sind, die Verantwortung zu übernehmen. Wenn Sie aber immer zu Ihren Fehlern gestanden und alles versucht haben, diese zu beheben, dann wird man es Ihnen auch sofort glauben, wenn Sie einmal tatsächlich unschuldig sind.

Natürlich bedeutet das nicht, dass Sie sich in jeder Situation selbst ausliefern sollten. Wenn Sie einen Fehler finden und beheben können, bevor der Kunde etwas mitbekommt, dann lösen Sie das Problem so schnell wie möglich und halten selbstverständlich den Mund.

2.2 Planung für sich selbst

Der Umgang mit dem Kunden ist die eine Seite der Projektplanung. Die andere ist die Art und Weise, wie Sie an Ihre eigene Arbeit herangehen. Drehen wir daher die Zeit noch einmal zum Beginn des Projekts zurück und schauen uns an, was Sie für sich selbst unternehmen sollten, um möglichst effizient vorgehen zu können.

Tipp
Teilen Sie Ihre Arbeit in kleine, überschaubare Abschnitte ein und arbeiten Sie diese nacheinander ab. Konzentrieren Sie sich dabei immer nur auf einen einzelnen dieser Abschnitte und niemals auf alle gleichzeitig.

Ein Begriff aus dem wirklich netten Film »What about Bob?« mit Bill Murray beschreibt sehr schön die beste Vorgehensweise für Ihre eigene interne Projektplanung: »Baby steps«, also »Babyschritte«. Teilen Sie Ihre Arbeit in möglichst kleine Schritte ein, sodass Sie zu jedem Zeitpunkt an einem einzelnen dieser Schritte arbeiten können, ohne an die weiteren denken zu müssen, Schritte, die gerade so groß sind, dass Sie sie vollkommen überblicken und begreifen können.

2.2.1 Bevor Sie anfangen zu programmieren

Halten Sie sich noch einmal das Beispiel mit dem Schreiben des Buches vor Augen: Zusammen mit Ihrem Kunden haben Sie im ersten Gespräch die einzelnen Kapitel festgelegt. Jetzt liegt es an Ihnen, die beste Reihenfolge festzulegen, in der Sie diese Kapitel angehen. Betrachten Sie hierzu die jeweils aktuell umzusetzende Phase: Wie hängen die einzelnen Punkte und Bereiche zusammen? Welche Voraussetzungen und Beziehungen gibt es? Auch daraus müssen Sie keine große Sache machen, es soll Ihnen nur helfen, das Projekt einmal als Ganzes zu betrachten. Nehmen Sie sich einen Zettel und erstellen Sie eine einfache Skizze wie diese:

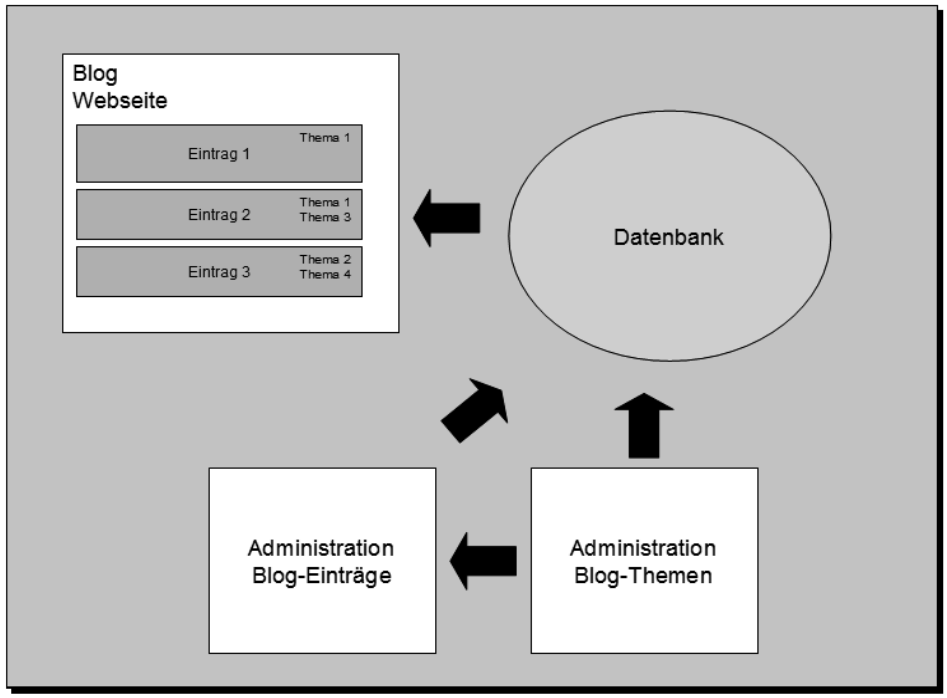
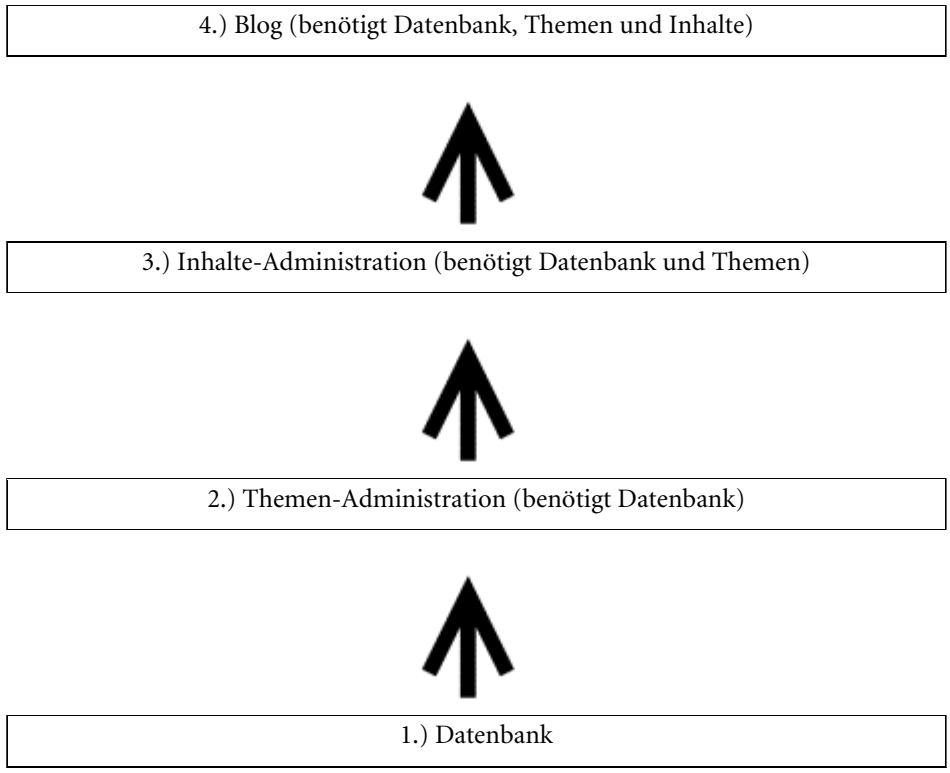


Bild 2.4: Eine grobe Skizze aller Module des Projekts.

Was sagt Ihnen das nun aber über die Reihenfolge, in der Sie die einzelnen Bereiche angehen sollten? In der Regel empfiehlt es sich, immer von unten nach oben vorzugehen, sodass, wenn Sie an einer bestimmten Stelle Ihrer Programmierung arbeiten, alle Bausteine bereits fertig sind, die Sie als Voraussetzung dafür benötigen. Was bedeutet das für das aktuelle Projekt? Schauen Sie sich Ihre Skizze an: Um überhaupt etwas auf der Webseite, dem eigentlichen Blog, anzeigen zu können, brauchen Sie eine Administration für die Blog-Einträge, um die anzuzeigenden Inhalte in die Datenbank zu bekommen. Für die Administration der Einträge benötigen Sie aber zunächst die Verwaltung der Blog-Themen, da Sie diese den Inhalten beim Erstellen ja zuordnen müssen. Und sowohl für die Administration der Inhalte als auch der Themen brauchen Sie natürlich erst einmal die Datenbank, denn sonst können Sie nichts speichern.



Wenn Sie sich an diese Reihenfolge halten, können Sie sich immer auf den aktuellen Schritt konzentrieren, da alle zuvor durchzuführenden Aufgaben bereits abgeschlossen wurden. Wenn Sie den Ablauf festgelegt haben, dann können Sie beginnen, mit dem Kunden die ersten Details zu klären. Stellen Sie ihm die Fragen, auf die Sie noch Antworten benötigen, bevor Sie wirklich mit der Programmierung loslegen können, zum Beispiel diese:

- Welche Informationen sollen für die Themen gespeichert werden?
  - Namen der Themen
  - Weitere: …
- Welche Informationen sollen für die Blog-Einträge gespeichert werden?
  - Thema
  - Titel
  - Text

  - Bild
  - Datum,
  - Weitere: …
- Kann ein Eintrag mehreren Themen zugeordnet sein?
- usw.

