

PHPmagazin

# PHPmagazin

Deutschland 9,80 €

Österreich 10,80 € | Schweiz 19,20 sFr  
Niederlande 11,25 € | Luxemburg 11,25 €

**PHP • JavaScript • Open Web Technologies**

## PHP Frameworks

### Die Nutzung in der Praxis

#### CakePHP

Vor- und Nachteile im Fokus

#### Das Web für die Hosentasche

Client- und serverseitige Web-Apps

Daten streamen

Vom Client bis zur Datenbank

low.js

Die kleine Alternative



## Führen in agilen Organisationen

# Im Team führen

In agilen und agil werdenden Organisationen verändert sich die Arbeit der Führungskräfte. Während in klassischen Systemen prozessuale, disziplinarische und fachliche Führung zusammenfallen, splitten sich diese in agilen Organisationen häufig auf. Das erfordert eine neue Art der Zusammenarbeit der entsprechenden Personen. Führen im Team ist eine probate Antwort auf komplexe und chaotische Anforderungen.

von Judith Andresen

Selbstorganisierte Teams sind eine strukturelle Antwort auf komplexe oder chaotische Herausforderungen. In agilen Organisationen „gehören“ tägliche Entscheidungen und tägliche Arbeit den selbstorganisierten Teams [1]: „The best architectures, requirements, and

### Sieben Delegationsstufen bewusst nutzen

Die Organisationsmitglieder bestimmen in Lernzyklen, welche Delegationsstufe zur Definition von „Richtung und Leitplanken“ für die Organisation am besten ist. Führungskräfte meint in diesem Beispiel sowohl laterale wie disziplinarische Führungskräfte.

1. Verkündern: Die Führungskräfte verkünden die von ihnen formulierten Sätze.
2. Verkaufen: Die Führungskräfte entscheiden über die Formulierung und versuchen anschließend, ihre Entscheidung zu erklären und so die anderen zu überzeugen.
3. Konsultieren: Die Führungskräfte befragen die Beteiligten, diskutieren mögliche Lösungen und entscheiden sich dann für eine Formulierung.
4. Vereinbaren: Die Beteiligten vereinbaren gemeinsam die Formulierung nach entsprechender Auseinandersetzung.
5. Beraten: Die Teammitglieder entscheiden über die Formulierung, nachdem sie die Führungskräfte gehört haben.
6. Erkundigen: Die selbstorganisierten Teammitglieder entscheiden über die Formulierung. Die Führungskräfte geben diesen nach der Entscheidung Feedback.
7. Abgeben: Die Teammitglieder übernehmen die Aufgabe, „Richtung und Leitplanken“ zu formulieren, vollständig. Eine ritualisierte Rückmeldung durch die Führungskräfte ist nicht vorgesehen.

designs emerge from self-organizing teams.“ Die Beteiligten handeln es aus und lernen, wer in ihrem System am besten strategische und/oder normative Entscheidungen trifft. So finden einige Organisationen Verfahren, in denen strategische Aufgaben gemeinsam bzw. selbstorganisiert in den Teams erarbeitet werden („Ausrichten an einem True North“). Andere Organisationen setzen auf Verfahren, in denen strategische Zielsetzungen vorgegeben sind, die Operationalisierung aber geteilt wird (OKRs, Objectives and Key Results).

Damit Teams sich selbst organisieren können, brauchen sie Richtung. Mit „Mission und Constraints“ können die Beteiligten ihre täglichen Entscheidungen und ihre tägliche Arbeit gut im Team erledigen, ohne Absicherung und/oder Entscheidungen „von außen“ einzufordern.

Wie im Artikel „Ins Lernen begleiten“ [2] ist eine Führungsaufgabe, für „Richtung und Leitplanken“ zu sorgen, um die selbstorganisierte Teamarbeit zu ermöglichen. Für „Richtung und Leitplanken“ zu sorgen, heißt nicht, diese immer zu setzen. Laterale wie disziplinarische Führungskräfte stellen sicher, dass die selbstorganisierten Teams mit diesen arbeiten können. Sie finden mit geeigneten Lernzyklen heraus, welche Art der Beteiligung für die Organisationsmitglieder besonders hilfreich ist, um „Richtung und Leitplanken“ zu setzen. Folgt man Jurgen Appelo, können bis zu sieben Delegations- (hier: Entscheidungs-)stufen (Kasten) genutzt werden.

Das Entwickeln der notwendigen „Richtungen und Leitplanken“ ist ein lernender iterativer und inkrementeller Prozess. Keine Organisation wird aus dem Stand heraus die für sie zielführenden Instrumente durch langes Nachdenken, auch nicht durch kluges Kopieren

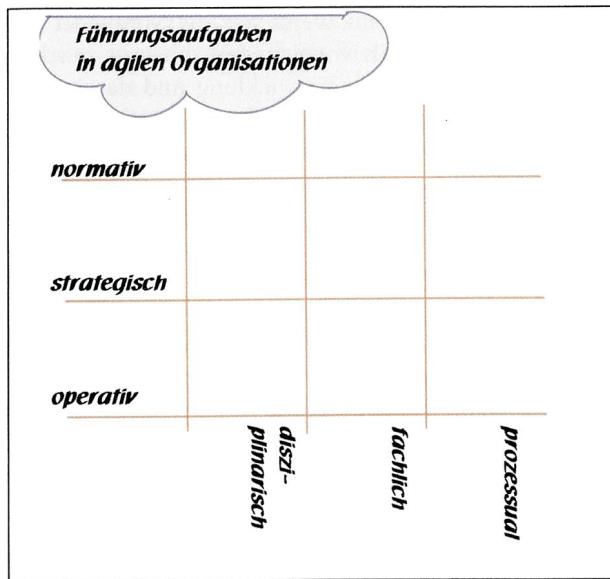


Abb. 1: Führungsaufgaben lassen sich in unterschiedliche Führungsfunktionen aufteilen

anderer Verfahrensweisen in anderen Organisationen, finden. Jede Organisation findet die für sie passenden Führungsinstrumente. In klassischen Systemen übernehmen Führungskräfte die Verantwortung häufig gleichzeitig in disziplinarischer, fachlicher und prozessueller Hinsicht. In agilen Systemen teilen sich diese Führungsfunktionen häufig auf (Abb. 1). So werden zum Beispiel die operativen prozessualen Themen vom Entwicklungsteam verantwortet, während die operative fachliche Verantwortung im Scrum-Team liegt. Die strategische fachliche Verantwortung liegt in Scrum bei den Product Owners. Dieses Beispiel illustriert, dass Führung in agilen Systemen anders funktioniert als in klassischen Systemen.

Führungsfunktionen werden entweder über bestimmte Prozesse, Personen oder Personengruppen repräsentiert. Entsprechend gibt es in agilen Organisationen kein Organigramm mit Rollenbeschreibungen, aus denen Rechte und Pflichten der Beteiligten hervorgehen. In agilen Organisationen herrscht Klarheit über die ausgehandelten Delegationsgrade. Verantwortung und Entscheidungen sind nicht bestimmten Rollen zugeordnet, sondern werden von Menschen im System erfüllt. Jeder dieser Menschen, der Führungsaufgaben übernimmt, lässt sich von der Frage leiten: „Was kann ich tun, damit das Team zur Höchstleistung kommt?“ Höchstleistung im Sinne der dienenden Leitung meint nicht Arbeitsverdichtung und Druck, sondern das Schaffen eines selbstorganisierten Teams in gleichmäßiger Geschwindigkeit [1]: „Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.“ Wenn die unterschiedlichen Führungsaufgaben jeweils durch unterschiedliche Prozesse, Personengruppen und/oder Personen repräsentiert werden, ergibt sich für die selbstorganisierten Teams ein sehr wirres Bild. Sie haben viele unterschiedliche Führungsimpulse zu verarbeiten.

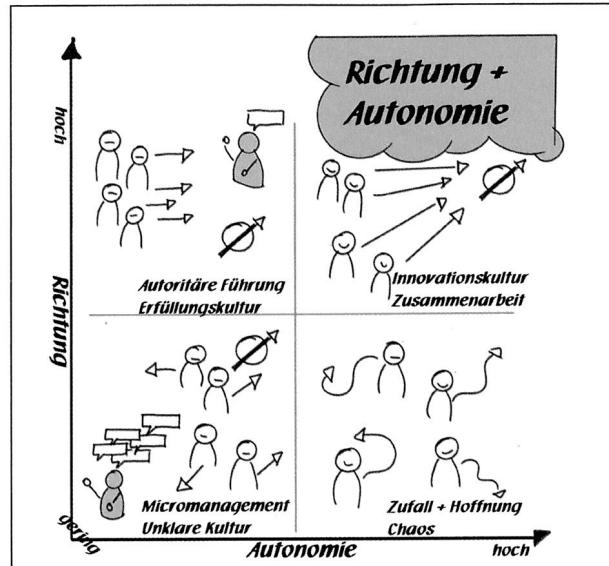


Abb. 2: Je nach Autonomie und Richtung entstehen unterschiedliche Organisationskulturen

### Teams in die Selbstorganisation begleiten

Selbstorganisierte Teams benötigen die notwendige Autonomie (Autonomy) und eine klare Ausrichtung (Alignment), um für ihre tägliche (operative) Arbeit gut zu entscheiden (Abb. 2). Henrik Kniberg stellte den Zusammenhang für selbstorganisierte Teams her. Organisationen, die in agiler Zusammenarbeit gestartet sind, müssen genauso im Wachstum sicherstellen, dass genügend Richtung und genügend Autonomie gewährt werden wie Organisationen, die agil(er) werden möchten, dies sicherstellen müssen.

Wenn die oben aufgeführten Führungsfunktionen maximal verteilt sind und von allen unterschiedliche Führungsimpulse ausgehen, entsteht im Ergebnis eine unklare Kultur, die sich trotz agiler Prozesse wie Micromanagement anfühlt. Teams werden dann eine Innovationskultur erreichen, wenn sie in hoher Autonomie ihre täglichen Entscheidungen an einer klar formulierten Ausrichtung orientieren können. Unterschiedliche Führungsimpulse führen zu keiner klaren Ausrichtung, sondern sind für die betreffenden, Selbstorganisation anstrebenden Teams maximal verwirrend. Es gilt also, die unterschiedlichen Führungsimpulse so zu formulieren, dass sie insgesamt Autonomie und Ausrichtung geben. Sofern die Führungsimpulse von Personen oder Personengruppen ausgehen, ist also dafür zu sorgen, dass diese Führungsimpulse insgesamt den selbstorganisierten Teams dienen: „Was können wir tun, damit die Teams zur Höchstleistung kommen?“

### Führungsteamrituale finden

Bisherige Führungskräfte sind es tendenziell gewohnt, für sich alleine zu arbeiten. „Oben wird die Luft dünn“ heißt es landläufig, wenn es um die Arbeit als Führungskraft geht. Diese Aussage beschreibt eine einsame Arbeit. Ähnliches lässt die in großen Unternehmen genutzte Zuschreibung der „Fürstentümer“ für unterschiedliche

Unternehmensbereiche vermuten. Für normative und strategische Entscheidungen auf prozessualer, disziplinarischer und fachlicher Ebene so zu sorgen, dass diese Selbstorganisation ermöglichen und stützen, ist ein komplexes Vorhaben. Komplexe Vorhaben lassen sich am effizientesten im Team bewerkstelligen.

