



## Wie PHP den Web-2.0-Alltag der deutschen Bild- und Videoplattform sevenload meistert

# Film ab!

von Thomas Bachem

Der Begriff „Web 2.0“ fasst zusammen, was moderne Web-Plattformen leisten müssen: Große Mengen an Daten und Informationen sollen innerhalb von Millisekunden miteinander verknüpft, weiterverwendet und ausgetauscht werden. Die Anforderungen an die Entwicklung moderner Web-Applikationen sind gewaltig – und mit PHP zu meistern. Die deutsche Bild- und Videoplattform sevenload macht vor, wie Web 2.0 mit PHP funktioniert.

„Geht das denn mit PHP?“ Diese Frage kriegen wir bei sevenload regelmäßig zu hören – viele Entwickler unterschätzen die Einsatzmöglichkeiten von PHP anscheinend deutlich. Dabei machen viele große Websites vor, dass es möglich ist: Branchengrößen wie flickr [1] machen kein Geheimnis daraus [2], dass sie PHP nutzen, um täglich Millionen von Nutzern ihren Service zu ermöglichen. Dieser Artikel greift ein paar konkrete Praxisbeispiele auf und erklärt, was bei der Entwicklung und Wartung komplexer Webanwendungen zu beachten ist. sevenload ist eine Plattform für Bilder und Videos, die direkt nach dem Onlinestart bereits mehrere hundert Anfragen pro Minute verkraften musste. Von Benutzern hochgeladene Bilder und Videos müssen in verschiedenste Formate kodiert, in diversen Größen skaliert und dauerhaft

und zuverlässig verwaltet werden. All dies ist auf Dauer unmöglich mit nur einem einzigen Serversystem zu bewerkstelligen – sevenload brauchte also eine frei skalierbare und redundante Serverumgebung, um weiterhin für seine Nutzer da sein zu können.

### Und MySQL?

Kommt PHP zum Einsatz, liegt natürlich auch die Verwendung von MySQL nahe. Der Erfahrungsschatz beim Einsatz von MySQL war bereits sehr hoch, weshalb MySQL auch die erste Wahl für das dahinterliegende Datenbanksystem bei sevenload war. Als Datenbank-Engine greifen wir auf InnoDB zurück, das in unseren Benchmarks eine bessere Performance als das standardmäßige MyISAM lieferte. Da wir in der Datenbank ausschließlich Textdaten speichern und

alle Multimedia-Dateien über das Dateisystem verwalten, umfasst die sevenload-Datenbank derzeit nur knapp 180 MB bei ca. 930.000 Zeilen.

MySQL bringt zudem einen Mechanismus zur Replikation mit sich [3], mit dem sich auch die Datenbank zu großen Teilen skalieren lässt (mehrere so genannte MySQL-Slaves spiegeln alle Daten in Echtzeit und stehen für selects zur Verfügung), so können Abfragen und damit Serverlast verteilt werden. Die Datenbank ist das zentrale Herzstück moderner Web-Applikationen. Hier lagert, was eine interaktive Website zu dem macht, was sie ist: die Inhalte und das Gedankengut der Nutzer. Vergessen Sie im Interesse Ihrer Benutzer wie im eigenen Interesse also niemals, von diesen Daten regelmäßig – je nach Frequenz der Website vielleicht sogar

mehrmals täglich – Backups anzulegen. Verlorene gegangene Daten sind gegenüber Benutzern so gut wie unmöglich zu rechtefertigen. Denken Sie daran: Auch Replikation schützt nicht vor ungewollten deletes, denn diese werden in Echtzeit auf alle Slaves übertragen.