Sie sollten diese Fragen immer schriftlich, also per E-Mail, stellen, und wenn der Kunde nicht per Mail, sondern telefonisch antwortet, sollten Sie ihm unbedingt eine kurze, aber konkrete Zusammenfassung per Mail schicken und sich diese von ihm bestätigen lassen. Nicht nur, um sicherzustellen, dass Sie beide von derselben Sache reden, sondern auch, um später darauf verweisen zu können, was ganz genau vereinbart worden ist. Sie mögen unter der Möglichkeit, einen Text einzugeben, ein simples Eingabefeld verstehen, Ihr Kunde denkt dabei aber vielleicht an eine komplexe Benutzeroberfläche wie in einer professionellen Textverarbeitung.

2.2.2 Während der Entwicklung

Es ist meist recht einfach, ein Projekt schön zu planen, bevor es richtig losgegangen ist. Sie sind noch voller Energie und haben Zeit und Ruhe. Wenn Sie erst einmal mit der Programmierung begonnen haben und auch der Kunde immer mal wieder einen Blick auf Ihr Werk wirft, ist es meist deutlich schwieriger, sich vom Stress des Alltags nicht ins Chaos stürzen zu lassen. Das Wichtigste ist daher sicher die Einteilung der anstehenden Aufgaben in kleine, überschaubare Einheiten – die bereits angesprochenen »Babyschritte«. Sie werden sonst das konstante Gefühl haben, zusätzlich zu den Herausforderungen und Aufgaben der eigentlichen gerade anliegenden Programmierung auch noch ständig alles andere im Kopf behalten zu müssen. Das führt zwangsläufig dazu, dass Sie sich überfordert fühlen und das Projekt alles andere als effizient abläuft.

Teilen Sie sich daher das Projekt in kleine Einheiten ein, und arbeiten Sie dann Einheit für Einheit ab. Bleiben Sie beim jeweils aktuellen Abschnitt und nur dort. Natürlich können und sollten Sie die restlichen Bereiche des Projekt nicht völlig vergessen – immerhin hängt alles irgendwie zusammen –, aber wenn Sie an dem Formular für die Blog-Themen arbeiten, dann reicht es völlig aus zu wissen, dass es später noch ein Formular für die Inhalte und schließlich die eigentliche Webseite geben wird. An die konkrete Ausführung dieser Abschnitte brauchen Sie erst einmal noch nicht zu denken. Es ist in der Tat wie beim Schreiben eines Buches: Wenn Sie an einem Kapitel arbeiten, dann müssen Sie natürlich wissen, welche Kapitel noch kommen werden. Was dort exakt geschrieben stehen wird, sollte Sie aber erst dann interessieren, wenn Sie das jeweilige Kapitel auch tatsächlich bearbeiten.

Wie sähe so eine Einteilung in »Einheiten« bei unserem Blog-Projekt aus? Im Grunde haben Sie bereits eine erste Strukturierung vorgenommen: Sie haben das Projekt in grobe Abschnitte eingeteilt, und zwar in der Reihenfolge, wie diese immer aufeinander aufbauen:

1. Datenbank
2. Themen-Adminstration
3. Inhalte-Administration
4. Blog (Webseite)

In vielen Fällen hilft es, an diesem Punkt mit der Einteilung nicht aufzuhören. Für einen ganz konkreten Arbeitstag können und sollten Sie die Babyschritte noch viel, viel kleiner wählen. Schnappen Sie sich einen Zettel, schreiben Sie kurz jeden einzelnen Schritt auf, der zu tun ist, und genießen Sie das befriedigende Gefühl, wenn Sie hinter einen Punkt in Ihrer Liste einen Haken setzen und sich dem nächsten widmen können.