So wie für die Teams, die in die agile Zusammenarbeit starten, die erste Hürde der Weg in die Teamarbeit ist [3], ist dies auch das erste große Lernmoment für Führungskräfte. Sie werden herausbekommen, wie ihre Zusammenarbeit als Führungskräfte funktioniert.

1. Was ist unser Beitrag zum Erfolg des Teams?
2. Wie lauten unser Auftrag, unsere Richtung und unsere Leitplanken?
3. Um diese Fragen zu klären, werden Führungsteams genauso wie Umsetzungsteams Lernzyklen aufsetzen, um ihre Zusammenarbeit und damit ihre Führungsleistung zu verbessern. Es sollten fünf Führungsfunktionen erfüllt sein (siehe Textkasten „Fünf Führungsfunktionen erfüllen“).

Dabei stellen den Führungsteammitgliedern die gleichen Fragen wie Umsetzungsteams, die in die agile Zusammenarbeit starten:

1. Wie funktioniert unsere Zusammenarbeit?
2. Wie kommen wir ins Liefern?

Fatal wäre es, wenn die Führungsteammitglieder versuchen, alle ihre Führungsaufgaben im Konsens zu lösen. Agile Systeme sorgen für Entscheidbarkeit von Fragen, indem sie sich jeweils auf eine Domäne fokussieren. Sie eint der Auftrag, Teams, Individuen und Organisationen

## Fünf Führungsfunktionen erfüllen

Führung in agilen Organisationen setzt auf Moderation, Delegation und Feedback, um ihre fünf Führungsfunktionen zu erfüllen:

1. Für „Ausrichtung und Leitplanken“ sorgen: Teams brauchen Ausrichtung und Leitplanken, um zielgerichtet und wirkungsvoll lernen zu können.
2. In die Selbstorganisation begleiten: Während das Team lernt, selbstorganisiert zu arbeiten und tägliche Entscheidungen zu übernehmen und zu tragen, begeben sich agile Führungskräfte in dienende Haltung: „Was kann ich tun, damit das Team zur Höchstleistung kommt?“
3. Für Entscheidungen sorgen: Agile Führungskräfte stoßen einen Klärungsprozess an, wer normative, strategische und operative Entscheidungen trifft und wie sie getroffen werden.
4. Delegieren und Entwicklung begleiten: Agile Führungskräfte sorgen für eine bewusste Aufgaben- und Verantwortungsübernahme durch alle Beteiligten.
5. Lernen des Teams und der Teammitglieder fördern: Agile Führungskräfte identifizieren (gegebenenfalls zusammen mit dem Team) weitere Lernfelder und setzen entsprechende Lernzyklen auf. Sie nutzen „Target und Track“, um die Entwicklung positiv zu begleiten.

ins Lernen zu begleiten. Wenn sich die Mitglieder des Führungsteams jeweils voneinander abhängig machen würden, würden sie die Entwicklung und das Lernen des Teams bremsen. Die Mitglieder des Führungsteams werden auf Schwierigkeiten des Teams individuell anhand der Leitfrage der dienenden Leitung antworten: „Was kann ich tun, damit das Team zur Höchstleistung kommt?“

Diese Frage beantworten sie jeweils individuell und konkret im jeweiligen Moment. Ziel ist es, die Teams in ihrer Entwicklung zu fördern, nicht auszubremsen. Die Führungsteammitglieder fördern, dass die Teams ins Handeln kommen. Die Führungsteammitglieder suchen nicht nach der einen perfekten Entscheidung für die Teams, stattdessen fördern sie operative Entscheidungen in den Teams. Sie handeln mit den Teams aus, welche Entscheidungen strategischer Natur von ihnen selbst, welche zusammen mit den Teams und welche von den Teams getroffen werden. Sie identifizieren diejenigen Entscheidungen auf normativer Ebene, die die Teams in ihrer Selbstorganisation stützen.

Die Mitglieder des Führungsteams sorgen im Sinne Fredmund Maliks für „Richtlinien und Leitplanken“, in denen sich die Selbstorganisation entfalten kann. Sie sorgen für Entscheidungen. Sie unterstützen also Teams mit Entscheidungen und/oder darin, jenseits von Konsens (schnell und sicher) entscheidungsfähig zu werden. Sie explizieren anstehende, durch das Team zu lösende Aufgaben. Sie unterstützen die Teams dabei, diese Aufgaben selbstverantwortlich und möglichst autonom zu lösen. Dabei identifizieren sie nach und nach alle operativen Aufgaben, die noch bei ihnen liegen, und transferieren Wissen und Verantwortung in die zugehörigen Teams. Alle diese Vorgänge bedürfen des Lernens durch die Beteiligten. Führungskräfte sorgen für dieses Lernen.

## In Zyklen lernen

Genauso, wie die Teams ihre Produkte iterativ, inkrementell und lernend entwickeln, erlernen Führungskräfte in agilen Organisationen ihre Führungsaufgaben in Zyklen. Das Aufdecken von Hebeln und das Ableiten notwendiger Lernzyklen können Führungskräfte für sich selbst gestalten. Mehrere Augen- und Ohrenpaare werden mehr Beobachtungen machen, sodass sich mögliche Hebel für eine erfolgreiche Zusammenarbeit besser identifizieren lassen. So kann es sein, dass sich fachliche, disziplinarische und prozessuale Führungskräfte in einer ihrer Iterationen treffen, und alle ähnliche Beobachtungen machen. Aus diesen Beobachtungen formulieren die Beteiligten Führungsimpulse, die sie gemeinsam und gleichzeitig individuell setzen werden.

Sehen wir uns Beispiele an, die zeigen, welche Schwierigkeiten in der Kommunikation im Zusammenhang mit Führungsaufgaben auftreten können.

### 1. Beispiel: Kritisches Feedback fällt schwer

Während der Iterationsüberprüfung erzählen alle Führungsteammitglieder von ähnlichen Erlebnissen rund

# Treffen Sie uns auf unseren Konferenzen!

[www.sandsmedia.com](http://www.sandsmedia.com)



ums Feedback zwischen den Teammitglieder und/oder möglichen Feedbacksituationen mit außerhalb des Teams stehenden Personen:

- „Als ich dann fragte, ob das Teammitglied bereits sein Feedback gegenüber der Person XYZ geäußert habe, sagte das Teammitglied, dass das Feedback nun auch nicht so wichtig sei und er wirklich wichtigere Dinge erst tun müsse.“
- „Als ich verblüfft nachfragte, ob ihr Ansinnen tatsächlich sei, dass ich ihr Feedback als „Feedback über Dritte“ weitertragen sollte, sagte sie, dass es doch meine Aufgabe sei, Impedimente zu mindern.“
- „Ich saß im Review. Ein Teammitglied betonte, dass es mit der Entscheidung des Teams nicht einverstanden gewesen sei. Im Grunde bat er mich um fachliche Hilfe. In den direkten Konflikt mit dem Team schien er vorher nicht gegangen zu sein“
- „Was mir auch im Review auffiel: Ein Teammitglied formulierte mir gegenüber den Vorwurf, dass ich in einer bestimmten Entscheidung nicht genügend unterstützt hätte. Diese Meinung schien geteilt zu werden. Auf meine Frage, was es gebraucht hätte, dass ich dieses Feedback früher bekommen hätte, bekam ich keine direkte Antwort. Das Team wollte die Frage mitnehmen.“

Die Führungsteammitglieder sind sich einig, dass die Fähigkeit, „Feedback zu geben und zu nehmen“, das selbstorganisierte Team voranbringen wird. Die prozessuale Führungskraft wird sich fachlich mögliche Unterstützungsformen (wie z. B. Trainings oder andere Interventionen) überlegen. Diese sind nicht im Detail mit den anderen Teammitgliedern abgestimmt. Alle Führungskräfte werden in ihren Gesprächen mit dem Team und/oder einzelnen Teammitgliedern klares Feedback geben und einfordern. Sie werden alle darauf hinweisen, wenn aus ihrer Sicht Feedbacksituationen vorliegen, und daraufhin Unterstützung beim Formulieren von Feedback anbieten und das notwendige Feedback einfordern, sodass die Teammitglieder ins Handeln kommen.

Den Paradigmenwechsel zu gehen, bedeutet für alle Beteiligten, neue Erfahrungen zu machen. Für die Teammitglieder bedeutet dies im konkreten Fall: mit jedem direkt gegebenen oder erhaltenen Feedback werden die Teammitglieder lernen, souverän und sicher Feedback zu geben und zu nehmen. Die Führungskräfte in der agilen Transition lernen, nicht für die Teammitglieder Feedback zu geben und Entscheidungen herbeizuführen. Sie lernen stattdessen, wie sie das Lernen für die Teammitglieder und für sich selbst ermöglichen können. In der kommenden Iteration werden die im Beispiel genannten Führungskräfte weitere Beobachtungen teilen und festlegen, ob „Feedback geben und nehmen“ weiterhin ein Fokus ihre Führungsarbeit sein wird – oder ob diese Fähigkeit von allen Beteiligten in einem ausreichenden Maß beherrscht wird.

## 2. Beispiel: Entscheidungen haken

Die Führungsteammitglieder teilen Beobachtungen der letzten Iteration. Jedes der Teammitglieder hat individuell in der jeweiligen Situation Feedback gegeben:

- „Und dann rief mich das Teammitglied an. Auf meine Frage, was es genau von mir brauchte, hatte es keine Antwort. Im Austausch dazu stellte sich heraus, dass das Teammitglied sich eigentlich nicht vorstellen konnte, dass diese Art von Entscheidungen vom Team getroffen wird. Daher suchte es nach einer impliziten Entscheidung durch mich. Wir hatten das in der Sprachfähigkeit.“
- „Oh, jetzt, wo Du das sagst. Das ist mir original auch gestern passiert. Aber dummerweise habe ich auf die eigentlich nicht gestellte Frage eine für



### BASTA!

23. – 27.09.2019 | Mainz

[www.basta.net](http://www.basta.net)



### Serverless Architecture Conference

14. – 16.10.2019 | Berlin

[www.serverless-architecture.io](http://www.serverless-architecture.io)



### API Conference

14. – 16.10.2019 | Berlin

[www.apiconference.net](http://www.apiconference.net)



### Int. PHP Conference

21. – 25.10.2019 | München

[www.phpconference.com](http://www.phpconference.com)



### Int. JavaScript Conference

21. – 25.10.2019 | München

[www.javascript-conference.com](http://www.javascript-conference.com)



### W-JAX

04. – 08.11.2019 | München

[www.jax.de](http://www.jax.de)



### Blockchain Technology Conference

11. – 13.11.2019 | Berlin

[www.blockchainconf.net](http://www.blockchainconf.net)



### DevOps Conference

02. – 05.12.2019 | München

[www.devopsconference.de](http://www.devopsconference.de)



### ML Conference

09. – 11.12.2019 | Berlin

[www.mlconference.ai](http://www.mlconference.ai)