## Gemeinsam stark

Eine besondere Herausforderung bei der Entwicklung von sevenload ist die Benutzung mehrerer Server. Load Balancer und exakt gespiegelte Serversysteme helfen zwar beim Verarbeiten vieler Anfragen an den Webserver, doch lösen Sie darüber hinaus gehende Problematiken nur schwerlich: Wie stellt man sicher, dass alle Server auf eine zentrale Datenbank zugreifen? Wo lagert man all die Giga- und vielleicht bald Terabytes an Daten? Wie schützt man die Daten gegen Verluste? Wie geht man mit den vielen hundert Gigabyte an täglichem Traffic um? Wo nimmt man die Rechenzeit her, um Bilder und Videos umzuwandeln und zu skalieren? Wie bewahrt man das System allgemein vor Performance-Zusammenbrüchen? Und wie stellt man das alles ohne ein Millionen-Budget an?

Die Antwort liegt zum einen im von Insidern als weiteren Kernbestandteil des „Web 2.0“ bezeichneten großen Angebot an kostengünstigen Open-Source-Lösungen und zum anderen in der Kreativität der Entwickler. Analog zum Entwicklerteam ist auch bei den Servern eine Aufgabenteilung erforderlich. Bei sevenload beispielsweise fallen Skriptausführung, Datenbank, Konvertierung, Datenlagerung/Auslieferung, Backup und Monitoring an. Jedem dieser Aufgabengebiete können beliebig viele Server zur Verfügung stehen, das ist wichtig für eine freie Skalierbarkeit bei steigenden Ansprüchen an die Server. Die Serversysteme sollten individuell auf die zu erfüllenden Aufgaben zugeschnitten werden: Die Datenbank braucht viel RAM, die Konvertierung viel CPU, die Datenlagerung/Auslieferung gute und große Festplatten und das Backup eigentlich einfach nur Speicherplatz.

Die Kunst besteht nun vorrangig darin, Serverzugriffe untereinander im normalen Betrieb (beim Ausliefern der Inhalte) zu mi-