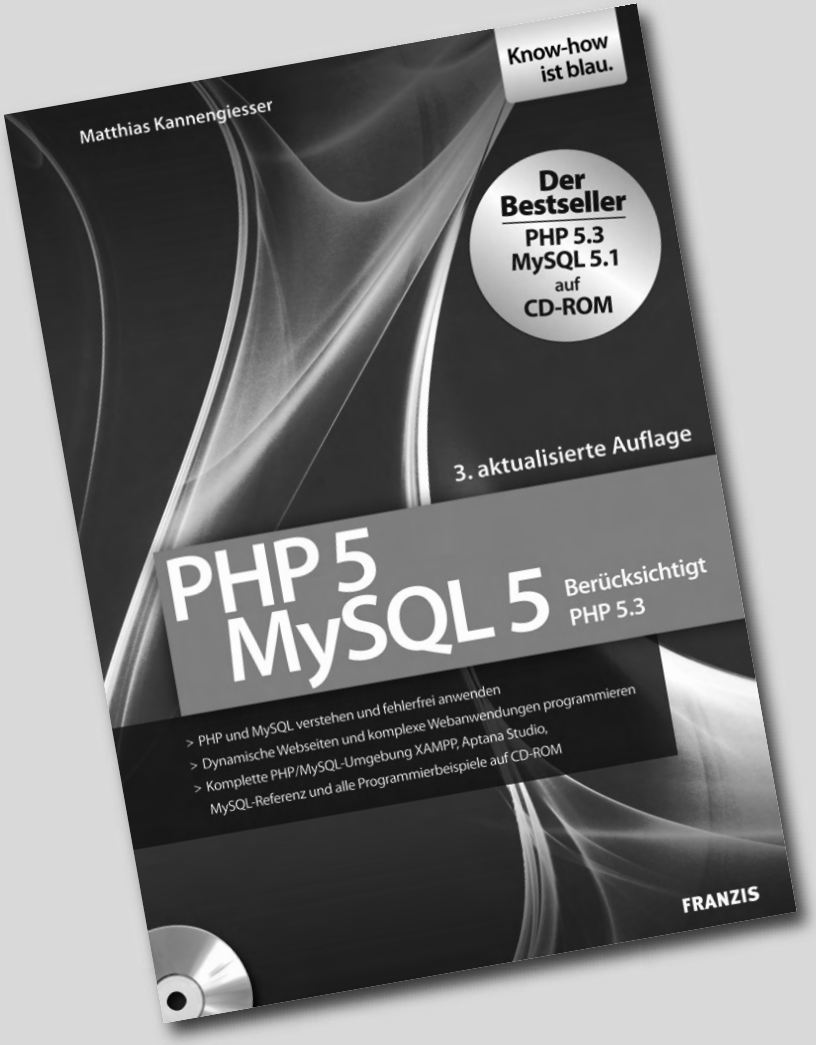
- Formular Themen ✓
- Speichern Thema in DB
- Ausgabe Liste bestehender Themen
- Themen editieren/löschen

Außerdem ist so eine Liste eine hervorragende Hilfe, um am nächsten Arbeitstag – insbesondere nach einem Wochenende – sofort wieder genau zu wissen, wo es weitergeht.

All das erscheint möglicherweise unglaublich banal, und Sie werden sich fragen, warum Sie ein Buch benötigen, um zu lernen, dass Sie sich Stichpunkte aufschreiben sollen. Machen Sie das nicht sowieso? Die Erfahrung zeigt, dass gerade diese einfachen Dinge oft in Vergessenheit geraten, wenn es stressig wird. Insbesondere bei einem Quick & Dirty-Projekt stellt dieser simple Weg – ein einfacher Zettel mit den notwendigen Schritten – die optimale Vorgehensweise dar, um in der alltäglichen Hektik einerseits nicht den Überblick zu verlieren und sich andererseits auf die gerade aktuelle Programmierung konzentrieren zu können.

Gerade erfahrene Programmierer sind der Meinung, dass sie alles schon gesehen haben und alles bereits kennen. Ein Shop ist wie der andere, ein Blog wie der nächste. Warum sich die Schritte für ein paar einfache Formulare aufschreiben? Das hat man doch schon hundert Mal gemacht, das erledigt sich alles aus dem Kopf heraus. Und genau das klappt sehr oft nicht. Und wenn es klappt, dann nur mit dem nagenden unruhigen Gefühl, trotz der vielen Erfahrung zu viel auf einmal im Blick behalten zu müssen. Machen Sie einfach »Babyschritte«. So verlieren Sie nie die Übersicht.

Stichwortverzeichnis



PHP und MySQL – das ist das bewährte Erfolgsduo für dynamische Internetanwendungen aller Art. Erfolgsautor Matthias Kannengiesser vermittelt in dieser aktualisierten Version seines Standardwerks das Profiwissen, das Sie für die anspruchsvolle Webprogrammierung brauchen. Ob objektorientierte Programmierung mit PHP, das Arbeiten mit Datenbanktabellen und Verzeichnissen oder komplexe Anwendungen wie Webshops oder Foren – Matthias Kannengiesser kennt die Lösung.

PHP 5 · MySQL 5

Matthias Kannengiesser; 2009; 744 Seiten + CD-ROM

ISBN 978-3-645-**60010-1**

€ 30,–

Besuchen Sie uns im Internet – www.franzis.de

Die ersten Schritte zur eigenen Website lassen sich mit Wordpress schnell zurücklegen. Doch wenn man eigene Vorstellungen verwirklichen möchte, wird's auf einmal knifflig. Ohne Kenntnisse der Programmiersprache PHP, in der dieses Content-System entwickelt wurde, geht es oft nicht weiter. Clemens Gull, selbst Webentwickler und Dozent, zeigt, wie's geht. Er führt auch Neulinge in PHP ein und vermittelt die notwendigen Kenntnisse, damit sie Ihre Wordpress-Seite ausbauen und verbessern können.