### Voice Conference

09. – 11.12.2019 | Berlin

[www.voicecon.net](http://www.voicecon.net)

## Führen in einem komplexen Umfeld ist eine Aufgabe, die am effizientesten im Team erfüllt wird.

mich passende Antwort gegeben. Mist, da habe ich ja diese Unsicherheit in Bezug auf die Selbstorganisation genau gefüttert.“

- „Mir fiel auf, dass dasselbe Teammitglied in einer Konfliktsituation mit einem anderen Teammitglied nicht zu einem Ergebnis kam. Zunächst wollten sie das Thema über mich entscheiden lassen, und als ich darauf hinwies, dass diese Entscheidung operativ sei, haben die beiden sich entschieden, dass Thema bis zum nächsten Teammeeting zu vertagen, um es dort von allen entscheiden zu lassen.“
- „Im letzten Review ist mir aufgefallen, dass dieses Teammitglied zum Teil aggressiv auf Ergebnisse reagierte. Es war unzufrieden damit. Es mahnte mehrfach an, die Kommunikation müsse sich verbessern. Es zeigte sich sowohl verbal als nonverbal aggressiv gegenüber allen anderen Teammitgliedern.“

Die Führungsteammitglieder teilen ihre Beobachtungen. Sie kommen zu der Interpretation, dass es den Beteiligten schwerfällt, mögliche Meinungsvielfalt zu äußern und sich dann auf eine Entscheidung zu einigen. Gleichzeitig scheinen nicht allen Beteiligten die Delegationsgrade innerhalb des Teams vertraut zu sein. Die Führungsteammitglieder einigen sich darauf, dass sie verstärkt bei allen Anfragen darauf achten werden, ob Teams oder Teammitglieder den Grad ihrer Autonomie indirekt in Frage stellen. Sie vereinbaren individuelle Strategien für sich, wie sie diese Anfragen nicht direkt lösen und so die Autonomie und damit die Selbstorganisation des Teams fördern. Sie nehmen sich vor, jede Anfrage für sich im jeweiligen Moment zu prüfen und angemessen zu reagieren. Ihnen ist klar, dass es nicht jedes Mal funktionieren wird. Sie nehmen sich vor, Rückdelegation bei der nächsten Retrospektive erneut zu beleuchten.

Darüber hinaus nehmen sich die Beteiligten vor, bei allen Meinungsverschiedenheiten, die sie beobachten, die Frage zu stellen, auf welche Weise die Beteiligten das Thema entscheiden möchten. Darüber hinaus einigen sich die Führungsteammitglieder darauf, innerhalb der nächsten Iteration in jeder Entscheidung, an der sie beteiligt sind, den Entscheidungstyp zu explizieren und – wo möglich – zu begründen. Die Führungsteammitglieder visualisieren diese Entscheidungen für sich auf Moderationskarten und hängen diese für sich gut sichtbar auf, um sich während der Iteration an die Beschlüsse und deren Vollzug zu erinnern. Sie werden die Beschlüsse am Ende der Iteration auf ihre Wirksamkeit überprüfen.

### Retrospektiven agiler Führung

Führungskräfte in agilen Organisationen setzen für ihre Themen auf allen drei Ebenen (normativ, strategisch und operativ) gesonderte Lernzyklen auf. Wichtiges Moment eines Lernzyklus ist das Lernen als solches. Um Beobachtungen teilen, interpretieren und mögliche Führungsimpulse zu formulieren, treffen die beteiligten Führungskräfte sich in Retrospektiven. Sie überprüfen in den Retrospektiven ihren Erfolg, wie gut sie Teams, Individuen und Organisationen ins Lernen begleiten konnten. Bei der Identifikation möglicher Hebel und der entstehenden Lernzyklen können die fünf Führungs-funktionen nach Malik helfen.

Führen in einem komplexen Umfeld ist eine komplexe Aufgabe, die am effizientesten im Team erfüllt wird. Jedes Führungsteam wird für sich herausfinden müssen, ob und in welcher Form Aufgaben verteilt, gemeinsam oder individuell erfüllt werden. Es ist der Weg zu erlernen, der für das jeweilige System am besten ist. Führung im Team entwickelt sich – genauso wie Selbstorganisations- und Führung im Team – lernend und Schritt für Schritt.



Die BERATUNG JUDITH ANDRESEN macht mit fünfzehn agilen Coaches echte Zusammenarbeit möglich. **Judith Andreesen** ist Organisationsentwicklerin und agile Coach, die Teams und Unternehmen bei der Einführung agilen Arbeitens, agilen Denkens und Führens begleitet. Judith hat die Fachbücher „Retrospektiven in agilen Projekten“ und „Agiles Coaching“ geschrieben.

### Links & Literatur

- [1] Prinzipien hinter dem agilen Manifest: <https://agilemanifesto.org/principles.html>
- [2] Andreesen, Judith: „Ins Lernen begleiten“, PHP Magazin 4.19
- [3] Agile Reifegrade anstreben: <https://www.judithandreesen.com/2018/01/08/agiles-coaching-agile-reifegrade-anstreben/>



## Teil 3: Agile Organisationsformen, Veränderungen in Unternehmen

# Denken hilft zwar, aber es nützt nichts

Systemisches Denken hilft dabei, über den Tellerrand hinauszusehen. Über den Tellerrand des eigenen Teams, der eigenen Abteilung, der eigenen Organisation. Gemeinsam mit anderen Disziplinen ermöglicht es Organisationen, Verhaltensmuster schneller zu erkennen und sich weiterzuentwickeln.

von Thomas Mahringer

„Organisationales Lernen“ und „moderne Organisationsformen“ kommen in Wellen immer wieder in Mode. In unseren Breiten hat besonders das Buch von Nonaka und Takeuchi [1] Verbreitung gefunden, in dem der „SEKI-Modell“-Wissenskreislauf von implizitem und explizitem Unternehmenswissen vorgestellt wurde. Nicht ganz so bekannt ist bei uns Peter Senges Buch „The Fifth Discipline“ [2], das vor allem den Aspekt des systemischen Denkens in die Organisationsentwicklung eingebracht hat.

In den letzten Jahren gab es zahlreiche Veröffentlichungen zu und viele Versuche von agilen Organisati-

onsformen. Da sie im Vergleich zu den hierarchischen Führungsstilen der meisten Unternehmen noch uto-pisch anmuten, stellt sich die Frage, wie organisationales Lernen trotz Kostendruck, Kundendruck, steigender Geschwindigkeit und mehr Mitbewerbern möglich wird.

### Artikelserie

Teil 1: Systemische Denkmuster in Organisationen

Teil 2: Systemarchetypen und skalierbares Lernen in Organisationen

**Teil 3: Agile Organisationsformen, Veränderungen in Unternehmen**

Der erste Schritt zur Veränderung ist die ehrliche Bestandsaufnahme. Dabei helfen die aus Senges Buch stammenden „Systemarchetypen“ des zweiten Teils dieser Artikelserie [3]. Neben dem systemischen Denken gibt es aber noch die vier restlichen Disziplinen, die lernende Organisationen meistern.

### Die fünf Disziplinen im Kurzüberblick

Die fünf im letzten Teil kurz angesprochenen Disziplinen sind persönliche Weiterentwicklung, Erkennen und Anpassen von mentalen Landkarten, Lernen im Team, gemeinsame Visionsfindung und als Klammer darüber das Systemische Denken. Jede dieser Disziplinen wird im Folgenden einer genaueren Betrachtung unterzogen.

### Persönliche Weiterentwicklung und persönliche Meisterschaft

Die persönliche Weiterentwicklung der Mitarbeiter ist die notwendige Grundvoraussetzung für eine erfolgreiche Veränderung in Richtung einer lernenden und agilen Organisation. Das World Economic Forum (WEF) hat im Jahr 2016 in der Studie „The Future Of Jobs“ [4] erhoben, was aufgrund der schnellen, teilweise exponentiellen Veränderung der Arbeitswelt die größten Herausforderungen für die Unternehmen sein werden. Die Hauptherausforderung schlechthin sind die sich rasch ändernden Anforderungen an die Skills: Egal, wie viel Prozent welcher Jobfamilien durch die Digitalisierung verschwinden werden, eines ist klar: So gut wie jeder Job wird sich verändern. Das WEF nennt es „Skill Disruption“ und schätzt, dass ein Drittel der in zwei bis drei Jahren benötigten Skills jetzt noch nicht als so wichtig angesehen werden. Zu diesen Skills zählen z. B. kognitive Fähigkeiten (logisches Denken, Problemorientierung, kognitive Flexibilität, Kreativität), komplexes Problemlösen, systemische Fähigkeiten (Urteilsvermögen, systemisches Denken), soziale Kompetenz, Ressourcenmanagement (Zeit, Geld, Aufgaben). Kunden erwarten diese Fähigkeiten von ihren Ansprechpartnern. Selbst wenn es z. B. „nur“ um eine Entwicklertätigkeit geht: Kunden haben nicht mehr die Zeit, ausführliche Pflichtenhefte zu schreiben, die außerdem in wenigen Monaten wieder überholt wären. Es wird erwartet, dass sich Entwickler schnell in die Prozesse der Kunden hineindenken, auf Augenhöhe mit Kunden sprechen und mit ihnen gegebenenfalls auch diskutieren können. Dass Entwickler und Softwarearchitekten ihr Handwerk und ihre Werkzeuge beherrschen, ist Voraussetzung.

Warum wird in Unternehmen aber noch relativ wenig für diese Fertigkeiten und Fähigkeiten der Mitarbeiter getan? Wenn man genau hinsieht, bemerkt man, dass Unternehmen zwar in die Weiterbildung der Mitarbeiter investieren, es sich dabei aber um sehr klassische Schulungsmaßnahmen handelt: den nächsten Scrum-Kurs oder das x-te Kommunikationsseminar oder eine Vertriebsschulung und so weiter und so fort. Was aber fehlt, ist eine ausreichende Übung und Verinnerlichung in der Praxis. Warum kommt es kaum dazu? Alle Unter-

nehmen, die nicht gerade in einer Start-up-Phase sind, kämpfen mit der operativen Hektik: Auch wenn sie noch so innovativ sein und ihre Mitarbeiter weiterbringen möchten, stecken sie in ihren Prozessen und „Value Networks“ fest: Kunden erwarten Termintreue, Qualität, steigende Effizienzen und sinkende Kosten. Investoren, Eigentümer und Shareholder haben hohe Erwartungen, und Lieferanten folgen ebenfalls ihren eingeschliffenen Prozessen.

Das alles führt in Summe dazu, dass die Mitarbeiter explizit und implizit (durch Vorbildwirkung, gelebte Werte, Team/Projektkultur usw.) trainiert werden, die Prozesse des Unternehmens möglichst gut zu unterstützen, was ja im Sinne der Profitabilität sehr wünschenswert ist. Sie werden dadurch andererseits in vielen Fällen zu „compliant“ Mitarbeitern, die tun, was man (der Chef, die Firma, die Prozesse) ihnen sagt. „Committe“ (also engagierte) Mitarbeiter, die bestehende Abläufe gern hinterfragen und Neues kreieren, werden teilweise als Störenfriede empfunden. Doch in Zeiten von schnellen Änderungen und von „Skill Disruption“ bedarf es mehr selbstverantwortlicher, intrinsisch motivierter, „committeter“ Mitarbeiter. Nur wenn das Tätigkeitsfeld gut mit der persönlichen Vision, den persönlichen Werten und Interessen übereinstimmt, werden Mitarbeiter Höchstleistungen und Innovationen bringen, statt nur Dienst nach Vorschrift zu leisten.