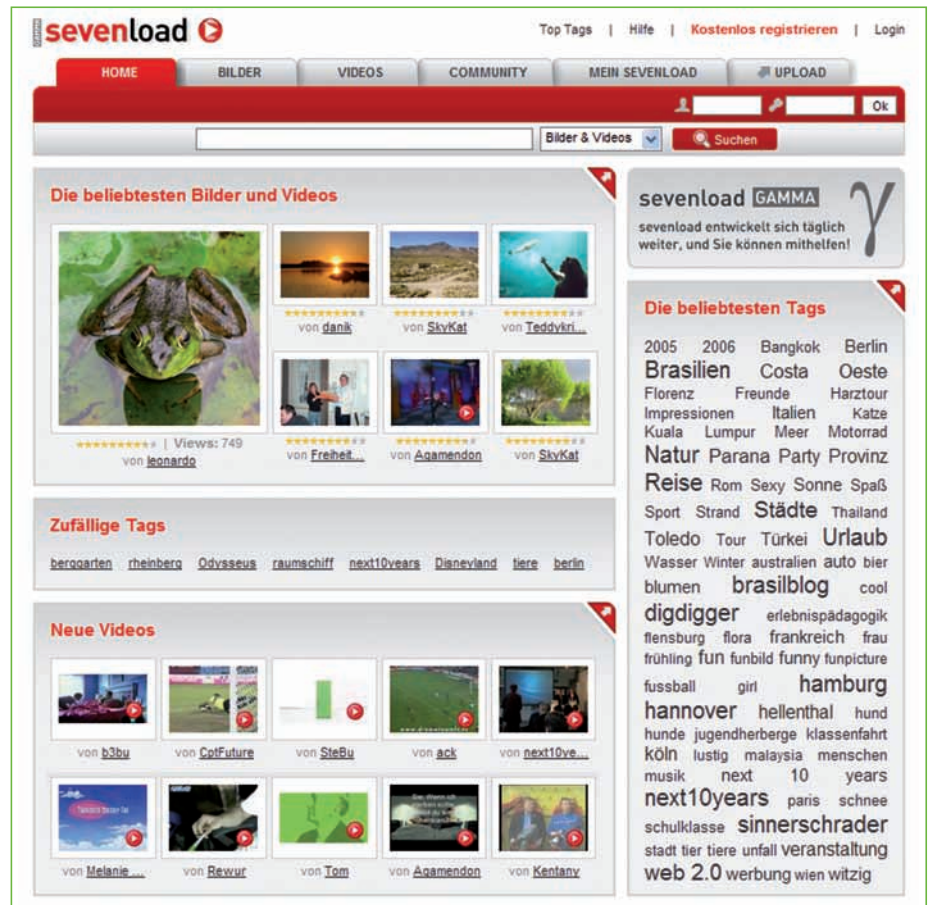


Abb. 1: Blick auf die Website von sevenload

nimieren. Denn sobald Daten die eigenen Servergrenzen verlassen, darf nicht mehr damit gerechnet werden, dass sie innerhalb von wenigen Millisekunden zur Verfügung stehen. Netzwerkverbindungen können

schnell fehlerhaft oder langsam sein, und die Server müssen weder nebeneinander, noch im selben Rechenzentrum, noch am selben Standort und vielleicht noch nicht einmal beim selben Provider stehen. Des-

Tabelle 1: Tabelle „images“

image_id	image_title	user_id	image_size
...	...	...	...
276	Strandpanorama	80	800x600
277	Blumen	15	1200x1600
278	Der Kölner Dom	24	640x480
...	...	...	...

Tabelle 2: Tabelle „images\_ratings“

rating_id	image_id	user_id	rating
...	...	...	...
1482	276	94	8.0
2845	277	73	6.0
3214	277	19	7.0
3735	276	63	9.0
4285	276	112	8.0
...	...	...	...

image_id	image_title	user_id	image_size	average_rating
...	...	...	...	...
276	Strandpanorama	80	800x600	8.3
277	Blumen	15	1200x1600	6.5
278	Der Kölner Dom	24	640x480	0.0
...	...	...	...	...

Tabelle 3: Tabelle „images“ mit Spalte „average\_rating“

halb ist ein zentraler Knotenpunkt wichtig, der viele Informationen bereithält: die Datenbank. Sie sagt uns, wo und bei wem wir welche Daten finden.

### Doppelt hält besser

Man muss sich als Entwickler davon frei machen, möglichst alle Daten normalisieren, d.h. ohne doppelte Vorkommen oder besondere Abhängigkeiten speichern zu wollen. Nur Mut zu doppelt vorhandenen Werten in der Datenbank! Ein kleines Beispiel hierzu: Gehen wir davon aus, eine Tabelle „images“ mit je einer Zeile pro eingestelltem Bild zu haben (Tabelle 1). Nun möchten wir diese Bilder gerne von unseren Benutzern bewerten lassen – und uns die Möglichkeit offen halten, diese Bewertungen später zum Vorteil der Nutzer noch weiter auszuwerten. Deshalb speichern wir jede abgegebene Bewertung eines Bildes als Zeile in einer anderen Tabelle „images\_ratings“ und verknüpfen diese Bewertungen mithilfe einer eindeutigen ID (eines Primary Keys) mit den Zeilen der Tabelle „images“ (Tabelle 2).

Möchten wir nun die Gesamtbewertung eines Bildes ermitteln, so müssen wir den Durchschnitt aller Einzelbewertungen ermitteln, d.h. alle miteinander addieren und dann durch ihre Anzahl teilen. Diesen Wert immer in Echtzeit zu berechnen ist so lange möglich, bis wir eine Auflistung aller Bilder sortiert nach ihrer Bewertung anbieten möchten – und unsere Tabelle „images“ bereits einige tausend Zeilen umfasst. Denn viele tausend Zeilen mit vielen anderen zehntausend Zeilen (z.B. 5.000 x 40.000 = 200.000 von MySQL intern zu verarbeitende Zeilen) in Echtzeit zu verknüpfen, erfordert Zeit. Zeit, die wir nicht haben – die unsere Benutzer uns nicht geben möchten. Wer will heute schon noch länger als fünf Sekunden auf seine Suchergebnisse warten müssen? Wir brauchen diesen Wert – die Gesamt-

bewertung eines Bildes – für eine schnelle Suchfunktion also direkt bei den Bildern in der Tabelle *images*. Eine zusätzliche Spalte *average\_rating* für die Tabelle *images* muss also her, in der wir bei jeder Veränderung der Bewertung den neuen Durchschnittswert aus der Tabelle *images\_ratings* speichern (Tabelle 3).