PHP für Wordpress

Clemens Gull; 2009; 420 Seiten

ISBN 978-3-645-**60011-8**

€ 30,–

Besuchen Sie uns im Internet – www.franzis.de

Marcus Straßer

PHP quick & dirty

Egal, ob Sie einen Shop, eine Blogsoftware oder ein bahnbrechendes neues Content-Management-System mit PHP entwickeln – Sie haben nie genug Zeit. Dieses Buch ist das richtige für Sie, wenn Sie sich als Einzelkämpfer dem Termindruck von Kundenprojekten stellen wollen oder müssen. In zwölf Praxis-Workshops beschreibt der erfahrene Webentwickler Marcus Straßer, wie Sie auch unter Zeitdruck Ihr Ziel erreichen. Anhand einer selbst entwickelten Blogsoftware zeigt der Autor, wie Sie effektiv programmieren und redundante Arbeitsgänge vermeiden.

Quick & dirty in PHP programmieren
In Kundenprojekten geht es nicht darum, einen Schönheitspreis zu gewinnen, sondern Termine einzuhalten. Das Motto dieses Buchs lautet daher: „Programmieren Sie quick & dirty" – so schnell wie möglich und so schmutzig wie nötig. Aufgrund der vielen vorgefertigten Funktionen bietet PHP ideale Voraussetzungen für diesen Programmierstil. Marcus Straßer demonstriert, wie Sie diese Features effizient nutzen, damit Sie schnell zum Erfolg kommen.

Bauen Sie sich Ihr Framework!
Die richtige Vorbereitung ist entscheidend, um in der Hektik der Entwicklungsarbeit nicht zu viel Zeit und Nerven zu verlieren. Dieses Buch zeigt Ihnen, mit welcher Ordnerstruktur Sie Ihre Dateien am besten verwalten können und wo Sie Codeteile speichern, die Sie erneut verwenden wollen – zum Beispiel für das Anlegen zusätzlicher Datenbanktabellen oder ein Skript für die Fehlersuche. So bauen Sie Ihr eigenes Framework, mit dem Sie für künftige Projekte gut gerüstet sind.

Sicherheit mit Augenmaß
Wenn es im Entwicklungsalltag hektisch zugeht, wird an der Sicherheit einer Webanwendung oft als Erstes gespart. Der Autor zeigt, dass sich schon mit der richtigen PHP-Konfiguration ein hohes Maß an Sicherheit herstellen lässt. Zudem demonstriert er, wie Sie mit einfachen Mitteln Cross-Site-Scripting und SQL-Injections verhindern.

Copy & Paste – aber mit Köpfchen
Durch die Wiederverwendung von Code lässt sich viel Zeit sparen – wenn Sie richtig vorgehen. Der Autor demonstriert, worauf Sie achten müssen, damit Sie beim Kopieren keine Fehler machen – und die eingesparte Zeit nicht für die Fehlersuche draufgeht.

Fehler suchen und beseitigen
Gerade unter Zeitdruck ist die Fehlerbehebung eine der zermürbendsten Aufgaben für Programmierer. Marcus Straßer gibt Tipps, wie Sie in Ihren Code Helfer einbauen, die Ihnen früh anzeigen, wo die Probleme liegen, wenn das Programm nicht wunschgemäß läuft. Darüber hinaus verrät er Strategien, wie Sie sich im laufenden Betrieb Ihrer Entwicklung über Fehler informieren lassen können – auch dann, wenn das Programm schon auf dem Live-Server läuft und der Kunde die Debugging-Funktion abgeschaltet hat: Lassen Sie sich in diesem Fall die Fehlermeldungen einfach per E-Mail nach Hause schicken!

Aus dem Inhalt:

Über den Autor:

Marcus Straßer ist Diplom-Physiker und arbeitet seit zehn Jahren als Softwareentwickler. Er beschäftigt sich hauptsächlich mit der Entwicklung webbasierter Anwendungen, der Datenbankprogrammierung und dem Webdesign. Marcus Straßer war für namhafte Unternehmen als Entwickler tätig und ist seit zwei Jahren Freiberufler. Er lebt in Essen.

Auf www.buch.cd

Die „Quick & Dirty"-Blogsoftware, das vollständige Beispielprojekt aus dem Buch

9 783772 364693

30,– EUR [D]
ISBN 978-3-7723-6469-3