Der erste Meilenstein der persönlichen Meisterschaft ist daher, die eigenen Ziele, inneren Antriebe, Visionen und Werte zu kennen und daraus ein Delta und einen Weg zu definieren. Kennen Sie viele Firmen, in denen die Mitarbeiter ebendas erlernen, anstatt nur das nächste Problem zu lösen? Warum suchen viele Mitarbeiter ihre Glücksmomente in Freizeitaktivitäten? Wäre es nicht besser für Unternehmen und Mitarbeiter, wenn sie sie in den rund acht Stunden ihrer täglichen Arbeit finden?

Persönliche Meisterschaft hat nicht unbedingt etwas mit Spaß oder Genießen zu tun, sondern mit dem Auseinandersetzen mit den eigenen Stärken, aber auch den eigenen Schwächen. Ohne kreative Spannung, Reibung und Anstrengung gibt es keine Weiterentwicklung, und das führt oft zu Widerständen, z. B. in Form von offenen oder versteckten Konflikten, Zynismus, Verteidigungshaltung und Resignation. Schließlich können die Widerstände, wenn sie nicht ordentlich erkannt und gemanagt werden, zu einer Verwässerung oder gar der Aufgabe der persönlichen Vision führen. Zu lernen, dass Misserfolg oder Kritik Wege zur Verbesserung darstellen, und nicht eine persönliche Schmach, wegen der man die Flinte ins Korn wirft, bedarf persönlichen Wachstums. Die Kombination aus Glauben an die Möglichkeit, Selbstwirksamkeit und Selbstwert wächst durch die Erfahrung von überstandenen Schwierigkeiten.

### Mentale Landkarten

Das Konzept der „mentalen Landkarten“ lässt sich in vielen Kommunikations- oder Persönlichkeitsweiterentwicklungskonzepten finden. Ursprünglich wurde es von

# Durch kraftvolle, gemeinsame Visionen empfinden wir Aufgaben als weniger oder gar nicht mehr getrennt vom eigenen Selbst. Dadurch steigen Wohlbefinden und Qualität.

Alfred Korzybski in seiner „Allgemeinen Semantik“ [5] beschrieben. Er hat damit unter anderem die Transaktionsanalyse, die Kommunikationstheorie von Bateson, die systemische Therapie von Watzlawick und das neurolinguistische Programmieren beeinflusst. Es geht im Wesentlichen darum, dass wir uns bewusst werden, wie sehr unsere Ideen, Annahmen, Überzeugungen, Verallgemeinerungen und Vorurteile unsere Wahrnehmung der Wirklichkeit beeinflussen. Wir sehen nur unsere „Landkarten“, aber nicht die der anderen. Nicht nur Individuen halten mentale Landkarten aufrecht, sondern auch Teams, Abteilungen und ganze Organisationen. Mentale Landkarten ganzer Organisationen beschreiben z. B., wie sich Mitarbeiter, Kunden oder Märkte verhalten werden. Eine andere klassische mentale organisationale Landkarte ist die Reaktion auf schwierige Situationen oder Krisen: Entscheidungen werden in Krisenfällen zentralisiert und tendenziell einer starken Führung überlassen.

Der Vorteil mentaler Landkarten ist, dass sie uns eine schnelle Orientierung und Entscheidung ermöglichen. Das ist gleichzeitig auch ihr Nachteil: Sie sind Abkürzungen im Denken, die, wenn unerkannt und unaufgedeckt, als selbsterfüllende Prophezeiungen schön langsam die Kontrolle über unser Verhalten übernehmen können.

Ebenso wie bei der persönlichen Weitereinwicklung wird das Verteidigungsverhalten zu einem großen Hemmnis beim Aufdecken von hinderlichen und beim Aufbauen von neuen mentalen Landkarten.

## **Eine gemeinsame Vision**

Bei der persönlichen Meisterschaft geht es zu Beginn um die Frage, was wir wirklich wollen: Die eigene Vision und die „Purpose“ und die Mission. In einer Organisation geht es darum, nicht einen faulen Kompromiss aus hunderten oder tausenden Einzelvisionen zu finden, sondern eine Vision, die möglichst viele Mitarbeiter mitzuziehen oder sogar mitzureißen vermag. Es geht ebenfalls nicht darum, dass das Management mit externen Unternehmensberatern eine Vision designt und diese dann von oben über die Mitarbeiter ergießt. Zu oft haben wir alle schon die Plakate und PowerPoint-Präsentationen in Firmenzentralen gesehen, die solche „Design Vision Statements“ verkünden. Eine gemeinsame Vision ist etwas, zu dem sich Mitarbeiter committen (bekennen) und dem sie nicht nur compliant (nachgiebig) folgen. Intrinsische Visionen (solche, die aus der Organisation selbst entstehen) haben das Potenzial dieser Sogwirkung auf Mitarbei-

ter. Viele Visionen sind extrinsisch motiviert und formuliert, z. B. „wir wollen um 30 % wachsen“ oder „wir wollen den Marktanteil auf x erhöhen“. Sie sind von Äußerlichkeiten (Mitbewerb, Markt ...) abhängig, ändern sich daher öfters und sind ein Nährboden für Defensivverhalten. Außerdem lassen sie sich kaum mit den Visionen der Mitarbeiter übereinbringen. Welchen Mitarbeiter fasziniert schon die Aussage „95 neue Mitarbeiter eingestellt“ im Vergleich zu „mit unserem MP3-Player haben wir 50 000 Songs in unserer Hostentasche“ oder „mit unserer App verbinden wir Familien“. Durch kraftvolle, gemeinsame Visionen empfinden wir Aufgaben als weniger oder gar nicht mehr getrennt vom eigenen Selbst. Dadurch steigen Wohlbefinden und Qualität. Letztere Art von Visionen bestärkt die Experimentierfreude im Unternehmen, sie bringt engagierte, investierte Mitarbeiter hervor: Auch wenn wir noch nicht genau wissen, wie wir es tun werden, so wissen wir doch, warum und was. Wie schon Antoine de Saint-Exupéry empfohlen hat: „Wenn du ein Schiff bauen willst, dann trommle nicht Männer zusammen, um Holz zu beschaffen, Aufgaben zu vergeben und die Arbeit einzuteilen, sondern lehre sie die Sehnsucht nach dem weiten, endlosen Meer.“

Wie kommt man nun von vielen persönlichen Visionen zu einer größeren gemeinsamen, wenn sie nicht nur ein Kompromiss oder eine verordnete Vision sein soll? Die gemeinsame Vision entsteht im Laufe der Zeit aus dem Verweben der einzelnen, persönlichen Visionen. Sie wird mehr als die Summe der Teile, sie wird ein höheres Ziel, in dem sich jeder wiederfinden kann und für das sich jeder verantwortlich fühlt. Gründer von erfolgreichen Unternehmen, insbesondere von Start-ups, geben oft die Vision vor, doch durch den Drive und die Spannung des Neuen finden sie die Mitarbeiter, die die Vision ebenfalls teilen. Die gemeinsame Vision wird zu einem selbstverstärkenden Regelkreislauf: Immer mehr Leute sind involviert, dadurch wird die Vision klarer und eventuell auch umfangreicher, dadurch wiederum steigt die Begeisterung, was dazu führt, dass immer mehr Leute involviert werden.

## **Lernen im Team – die Keimzelle der Agilität**

Die Keimzelle für das organisationale Lernen ist das Team. In einem überschaubaren Team wird schnell sichtbar, ob es gut ausgerichtet oder justiert ist oder ob es in verschiedene Richtungen strebt. Verschiedene Richtungen bedeuten Energie- und Zeitverschwendungen. Gute Sportteams oder Orchester sind ausgerichtet und

justiert und kommen dadurch in einen Flowzustand, der Befriedigung und Qualität schafft.

Agile Teams können die zuvor genannten Disziplinen in einem relativ gut geschützten und lernfreundlichen Rahmen besonders gut üben. Sie verfügen per Definition über die Methoden zur gemeinsamen Ausrichtung, z. B. über Daily Stand-ups.

Die ehrliche Selbsteinschätzung der aktuellen Skills und Erfahrungen fällt in einem agilen Team leichter. Eigene Schwächen sowie Verbesserungswegs lassen sich angstfreier artikulieren. Mit jedem Teilziel, jeder Iteration oder jedem Sprint sehen die Mitglieder Lernerfolge. Wenn etwas schiefläuft, ist nicht ein ganzes Projekt gefährdet, sondern man kann korrigieren und aus seinen Fehlern lernen. Defensive Verhaltensmuster werden immer unnötiger, weil sich Teamkollegen einander immer besser kennenlernen. Die persönliche Weiterentwicklung erfolgt daher laufend „on the Job“ statt in darauf ausgelegten Seminaren und Workshops.

Verschiedene Kommunikationswerkzeuge wie Diskussion, Dialog und kollegiales Coaching helfen dabei, mentale Landkarten aufzudecken und abzugleichen. Neue Landkarten für neue Situationen und eine veränderte Umwelt lassen sich schnell generieren. Eine weitere gute Methode zur Aufdeckung ist das Aufschreiben und Vergleichen der „offiziellen“ Landkarten/Annahmen und der dabei gedachten oder gefühlten: Nach Teambesprechungen (z. B. Sprint Planning oder Sprint Review) schreibt jeder für sich die beim Gespräch geäußerten und sichtbaren mentalen Landkarten auf, daneben die, die er bei sich wirklich gedacht hat. Je nach Vertrauensverhältnis können die Ergebnisse als persönliches Aha-Erlebnis oder als Teamlernerfahrung genutzt werden.

Aus den persönlichen Visionen lässt sich einfacher eine gemeinsame Teamvision ableiten als eine Organisationsvision. Die Bedeutung und Detaillierung der Teamvision ist und wird mit jeder Iteration immer klarer.

Systemisches Denken wird durch die Crossfunktionalität von agilen Teams fast schon vorgegeben. Die organisationalen „Lernschwächen“ aus systemischer Sicht, die wir im ersten Artikel beschrieben haben, werden minimiert:

- Dem Silodenken wird durch Crossfunktionalität entgegengewirkt.
- Reaktivität und „der Feind ist da draußen“ werden durch hohe Selbstverantwortung reduziert.
- „Der langsam gekochte Frosch“ oder sich aufstauende Probleme fallen durch laufende Messungen und Kennzahlen pro Iteration auf.
- Die „falsche Erfahrung“ (man macht etwas hunderte Male und meint, dass man es gut kann) wird ebenfalls durch kurze Iterationen und Ergebniskontrolle eingeschränkt.
- Statt des „Mythos des Managementteams“ („der Boss wird's schon richten“) zählen Selbstorganisation und Selbstverantwortung.