Und vor genau diesem Schritt schreckt man als Entwickler gewöhnlich zurück – redundant (doppelt) gespeicherte Daten produzieren Probleme, wenn sie unsynchron sind. Und sie bedeuten einen erhöhten Pflegeaufwand und damit weitere Zeilen PHP-Code. Ein Schritt, der in einer logisch-abstrakten Welt mit unbegrenzter Rechenkapazität eigentlich nicht nötig wäre – doch leider kennen auch heutige Rechnersysteme noch deutliche Grenzen. Werfen wir also diese Skrupel beiseite und trauen wir uns, redundante Spalten anzulegen und zusammengefasste Werte zu speichern. Neben der deutlich erhöhten Geschwindigkeit unserer Suchabfrage bietet dies nämlich auch noch einen weiteren großen Vorteil: Unsere SQL-Statements zur Abfrage der Daten werden wesentlich einfacher und unkomplizierter – was wiederum Fehlerquellen minimiert.

### Tagging

Ein „Web 2.0“ ohne „Tagging“? Undenkbar! Die früheren „Keywords“ heißen heutzutage „Tags“ und zeichnen sich vor allem durch ihren stichwortartigen Charakter aus. Das macht sie universell einsetzbar und leicht wiederverwendbar, auch über die Grenzen des eigenen Angebots hinweg. Die amerikanische Social-Bookmarking-Website *del.icio.us* [4] oder die beliebte Blog-Suchmaschine Technorati [5] machen mit einer schier unglaublichen Anzahl von Einträgen und Tags vor, dass Suchfunktionen auch bei diesen enormen Ausmaßen noch schnell sein kön-

nen – ohne gleich eine Serverstruktur wie Google vorweisen zu müssen. Doch das ist wahrlich keine Selbstverständlichkeit: Etliche Millionen Tags müssen hier mit vielen Millionen Einträgen in Verbindung gebracht und ausgewertet werden. Dafür ist ein gut durchdachtes Datenbankschema notwendig, das diese Suchanfragen verkraftet und sie durch geschicktes Caching und Indexierung schnell macht. Alternativ bieten sich Lösungen an, die nicht auf relationalen Datenbanksystemen basieren, wie z.B. Lucence der Apache Software Foundation [6]. Das sollte jedoch erst bei sehr extremen Datenvolumina nötig sein, weshalb wir hier nicht näher darauf eingehen.

Allgemein wird hauptsächlich zwischen drei verschiedenen Datenbankschemen unterschieden, nämlich zwischen der völlig denormalisierten Form, einer Mischform und einer vollständig normalisierten Form. Bei der völlig denormalisierten Form werden alle Tags eines Objekts einfach in einer Spalte der selben Tabelle (*objects*) gespeichert, die alle Tags enthält. Diese kann dann mit einer Volltext-Suche nach einzelnen Tags durchsucht werden. Während sich diese Form für kleine Websites durchaus eignet, ist sie für größere Projekte nicht zu empfehlen. Hier sollte man zu einer stärker oder vollständig normalisierten Form übergehen. Bei der Mischform wird jedes Tag als einzelne Zeile in einer weiteren Tabelle *tags* gespeichert und über eine ID (einen Primary Key) mit den Objekten aus der Tabelle *objects* verknüpft, die vollständig normalisierte Form greift gar auf drei Tabellen zurück, versieht jedes Tag aus der Tabelle *tags* mit einer eigenen eindeutigen ID und schaltet noch eine weitere Tabelle *tags\_objects* dazwischen, die die einzelnen Tag-IDs mit den Objekt-IDs verknüpft. Dieses vollständig normalisierte Schema wird beispielsweise vom Blogging-System WordPress benutzt. Philipp Keller stellt diese Schemata in seinem Blog ausführlich vor und hat hierzu auch einige sehr interessante Benchmarks durchgeführt [7].