## Die fünfte Disziplin

Das systemische Denken bildet als fünfte Disziplin eine Klammer über die anderen Disziplinen. Sie ergänzen und durchdringen einander. Wenn man die „Sprache“ der Regelkreise und der Systemarchetypen regelmäßig übt, fällt es leichter, bestimmte Muster zu erkennen – sei es bei operativen Prozessen, organisationalen oder persönlichen mentalen Landkarten, strategischer Ausrichtung und Visionsfindung oder bei den Lernprozessen einer Organisation.

## Vom Team zur Organisation

Viele Unternehmen haben agile Prinzipien auf die gesamte Organisation übertragen. Bekannte Beispiele dafür sind Spotify mit dem Ansatz der „Soziokratie“, Zappos mit der „Holokratie“ oder die Haufe-umantis AG als „demokratisches Unternehmen“. Aber auch, wenn nicht gleich die ganze Organisation mitmacht, lassen sich agile Teams und Abteilungen als Bottom-up-Lernlabors für zukünftige Wissens- und Lernorganisationen nutzen. 2017 formulierte es der CIO der BMW-Group-IT folgendermaßen: Perspektivisch sollen bald alle 4 500 Mitarbeiter in der BMW-Group-IT agil arbeiten – auch solche, die in den Infrastrukturbereichen beheimatet sind und selbst nichts entwickeln.“ [6]



**Thomas Mahringer** ist in der Lösungsberatung und im Lösungsvertrieb tätig. Als gelernter Informatiker hat er im Lauf seiner zweidundzwanzigjährigen Berufslaufbahn bei verschiedenen – auch internationalen – Projekten Software entwickelt, Kunden beraten, Projekte und Teams geleitet, Application Performance analysiert und optimiert, Presales- und Salesstrukturen aufgebaut sowie Partnermanagement- und Wissenstransferkonzepte erarbeitet. Aus dieser Erfahrung ergibt sich seine Leidenschaft, Kunden bei der effizienten Einführung von zeitgemäßen und wertschöpfenden Lösungen zu unterstützen.

## Links & Literatur

- [1] Nonaka, Ikujiro; Takeuchi, Hirotaka: „Die Organisation des Wissens. Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen“, Campus Verlag, 1997
- [2] Senge, Peter M.: „The Fifth Discipline“, Penguin Random House, 1990
- [3] Mahringer, Thomas: „Typisch Arche: Teil 2: Systemarchetypen und skalierbares Lernen in Organisationen“ in: Windows Developer 2.2019
- [4] World Economic Forum: „The Future of Jobs“, 2016
- [5] Korzybski, Alfred: „Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics“, Institute of General Semantics, 1933
- [6] Straub, Klaus: „BMW-CIO hält Bimodal IT für einen Irrweg“, IDG Business Media GmbH

Laravel, Zend, Symfony, CodeIgniter und Yii im Vergleich

# Die fünf besten PHP Frameworks

Ein PHP Framework erleichtert die Programmierung von wiederkehrenden Funktionen und vereinfacht das Programmieren skalierbarer und leistungsstarker Anwendungen. Darüber hinaus bieten moderne Frameworks ein hohes Level beim Thema Sicherheit. Fünf beliebte und weit verbreitete PHP Frameworks werden in der folgenden Übersicht vorgestellt.

von Sascha Thattil

Der Artikel ist mit „Die fünf besten PHP Frameworks“ betitelt. Natürlich ist eine solche Aussage erst einmal subjektiv zu betrachten. Fest steht jedoch: Die hier vorgestellten Frameworks Laravel, Zend, Symfony, CodeIgniter und Yii gehören zu den führenden PHP Frameworks der letzten Jahre. Diese Frameworks bieten eine professionelle Ausstattung für PHP-Entwickler und verfügen über eine weltweite Community. Sie haben alle ihre Vor- und Nachteile, die ich gerne vorstellen möchte. Los geht es mit dem PHP Framework Laravel.

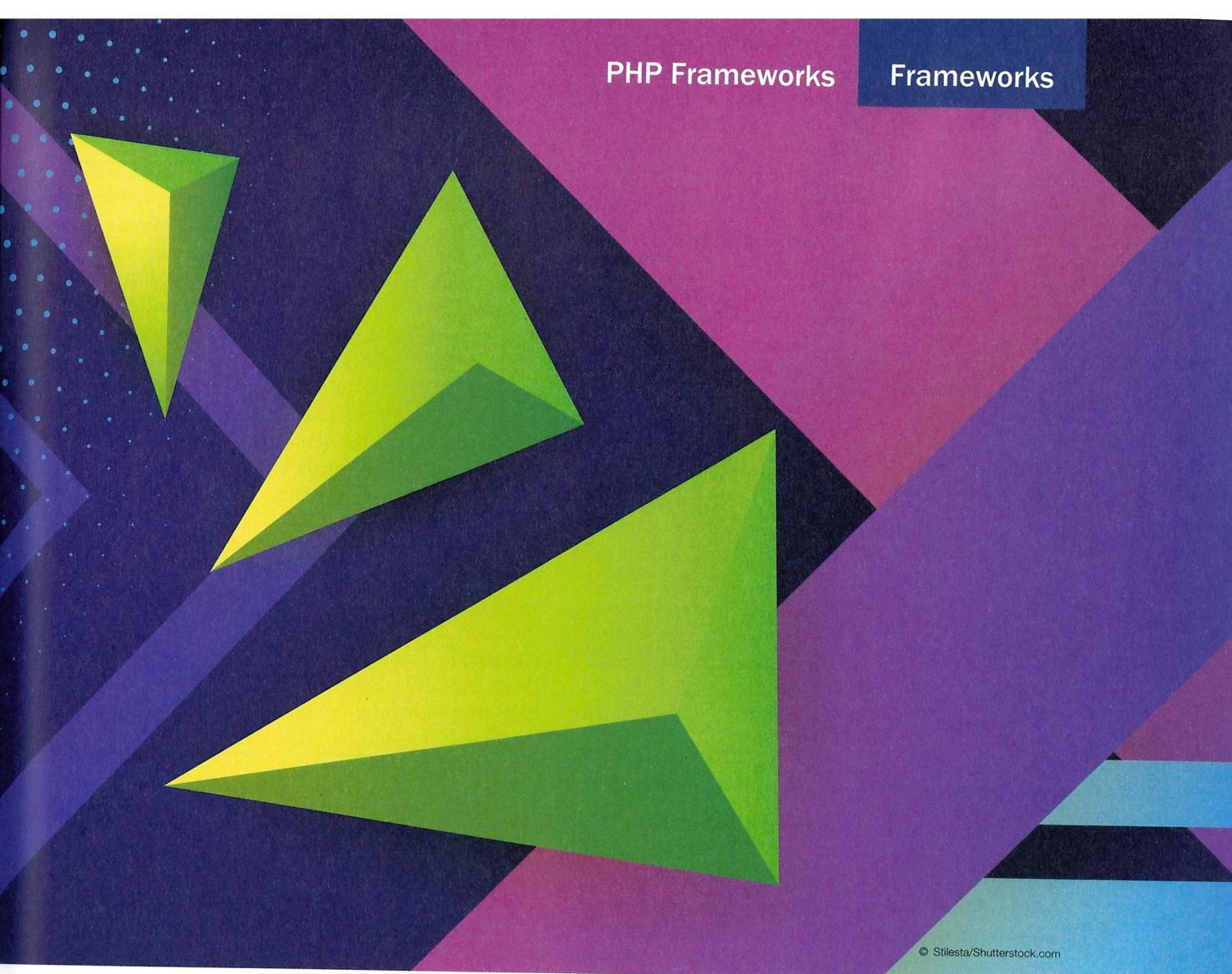
## Laravel – der Aufsteiger der letzten Jahre

Laravel ist eines der jüngsten und dennoch eines der beliebtesten PHP Frameworks. Das erste Release erschien im Jahr 2011. Der Gedanke hinter der Erstellung von Laravel war es, dass Frameworks wie CodeIgniter große Lücken aufwiesen. Zum Beispiel gab es bei CodeIgniter keine eingebaute Nutzerauthentifizierung. Auch im

Bereich Sicherheit gab es Herausforderungen. Der Entwickler Taylor Otwell entschied sich, etwas Besseres zu entwickeln. Basierend auf Symfony wurde Laravel geschrieben. Der Aufstieg nach dem Erscheinen war rasant. In Google Trends kann man sehen, wie Laravel ab 2013 seine beliebten Konkurrenten Zend und Symfony überholte (Abb. 1).

Besonders von Zend sind sehr viele Entwickler auf das neue Framework umgestiegen. Einige Punkte, die es so interessant machen, sind die folgenden:

1. *Moderner Ansatz:* Frameworks wie Symfony und Zend entstanden in den Jahren 2005 und 2006. Damals waren die Anforderungen an Webanwendungen noch andere, zu dieser Zeit gab es relativ wenige Webanwender. Auch waren Themen wie Sicherheit oder Skalierbarkeit nicht so wichtig. Das hat sich jedoch in den letzten Jahren stark geändert. Internetnutzer wollen schöne Webanwendungen, die einfach bedienbar, performant und sicher sind. All das ist in Laravel berücksichtigt.



**2. Höhere Performance:** Laut unterschiedlicher Quellen ist Laravel in der Ausführung um einiges schneller als andere Systeme. Taylor Otwell hat beispielsweise eine Benchmarkstudie durchgeführt und die Resultate auf Medium veröffentlicht [1]. Die Studie zeigt ausführlich, um wieviel schneller Laravel im Vergleich mit Zend und Symfony ist.

**3. Einfacher Einstieg:** Frameworks wie Yii und CodeIgniter sind so beliebt, weil es sehr einfach ist, dort einzusteigen. Mit einer kleinen Schulung kann man bereits die ersten Programme schreiben. Zend und Symfony dagegen haben eher eine hohe Lernkurve, und man braucht zum Teil Monate, um sich einzuarbeiten. Mit Laravel gibt es jetzt jedoch eine starke Alternative zu den beiden Frameworks Yii und CodeIgniter, die in Asien häufig verwendet werden. Jedoch steigt man auch dort mittlerweile auf Laravel um. Es gibt viele Tutorials, Onlineforen und Videoplattformen, auf denen man sich weiterbilden kann. Es wird von PHP-Entwicklern oftmals auch die starke Dokumentation der Software gelobt, die es noch einfacher mache, sich dort einzuarbeiten.

**4. Alternative zu Ruby on Rails:** Ruby on Rails ist eine performante und moderne Technologie, um Webanwendungen zu schreiben. Da Laravel jedoch ähnliche Performancemarkale hat, gilt es immer mehr als Alternative dazu.

Laravel ist ein spannendes PHP Framework. Es gibt jedoch einige Punkte, die nicht unerwähnt bleiben sollten:

- **Wenige Entwickler:** Laravel ist relativ neu, erst seit 2013 wächst es rasant über seine Gemeinschaft. Das bedeutet jedoch auch, dass es noch relativ wenige Programmierer gibt, die sich damit beschäftigt haben. Es ist einfacher, PHP-Experten in Zend oder Symfony zu finden.
- **Weniger getestet:** Andere Frameworks werden bereits seit vielen Jahren produktiv in Anwendungen genutzt und getestet. Man kennt die Herausforderungen und Möglichkeiten im Detail. Das ist anders bei Techno-

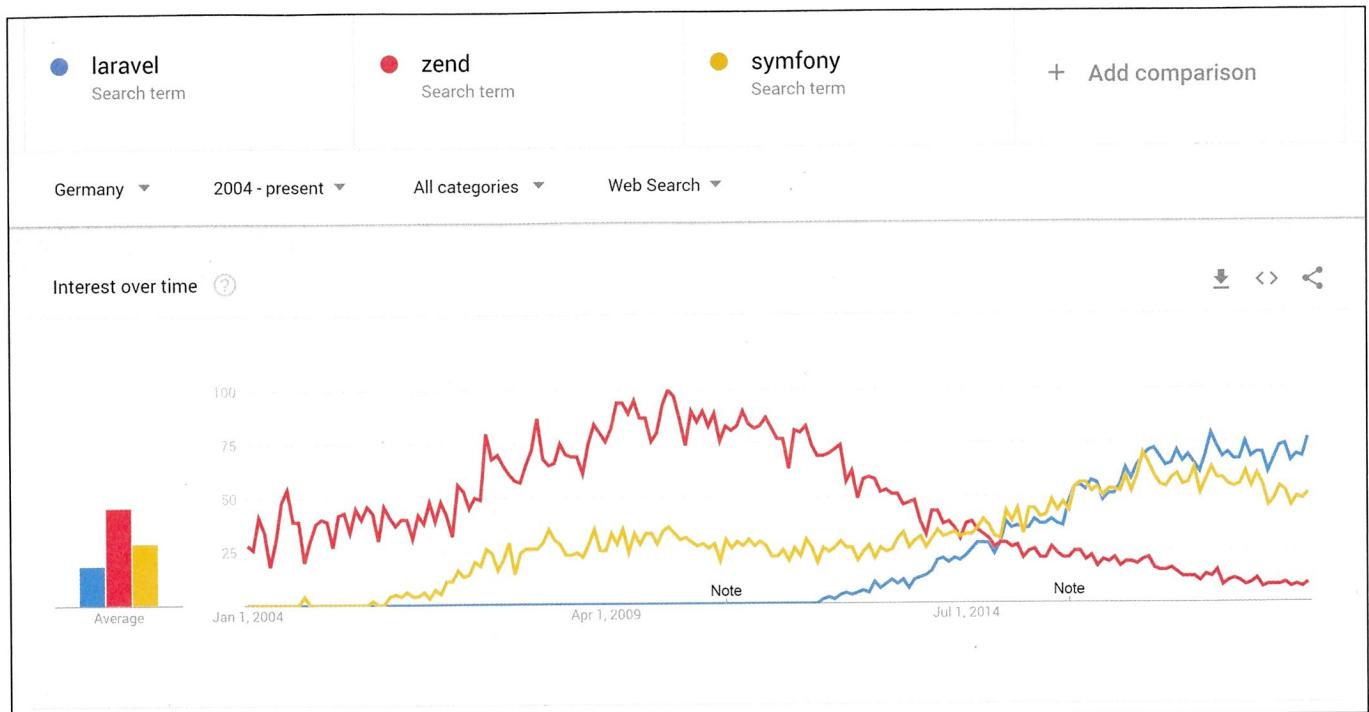


Abb. 1: Laravel (blaue Linie) ist der Aufsteiger der letzten Jahre (Quelle: Google Trends)

logien wie Laravel, die es erst seit ein paar Jahren im Markt gibt.

- *Geringe Nutzung im Bereich Enterprise:* Zend und Symfony haben sich bereits in Unternehmensanwendungen etabliert. Laravel ist hier eher neu. Zudem laufen viele bestehende Anwendungen bereits auf den etablierten Frameworks. Ein Umzug ist daher nicht immer möglich.

### **Zend – die Enterprise-Variante**

Zend ist ein PHP Framework, das sich als Anwendung für den Unternehmensbereich etabliert hat. Das erste Release erschien im Jahr 2006. Damit gibt es diese Technologie schon seit mehr als zehn Jahren. Zend kann mit einigen Vorteilen aufwarten:

- *Viele Experten:* Der durchschnittliche PHP-Entwickler hat hier und da bereits mit Zend gearbeitet und weiß, wie das System funktioniert. Die Gemeinschaft ist relativ groß. So findet man im Internet auch viele Foren und andere Hilfestellungen, falls während der Entwicklung Fragen auftauchen.
- *Onlineshopanwendungen programmierbar:* Eines der beliebtesten E-Commerce-Systeme (Magento) basiert auf Zend. Damit eignet sich das Framework besonders für Applikationen aus diesem Bereich.
- *Professioneller Betreiber:* Im April 2019 hat Zend verkündet, das Zend Framework und alle seine Teilprojekte auf eine neue Open Source Foundation unter der Linux Foundation unter dem Namen Laminas umzustellen. Ein großer Schritt, der die Professionalität der Macher des Frameworks weiter unterstreicht.

Natürlich steht man als PHP-Entwickler auch bei Zend vor einigen Herausforderungen:

- *Nicht leicht erlernbar:* Man braucht im Schnitt mehrere Monate, bis man richtig fit in diesem PHP Framework ist. Wenn man also schneller entwickeln möchte, sollte man auf andere Frameworks setzen.
- *Nicht mehr so modern:* Viele Zend-Entwickler sind in den letzten Jahren zu Laravel gewechselt, weil es leichter zu erlernen ist und auch einen moderneren Ansatz bietet.

### **Symfony – die Zend-Alternative**

Symfony wird, wie Zend auch, in mittelständischen Unternehmen für kleine bis mittelgroße Webanwendungen verwendet. Laravel basiert z. B. auch auf dem Symfony Framework. Neben Zend war Symfony lange Zeit das am weitesten verbreitete PHP Framework. Hier einige seiner Vorteile:

- *Tried and Tested:* Das Framework wird von vielen bekannten Unternehmen genutzt, um deren Portale zu betreiben. Firmen wie Notebooksbilliger, BlaBlaCar und ähnliche verwenden es, um ihre Websites damit zu bauen. Diese Plattformen sind dafür bekannt, dass sie viele Nutzer haben. Das bedeutet auch, dass sich Symfony bereits in der Praxis bewährt hat. Viele Bugs wurden über die Jahre behoben. Somit ist ein starkes Framework entstanden.
- *Große Community:* Bei der Entwicklung von Webanwendungen kommen oftmals viele Fragen auf. Diese lassen sich einfacher beantworten, wenn es eine große Gemeinschaft gibt, die auch an dieser Technologie arbeitet. Laut unterschiedlichen Quellen gibt es weltweit zirka 600 000 Symfony-Entwickler. Es lassen sich viele Onlineforen finden, in denen man seine Fragen beantworten lassen kann. In vielen

Fällen wurden ähnliche Fragestellungen wie die eigenen bereits beantwortet. Das erleichtert besonders Einsteigern die Arbeit.

- *Wiederverwendbarkeit von Funktionalitäten:* Mit Symfony kann man es vermeiden, gleiche Funktionalitäten immer wieder neu zu programmieren. Das Framework ermöglicht durch eine gute Strukturierung die Wiederverwendbarkeit von Code.
- *Ermöglicht das agile Arbeiten in Teams:* An größeren Webprojekten arbeitet oftmals nicht nur ein Entwickler. Symfony ermöglicht durch eine gute Struktur, eine gute Dokumentation und einen sauberen Code, dass viele Programmierer gleichzeitig an den Projekten arbeiten können.

### **CodeIgniter – ein leichtgewichtiges Framework**

Besonders in Asien war das Framework CodeIgniter lange Zeit verbreitet, mittlerweile sind jedoch viele Entwickler auf Laravel umgestiegen. Dennoch eignet sich CodeIgniter für kleinere Webprojekte. Wie auch Zend und Symfony, gehört CodeIgniter zu den älteren PHP Frameworks. Die Technologie wurde von der Firma EllisLabs im Jahr 2006 herausgebracht. Im Jahr 2014 wurde das Projekt aufgrund von fehlenden Ressourcen auf Seiten von EllisLabs an das British Columbia Institute of Technology (BCIT) übergeben. Hier einige Vorteile des Frameworks:

- *Einfach zu erlernen:* Besonders Einsteiger in PHP können bereits nach einer kurzen Einarbeitungszeit Webanwendungen erstellen.
- *Kleines Programmgerüst:* Die Idee hinter dem Framework ist es, im geringstmöglichen Programmgerüst die höchstmögliche Performance und Flexibilität zu bieten.
- *Für kleine Webanwendungen geeignet:* Wer nur wenige Funktionalitäten benötigt, jedoch schnell entwickeln möchte, für den ist CodeIgniter ideal.
- *Gut dokumentiert:* Besonders in der englischen Sprache gibt es viele Einsteigertutorials. Auch der Quellcode ist sauber und gut kommentiert. Das erleichtert das Arbeiten mit dem Framework.

Natürlich gibt es auch einige Punkte, die nicht für CodeIgniter sprechen:

- *Immer weniger Entwickler:* Es gibt die Tendenz, dass CodeIgniter-Programmierer immer mehr auf Laravel umsteigen, da es ähnlich einfach zu erlernen ist und ebenfalls eine gute Dokumentation aufweist.
- *Weniger geeignet für große Projekte:* Großprojekte, die skalierbar sein und eine hohe Sicherheit aufweisen sollen, eignen sich meistens nicht für CodeIgniter.
- *Weniger Bibliotheken:* Einige Programmierer sehen das als Vorteil. Wenn man jedoch Standards wie E-Mail-Versand, Validierung von Formulardaten etc. einbauen möchte, müssen unter anderem Ressourcen von Drittanbietern eingebunden werden.

### **Yii – die Alternative**

Das Framework Yii kann als Alternative zu Laravel, Zend und Symfony gelten. Es gibt immer noch einige IT-Dienstleister und Entwickler, die es nutzen, um performante Webanwendungen zu erstellen. Von der Beliebtheit her ist es vergleichbar mit CodeIgniter. Es wird heutzutage jedoch eher weniger genutzt. Wie CodeIgniter auch, ist es sehr stark in Asien verbreitet. Auch von Yii steigen viele Entwickler auf Laravel um. Hier einige Vorteile des Frameworks:

- *Schnelle Entwicklung möglich:* Ein großer Vorteil dieses Frameworks ist es, dass es die schnelle Programmierung von Webanwendungen unterstützt.
- *Verbindung mit Zend möglich:* Es lässt sich Code von anderen Frameworks wie Zend oder PEAR (eine Quelle für Standard-PHP-Code zum Wiederverwenden) einbinden.
- *Websicherheit besser als bei CodeIgniter:* Webanwendungen sollen sicher sein. Yii hat hier einen besonderen Fokus.