sevenload greift auf ein eigens konzipiertes Tagging-Schema zurück, das auf der Datenbank basiert. Es ist dem zuletzt beschriebenen Schema sehr ähnlich und

Tags liegen hierbei in vollständig normalisierter Form vor.

## Dateisalat

Einer der kompliziertesten Aspekte beim Umgang mit Bildern und Videos ist das Generieren und Verwalten etlicher verschiedener Vorschaubilder (neudeutsch: Thumbnails) in diversen Größen und Formen. Zwar wäre es möglich, diese Thumbnails in mittlerer Größe zum Client (Browser) des Benutzers zu senden und sie dann von diesem in die richtige Größe und Form pressen zu lassen, doch verschwenden wir damit sowohl Bandbreite als auch Qualität. Der Browser sollte nur die Daten bzw. Bildpixel empfangen müssen, die er für das fertige Rendering braucht. Doch das Generieren dieser Thumbnails on-the-fly, also während der Auslieferung der Seite, würde viel zu lange dauern. Deshalb müssen alle auf der Seite verwendeten und angebotenen Bildergrößen zuvor generiert und abgespeichert werden, um im Bedarfsfall blitzschnell ausgeliefert werden zu können. Diese Speicherung sollte unter keinen Umständen in der Datenbank erfolgen. Denn nur wenn die Bilder direkt im Dateisystem gespeichert werden, können weitere Ansätze zur Performance-Steigerung greifen. So lassen sich auf den Servern, die mit der Lagerung der Daten und der Auslieferung beauftragt sind, schlanke Apache-Alternativen installieren, die sich auf die Auslieferung von Dateien spezialisiert haben und sehr viel kleinere Ansprüche an Speicherverbrauch und CPU-Zeit stellen als der große Bruder Apache. Einer dieser schlanken Webserver ist Jan Kneschkes *lighttpd* [8].

## APIs, RSS und XML

Application Programming Interfaces kannte man bis vor wenigen Jahren vor allem als Entwickler konventioneller Software für Betriebssysteme. APIs sind Programmschnittstellen, die Applikationen für andere Applikationen und Entwickler anbieten, um Funktionalitäten miteinander kombinieren und gemeinsam nutzen zu können. Im Web 2.0 erleben diese offenen Schnittstellen nun auch ihren Einsatz über die Grenzen von Websites und Webservern hinweg. Auch sevenload greift dieses Konzept auf und

ist begeistert von der Idee und den Möglichkeiten, die eine offene Kollaboration für alle Beteiligten bietet. So können Entwickler, Anfänger wie Profis, Ihre Bilder und Videos bei sevenload auf vielfältige Weise in eigenen Websites und Projekte einbinden. Warum das Rad neu erfinden, wenn man bestimmte Teile eigener Projekte einfach und unkompliziert auslagern kann?

API-Schnittstellen können manchmal auch nur aus RSS-Feeds bestehen – eine genaue Definition gibt es nicht. Gemein haben die meisten Webservice-APIs, dass sie auf XML basieren. Hierbei haben sich drei Formate etabliert: REST, SOAP und XML-RPC. Das einfachste und für Einsteiger besonders gut geeignete Format ist REST, denn Anfragen werden über ganz gewöhnliche GET-Anfragen durchgeführt und Antworten kommen in sehr einfachem XML daher. SOAP und XML-RPC sind etwas anspruchsvoller, bieten jedoch mehr Flexibilität, insbesondere durch umfangreichere PUT-Anfragen, die ebenfalls in XML formuliert sind. Sowohl für SOAP [9] als auch für XML-RPC [10] stehen PEAR-Packages für PHP-Entwickler bereit.