Auch wenn das Yii Framework über lange Jahre sehr beliebt war, gibt es heutzutage mit Laravel eine gute Alternative. Viele Agenturen, IT-Dienstleister und IT-Abteilungen steigen direkt bei Laravel ein und vermeiden den Einsatz von Yii. Zudem ist der Hauptentwickler und Erfinder Qiang Xue nicht mehr an dem Projekt beteiligt. Seitdem ist es etwas ruhiger um das Framework geworden.

### **Fazit**

Es gibt heutzutage wirklich nur noch drei sehr gute PHP Frameworks: Laravel, Symfony und Zend. CodeIgniter und Yii werden dagegen nur noch selten genutzt. Besonders Laravel hat in den letzten Jahren überzeugt. Es hat die komplette PHP-Landschaft wieder „sexy“ gemacht. Zudem konnte es helfen, das Klischee vom PHP-Spaghetticode aufzulösen. Mit Laravel lassen sich skalierbare, performante und sichere Webanwendungen programmieren. Zudem ist es einfach zu erlernen und man kann damit schneller entwickeln, da es bereits viele Standardkomponenten enthält.

Wer auf Nummer sicher gehen will, kann auf bewährte Enterprise PHP Frameworks wie Zend oder Symfony setzen. Diese beiden Frameworks haben über viele Jahre bewiesen, dass sich mit ihnen wartbare und gute Webplattformen umsetzen und betreiben lassen.



**Sascha Thattil** arbeitet bei YUHIRO als Geschäftsführer. YUHIRO stellt indische Entwickler für Agenturen, Softwareunternehmen und IT-Abteilungen bereit.

### **Links & Literatur**

[1] <https://medium.com/@taylorotwell/benchmarking-laravel-symfony-zend-2c01c2b270f8>

Laravel: ein PHP Framework (nicht nur) für coole Kids

# Mehr Eleganz beim Programmieren

Bei Laravel [1] handelt es sich um ein typisches Open-Source-PHP-Framework, das dem MVC-Architekturmuster folgt. Seit der Veröffentlichung der ersten Version im Juni 2011 hat die Popularität stetig zugenommen. Spätestens mit Version 4.0, die seit Mai 2013 verfügbar ist, hat sich Laravel etabliert. Mit dieser Version ging auch die Umstellung auf eine Composer-basierte Installation und Paketverwaltung einher.

von Max Erkemann

Seit jeher wird die Entwicklung von Laravel maßgeblich durch Taylor Otwell, dem Initiator und Maintainer des Projekts, vorangetrieben und geprägt. Während es gerade in der Anfangszeit noch des Öfteren selbst in Minor-Releases nichtabwärtskompatible Änderungen gab, hat sich die Situation mittlerweile deutlich entspannt. Die Dokumentation von Laravel enthält für jedes Release einen ausführlichen Upgrade-Guide mit einer Liste der relevanten Änderungen und möglichen Stolpersteine. Zudem wurde erstmalig im Juni 2015 eine Version mit Langzeitunterstützung veröffentlicht. LTS-Versionen von Laravel werden für zwei Jahre nach Erscheinen mit Bugfixes und im Anschluss daran für ein weiteres Jahr mit Sicherheitsupdates versorgt.

Folglich ist gerade jetzt ein guter Zeitpunkt, um den Einsatz von Laravel als Framework für neue oder bestehende Projekte in Betracht zu ziehen. Abgesehen von der wichtigen Langzeitunterstützung ist es vor allem auch die Akzeptanz im professionellen Umfeld, die für einen Einsatz spricht. Zu den bekannten Anwendern von Laravel zählen unter anderem das Schnäppchenportal mydealz und der Onlineshop ABOUT YOU.

## MVC 101

Laravel bietet als Full-Stack-Framework alle nötigen Funktionen, um professionelle Webapplikationen zu entwickeln. Mittels Composer ist ein neues Projekt auf

Basis von Laravel schnell mit der folgenden Zeile eingerichtet: `composer create-project --prefer-dist laravel mein-shop`.

In den grundlegenden Funktionen weist Laravel viele Parallelen mit anderen Frameworks wie Symfony auf. So werden die einzelnen Seiten respektive HTTP Endpoints einer Webanwendung dem MVC-Entwurfsmuster entsprechend in Controller-Klassen zusammengefasst. Die URLs für den Zugriff darauf werden als sogenannte Routes definiert. Hierbei ist man nicht eingeschränkt und kann das URL-Schema frei gestalten. Verwendete Platzhalter können optional auf einen regulären Ausdruck geprüft werden oder dazu dienen, ein benötigtes Model automatisch aus der Datenbank zu laden und im Controller bereitzustellen:

```
// routes.php
Route::get("orders/{order}/invoice.pdf", "OrdersController@getInvoicePdf");

// OrdersController.php
public function getInvoicePdf(Order $order) {
    return response()->download($order->getInvoicePath(), "invoice.pdf");
    // Das Model mit der im Platzhalter genutzten ID ist direkt in der
    // Controller-Methode verfügbar
}
```

Ebenfalls lassen sich Routes speziell für einzelne oder alle Subdomains definieren. Gerade für SaaS-Anwendungen, bei denen jeder Mandant eine eigene Subdomain erhält,

ist das durchaus praktisch. Das von einem Platzhalter erfasste Segment der Subdomain kann dabei als Parameter an die Controller-Methoden weitergereicht und dort verwendet werden.

Laravel unterstützt die Datenbanken MySQL, PostgreSQL, SQLite, SQL Server und den Key-Value-Store Redis. Für die Arbeit mit einer Datenbank bietet Laravel neben einem Query Builder, der das Erstellen sicherer, vor SQL Injections geschützter SQL-Abfragen vereinfacht, auch ein ORM namens Eloquent. Eloquent nutzt, anders als beispielsweise Doctrine, kein Data-Mapper-Pattern, sondern das Active-Record-Pattern. Während so ein unkomplizierter Einstieg in die Entwicklung möglich ist, sollte man sich bewusst sein, dass Active Record zumindest in Hinsicht auf die SOLID-Prinzipien ein Antipattern ist. Es werden verschiedene Zuständigkeiten – vor allem Domain-/Businesslogik und Persistenz – in einer Klasse vermischt, was nicht selten zu sogenannten Gottobjekten [2] führt.

Der Einsatz von Eloquent ist dementsprechend eine Geschmacksfrage und kann gerade bei komplexen Projekten problematisch sein. Während Laravel standardmäßig auf die Verwendung von Eloquent angepasst ist, spricht aber nichts dagegen, stattdessen ein anderes ORM einzusetzen oder die Datenbankabfragen direkt mittels Query Builder zu erstellen und in Repository-Klassen zu kapseln. Abseits vom reinen Datenbankzugriff bietet Laravel mit den Migration-Klassen eine leicht zugängliche Funktion zum Erstellen und Modifizieren des Datenbankschemas. Gerade für das automatisierte Testen und während der Entwicklung werden realistische Testdaten benötigt. Damit diese nicht jedes Mal von Hand angelegt oder umständlich in Form von SQL-Dumps weitergegeben werden müssen, lässt sich die Generierung von Testdaten in separaten Seeder-Klassen abbilden. Hierbei stehen verschiedene nützliche Hilfsfunktionen bereit.

Für die Ausgabe von HTML-Code bringt Laravel eine eigene Templatesprache mit. Die Ähnlichkeiten von Blade, so der Name, und dem Templatesystem Razor aus dem ASP.NET Framework sind nicht zufällig. Die Idee zu Laravel entstand, als Taylor Otwell kein mit ASP.NET vergleichbares Framework für PHP finden konnte. Im Kern handelt es sich bei Blade-Templates um gewöhnliche HTML-Dokumente, die durch spezielle Tags um Logik erweitert werden. Davon bietet Laravel eine ganze Reihe, was den Entwickleralltag komfortabler gestaltet:

```
@forelse($orders as $order)
<li>{{ $order->id }}: {{ $order->total }} €</li>
@empty
<p>Keine Bestellungen vorhanden.</p>
@endforelse
// Nützlich: Das Handling einer leeren Ergebnisliste wird durch Blade
// vereinfacht
```

Ebenfalls Bestandteil von Laravel ist ein Frontend-Grundgerüst auf Basis von Vue.js und Bootstrap. Wem

Vue.js nicht zusagt, der kann die Vorlage mittels Artisan auf React umstellen. Artisan ist das Laravel-eigene Kommandozeilenwerkzeug, mit dem viele Aufgaben, wie z. B. Codegenerierung, erledigt werden können. Um die Konfiguration von webpack zu vereinfachen, wird das npm-Paket Laravel Mix mitgeliefert. Standardaufgaben wie das Kompilieren und Minifizieren von Assets sind so schnell eingerichtet und leicht anpassbar.

Für reine API-Projekte, z. B. im Zusammenspiel mit einer Single Page Application, wird eine JWT-basierte Authentifizierung angeboten. Diese ist auch bereits im Frontend-Grundgerüst vorkonfiguriert. Neben der vollständigen Laravel-Installation steht mit Laravel Lumen auch eine Art „Laravel light“ für die Entwicklung von APIs bereit. Zur Steigerung der Performance wurden hier einige für APIs weniger relevante Funktionen entfernt. So fehlt unter anderem das gesamte Session-Handling, wodurch eine Authentifizierung immer mittels API Key oder Token erfolgen muss. Sollten sich die Anforderungen eines Projekts ändern, ist ein Upgrade auf die vollständige Laravel-Version jederzeit möglich.

## Warum Laravel?

Zu Recht stellt sich nun vermutlich die Frage, was Laravel konkret von der Vielzahl an konkurrierenden PHP Frameworks abgrenzt und zu einer guten Wahl macht. Da die bisher genannten Funktionen so oder in ähnlicher Form in fast jedem Framework enthalten sind, werde ich im Folgenden auf eine Auswahl von Features eingehen, die aus meiner Sicht herausstechen.

Bereits die Laravel zugrunde liegende Softwarearchitektur ist sehr elegant. So wird durchgängig auf Dependency Injection als Entwurfsmuster gesetzt, um die Wart- und Testbarkeit der Anwendungen zu erhöhen. Dreh- und Angelpunkt ist hierbei der Service-Container, der für die Auflösung und Bereitstellung von Abhängigkeiten zuständig ist. Dadurch ist es nicht nötig, auf globale Objekte oder statische Klassen zurückzugreifen. Statt einer manuellen Instanzierung der Abhängigkeiten im Code werden diese einfach vom Service-Container angefordert. Das geschieht entweder explizit durch den Aufruf einer Hilfsfunktion oder implizit innerhalb von Objekten, die ebenfalls vom Service-Container bereitgestellt werden. Hierzu zählen auch alle Controller-Klassen. Enthält die Parameterliste eines Objektkonstruktors eine typisierte Abhängigkeit, wird der Service-Container versuchen, sie im Konstruktor zu injizieren:

```
CartController extends Controller {
    public function __construct(CartService $cartService) {
        $this->cartService = $cartService;
    }
}
// Die Abhängigkeit zum CartService wird automatisch aufgelöst, da der Cart
// Controller ebenfalls vom Service-Container instanziert wird
```

Standardmäßig erfolgt die Instanzierung schlicht mittels `new Dependency();`. Für komplexe Objekte mit ei-

## Hat die eigene Anwendung eine gewisse Nutzerzahl, wird das Thema Performance wichtig.

genen Abhängigkeiten bieten sich Service-Provider an. Hierbei handelt es sich um gesonderte Klassen, die definieren, wie bestimmte Objekte vom Service-Container zu erzeugen sind.