Das A und O beim Anbieten solcher Webservice-APIs ist jedoch, den Benutzern absolute Verlässlichkeit garantieren zu können. Wer eigene Projekte von einer Plattform abhängig macht, beweist sehr viel Vertrauen und riskiert eigene Ausfälle bei Ausfällen dieser Schnittstellen. Bieten Sie solche Schnittstellen deshalb also nur an, wenn Sie für eine zukünftige und zeitlich nahezu unbegrenzte Erreichbarkeit garantieren können. Bei Änderungen und Erweiterungen an diesen Schnittstellen ist Abwärtskompatibilität zudem unverzichtbar.

## Fazit

So puristisch, simpel und offen das „Web 2.0“ daherkommt – die Entwicklung stabiler Plattformen für eine große Masse an Nutzern ist weiterhin ein Abenteuer. Doch insbesondere viele private Publikationen durch Blogger und der umfangreiche Austausch in Foren und auf Mailinglisten in Kombination mit aktuellen Open-Source-Lösungen machen Entwicklern und Unternehmern den Aufbau neuer Plattformen deutlich einfacher als noch vor vielen Jahren. Dahinter steht der Traum –Vieler, ge-

meinsam an etwas Großem zu arbeiten und sich dabei gegenseitig zu unterstützen – ein wichtiger Grundgedanke des „Web 2.0“.

Amerikanische Firmen wie del.icio.us, flickr oder YouTube [11] haben auch beim zweiten Anlauf der Startups die Nase vorn gehabt, doch ausländische Anbieter holen auf. sevenload ist „Web 2.0“-Pionier in Deutschland und freut sich auf eine aktive und innovative deutsche Gemeinde an Entwicklern und Unternehmern. Wir wünschen uns eine offene Kommunikation zwischen Benutzern, Homepagebetreibern, Entwicklern und Unternehmen und stehen jederzeit für Fragen, Lob, Kritik oder Kooperationen bereit.



Thomas Bachem ist Chief Software Architect bei der sevenload GmbH und leitender Entwickler der Front- und Backend-Systeme für sevenload.de.

## Links & Literatur

- [1] flickr – amerikanische Plattform für Fotos: [www.flickr.com](http://www.flickr.com)
- [2] PowerPoint-Slides einer Präsentation von Cal Henderson (Ludicorp) über die Entwicklung von flickr: [www.ludicorp.com/flickr/flickr\\_php\\_final.zip](http://www.ludicorp.com/flickr/flickr_php_final.zip)
- [3] Replikation bei MySQL: [dev.mysql.com/doc/refman/4.0/de/replication.html](http://dev.mysql.com/doc/refman/4.0/de/replication.html)
- [4] del.icio.us – amerikanische Plattform für Social Bookmarking: [del.icio.us](http://del.icio.us)
- [5] Technorati – Blog-Suchmaschine: [www.technorati.com](http://www.technorati.com)
- [6] Apache Lucene Project: [lucene.apache.org](http://lucene.apache.org)
- [7] Vergleich verschiedener Tagging-Schemata: [www.pui.ch/phred/archives/2005/04/tags-database-schemas.html](http://www.pui.ch/phred/archives/2005/04/tags-database-schemas.html)
- [8] lighttpd – ein schlanker Webserver: [www.lighttpd.net](http://www.lighttpd.net)
- [9] PEAR-Package für SOAP: [pear.php.net/package/SOAP/](http://pear.php.net/package/SOAP/)
- [10] PEAR-Package für XML-RPC: [pear.php.net/package/XML\\_RPC/](http://pear.php.net/package/XML_RPC/)
- [11] YouTube – amerikanische Plattform für Videos: [www.youtube.com](http://www.youtube.com)