Praktisch ist zudem, dass nicht nur reguläre Klassen, sondern auch abstrakte Klassen oder Interfaces als Abhängigkeiten angegeben werden können. Um sie auflösen zu können, ist die Registrierung einer konkreten Implementierung im Service-Provider notwendig. Möchte man bestimmte Objekte nicht stets neu erzeugen lassen, können sie auch als Singleton definiert werden. Beim ersten Zugriff auf diese Abhängigkeit wird sie dann vom Service-Container erzeugt und bei jeder folgenden Anfrage stets diese Instanz wiederverwendet.

Um Funktionalität zu kapseln, die mehr als einen HTTP Endpoint betrifft und eigentlich nicht Teil der jeweiligen Logik ist, kann eine sogenannte Middleware erstellt werden. Dabei handelt es sich um eine Klasse oder Funktion, die die eigentliche Controller-Methode umschließt und den Request oder die Response modifizieren kann. Es lassen sich mehrere Middlewares in Reihe schalten, sodass man bei einem API beispielsweise die Authentifizierung oder ein etwaiges Throttling in separaten Klassen implementieren kann. In Laravel selbst ist unter anderem das Session-Handling und die CSRF-Token-Validierung als Middleware umgesetzt.

Wie bereits erwähnt, liefert Laravel ein eigenes CLI-Tool namens Artisan mit. Hiermit lassen sich nicht nur Framework-spezifische Aufgaben wie das Generieren von Controller- oder Model-Klassen durchführen, sondern es können auch eigene Funktionen (Commands) realisiert werden. Das sind typischerweise jene Funktionen, die man periodisch als Cronjob ausführen lassen möchte, um Back-ups zu erstellen, Zahlungserinnerungen zu versenden oder Daten mit Drittsystemen zu synchronisieren. Zwar lassen sich die eigenen Artisan Commands ganz normal als Cronjob konfigurieren, doch einfacher ist es, den Task Scheduler zu nutzen. So lässt sich das Intervall mit einer leichtverständlichen Syntax definieren, und es ist lediglich nötig, den Task Scheduler selbst als Cronjob zu hinterlegen:

```
$scheduler->command("meinshop:send-payment-reminder")->daily();  
// Das eigene Artisan Command wird täglich ausgeführt
```

In vielen Anwendungen wird eine Dateiuploadfunktion benötigt, und sei es nur, um dem Nutzer das Hochladen eines eigenen Profilbilds zu ermöglichen. Aber auch sonst fallen oft Dateien an, die irgendwo gespeichert werden müssen. Um das zu vereinfachen, bietet Laravel eine

separate Abstraktionsschicht, sodass es aus Entwicklersicht kaum einen Unterschied macht, ob die Daten auf der lokalen Festplatte, einem (S)FTP-Server oder in einem AWS S3 Bucket abgelegt werden. Die Funktionen zum Speichern, Laden oder Löschen von Dateien stehen hierbei in Form eines einheitlichen API zur Verfügung. Wie für Laravel üblich, besteht auch hier die Möglichkeit, weitere Services anzubinden. In der Dokumentation wird das am Beispiel von Dropbox als zusätzlichem Speicherort erläutert. Sobald die eigene Anwendung eine gewisse Nutzeranzahl erreicht, wird das Thema Performance immer bedeutsamer. Eine wirkungsvolle Optimierung ist klassischerweise das Caching, damit die Anzahl der nötigen Datenbankabfragen reduziert wird oder Daten, die aus dem API eines Drittanbieters bezogen werden, nicht in jedem Request erneut abgerufen werden müssen. Von Haus aus unterstützt Laravel Datenbanken, Redis, Memcached oder das Dateisystem als Backend für das Cachesystem.

Ein weiterer Ansatz ist das Auslagern nachrangiger oder zeitintensiver Vorgänge, wie z. B. das Versenden von Bestätigungs-E-Mails, aus der eigentlichen Controller-Logik. Diese Aufgaben werden in einer Queue abgelegt und von einem oder mehreren separaten Hintergrundprozess(en) abgearbeitet. Dadurch wird die Antwortzeit für den Endnutzer merklich reduziert, da die Antwort viel schneller generiert und vom Server an den Browser gesendet werden kann. Das Queue-System unterstützt nicht nur verschiedene Backends, sondern ist auch sonst sehr flexibel. Aufgaben können sowohl direkt ausgeführt als auch für einen späteren Zeitpunkt vorgeplant oder mit unterschiedlicher Priorität abgearbeitet werden, schließlich ist nicht jede Aufgabe gleich relevant. Während die Bestellbestätigung oder die „Passwort vergessen“-E-Mail am besten direkt im Postfach des Nutzers landen soll, darf der tägliche Report für das Team vielleicht auch mal etwas später generiert und versendet werden. Selbst wenn aktuell kein Einsatz von Laravel geplant ist, kann ich jedem nur empfehlen, schon alleine wegen dieses Features einen Blick in die umfangreiche und erstklassige Laravel-Dokumentation zu werfen.

Natürlich kommt auch das Thema automatisiertes Testing nicht zu kurz. Neben den üblichen Verdächtigen wie Unit-Tests mit PHPUnit unterstützt Laravel auch browserbasierte Tests mit Laravel Dusk. Gerade in Zeiten, in denen immer mehr Logik ins Frontend verlagert wird, ist es eigentlich unumgänglich, direkt in einem Browser zu testen. Diese Tests lassen sich wie gewohnt in PHP schreiben und können mit ChromeDriver oder einem Selenium-Server ausgeführt werden.

Für Neueinsteiger ist es verwunderlich, dass in Codebeispielen zu Laravel oft vermeintlich statische Klassen wie *Request*, *Session* oder *Cache* genutzt werden. Tatsächlich handelt es sich hierbei aber nicht um statische Klassen im eigentlichen Sinne, sondern um von Laravel als Facades bezeichnete Wrapper, die Objekte im Service-Container referenzieren. Es ist ohne Weiteres mög-

lich, einzelne Funktionen dieser Facades für Unit-Tests zu mocken. Auch erhält man dank Dependency Injection beispielsweise einfach Zugriff auf die Instanz des aktuellen Requests, indem man die Request-Klasse in die Parameterliste aufnimmt. So können alle bisherigen Aufrufe auf der Request Facade durch direkte Aufrufe auf dem eigentlichen Request-Objekt ersetzt werden.

### Darf es etwas mehr sein?

Ohne ein Feature im eigentlichen Sinne zu sein, dürfte einer der größten Pluspunkte das um Laravel herum entstandene Ökosystem sein. Während Laravel selbst schon einen beachtlichen Funktionsumfang liefert, erhält man hier nochmals eine Fülle an weiteren Features und Hilfestellungen. Für die Serverkonfiguration und das Deployment gibt es Laravel Forge [3] und Envoyer [4]. Das lästige Verwalten von Servern gehört damit der Vergangenheit an. Grundlegendes wie automatische Sicherheitsupdates des Betriebssystems und der Software, das Einrichten von SSL-Zertifikaten oder ein Git-basierter Zero-Downtime-Deployment-Prozess sind hier bereits sehr gut gelöst, sodass man sich voll und ganz auf das eigentliche Projekt und den eigenen Code fokussieren kann.

Für die lokale Entwicklung gibt es mit Laravel Homestead [5] eine fertige Vagrant-Box, die nichts vermissen lässt und somit den direkten Einstieg in die Entwicklung ermöglicht. Für Mac-Nutzer gibt es mit Laravel Valet [6] eine Alternative, die auf lokale Komponenten statt einer virtualisierten Linux-Maschine setzt. Das ist nicht nur ressourcenschonender, sondern für viele Projekte auch vollkommen ausreichend. Hinzu kommt, dass es für viele API-Clients und Pakete von Drittanbietern bereits speziell auf Laravel zugeschnittene Versionen gibt, die ihre Abhängigkeiten beispielsweise direkt im Service-Container registrieren oder ihre Funktionen in Form von Middleware bereitstellen.

SaaS-Projekte können vom Einsatz von Laravel Spark profitieren. Das kostenpflichtige Paket beinhaltet typische Funktionen, die in einer solchen Anwendung benötigt werden. Dazu zählen eine Team- und Rechteverwaltung oder die Abwicklung wiederkehrender Kreditkartenzahlungen mit Stripe. Wer neben einem schönen Endnutzer-Frontend auch Wert auf eine moderne und ansehnliche Administrationsoberfläche legt, kann einen Blick auf das ebenfalls kostenpflichtige Laravel Nova werfen. Beide Pakete stammen auf der Feder des Laravel-Schöpfers Taylor Otwell und bieten somit natürlich eine perfekte Integration mit dem Framework.

### Fazit

Wie so oft in der Softwareentwicklung lässt sich auch die Frage, ob man Laravel einsetzen oder gar bestehende Projekte nach Laravel migrieren sollte, mit einem klaren „it depends“ beantworten. Eine pauschale Antwort ist nicht möglich, und es kommt immer auf den konkreten Projektkontext an. Für neue Projekte, bei denen auch die benötigte Einarbeitungszeit keine Deadline sprengt,

sollte man sich Laravel unbedingt anschauen. Es zählt nicht ohne Grund zu den populärsten Frameworks und bietet rein aus technischer Sicht eine absolut solide Grundlage für kleine und große Projekte. Gerade die LTS-Releases sprechen für einen Einsatz in langfristig ausgelegten Projekten.

Da im Rahmen eines Artikels nicht vollumfänglich auf jede Funktion eingegangen werden kann, stellt die offizielle Dokumentation eine erste Anlaufstelle für tiefergehende Informationen dar. Sie ist gut gegliedert, leicht verständlich und vor allem aktuell. Persönlich kann ich nur feststellen, dass es nicht ausschließlich die vielen Features und Lösungen aus dem Laravel-Ökosystem sind, die Laravel nach wie vor zum Framework meiner Wahl machen. Vielmehr sind es die unzähligen kleinen Details, die mich immer wieder denken lassen: „Cool, da hat sich jemand wirklich Gedanken gemacht und eine smarte Lösung gefunden.“ Und das führt letztlich dazu, dass das Entwickeln mit Laravel einfach Freude bereitet.



**Max Erkmann** arbeitet als Softwareentwickler für die Firma SONNENGLAS. Zuvor war er unter anderem als Freelancer und als CTO für aboalarm tätig. Abseits vom Computer reist er gerne zusammen mit seiner Frau – stets auf der Suche nach dem nächsten Abenteuer und WiFi-Hotspot.

### Links & Literatur

- [1] <https://www.laravel.com>
- [2] <https://de.wikipedia.org/wiki/Gottobjekt>
- [3] <https://forge.laravel.com>
- [4] <https://envoyer.io>
- [5] <https://laravel.com/docs/5.8/homestead>
- [6] <https://laravel.com/docs/5.8/valet>