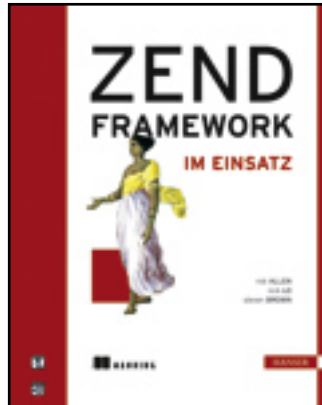


HANSER



Leseprobe

Rob Allen, Nick Lo, Steven Brown

ZEND Framework im Einsatz

Übersetzt aus dem Englischen von Jürgen Dubau

ISBN: 978-3-446-41576-8

Weitere Informationen oder Bestellungen unter

<http://www.hanser.de/978-3-446-41576-8>

sowie im Buchhandel.

12 Der Austausch mit anderen Applikationen

Die Themen dieses Kapitels

- Einführung in Webservices
- Einen RSS-Feed mit Zend_Feed erstellen und verarbeiten
- Einen Zend_XmlRpc-Server in eine Zend Framework-Applikation integrieren
- Einen REST-Server mit Zend_Rest erstellen

In diesem Kapitel beschäftigen wir uns mit den verschiedenen Komponenten des Zend Frameworks, die man lose unter dem Begriff „Webservices“ zusammenfassen kann. Diese definiert das World Wide Web Consortium (W3C) als „Softwaresystem, das eine interoperable Rechner-zu-Rechner-Interaktion über ein Netzwerk unterstützt“. Aus Gründen der Einfachheit werden wir unsere Verwendung des Begriffs „Webservices“ in diesem Kapitel auf dieser Definition gründen, anstatt auf dem spezielleren Schwerpunkt der Kombination von SOAP und WSDL (Web Services Description Language), den das W3C einnimmt.

Zu dieser Interoperabilität gehört zum Teil der Einsatz von XML, doch einer der Vorteile dieser Komponenten ist, dass wir uns nicht auf XML selbst konzentrieren müssen. Das Zend Framework enthält eine Reihe von Tools, die sich um die Formatierung und das Protokoll der Interaktion kümmern. So können Sie sich auf die Logik konzentrieren und brauchen dafür nur PHP. Nehmen Sie sich einfach mal einen Moment Zeit und zählen alle Formate auf, die nur für Newsfeeds im Netz verfügbar sind. Dann wird schnell der Vorteil offensichtlich, sich nicht mit dieser großen Bandbreite an Formaten und deren Änderungsrate herumschlagen zu müssen.

Bevor wir uns an den Einsatz der Zend Framework-Komponenten machen, schauen wir uns an, warum wir Applikationen anhand von Webservices integrieren wollen und auch sollten.

12.1 Die Integration von Applikationen

Es ist interessant, wie viele Webservices bereits zum Bestandteil unserer On- und Offline-Existenz geworden sind. Jedes Mal, wenn Nick seinen Rechner hochfährt, wird ein News-reader gestartet, der eine Liste mit XML-formatierten News von Sites abholt, bei denen er ansonsten nie auf dem Laufenden bleiben könnte. Kürzlich hat seine Frau einen ganzen Haufen Flohmarktsachen über GarageSale verkauft. Das ist eine Desktop-Applikation für Mac OS X, die über ihre auf XML basierende API mittels HTTP mit eBay spricht. Der Schlüssel zu all diesen Aktionen ist der Austausch von Informationen über ein Netzwerk und die Distribution der Rechenarbeit.

12.1.1 Austausch strukturierter Daten

XML steht für Extensible Markup Language und stammt von der Standard Generalized Markup Language (SGML) ab – einer der vielen Markup- oder Auszeichnungssprachen, deren Rolle einfach darin besteht, Text zu beschreiben. Die wahrscheinlich bekannteste ist HTML (Hypertext Markup Language). Diese Sprache beschreibt Textdokumente, die per HTTP übermittelt werden sollen.

Daten müssen nicht ausgezeichnet werden, um ausgetauscht werden zu können, doch in den meisten Fällen brauchen sie in irgendeiner Form eine Struktur. Hier sind einige Beispiele:

- Komma- oder tabulatorgetrennte Werte (CSV oder TSV)
- Strukturierter Text, dessen Struktur auf einer regelmäßigen Datensequenz basiert, die von konsistenten Begrenzern (*Delimiter*) getrennt werden.
- Serialisierte Daten wie solche, die von der PHP-eigenen `serialize()`-Funktion erstellt werden.
- Andere Formate wie die JavaScript Object Notation (JSON), die als Alternative zu XML in Ajax verwendet werden kann (und die im Zend Framework durch `zend_json` geleistet wird).

Diese Informationen sollten den Lesern dieses Buches kaum neu vorkommen, doch hier geht's darum, dass wir Informationen von einem System zu einem anderen übertragen wollen, sodass das empfangende System weiß, wie es mit diesen Daten umzugehen hat.

Wenn wir uns in Abbildung 12.1 die Applikation GarageSale anschauen, ist es eindeutig eine ziemlich komplexe Applikation, deren Daten nicht mit anderen ausgetauscht werden könnte, außer sie sind passend strukturiert, damit eBay sie verarbeiten und die jeweiligen Anfragen ausführen kann, z. B. ein neues Element für eine Auktion zu erstellen.

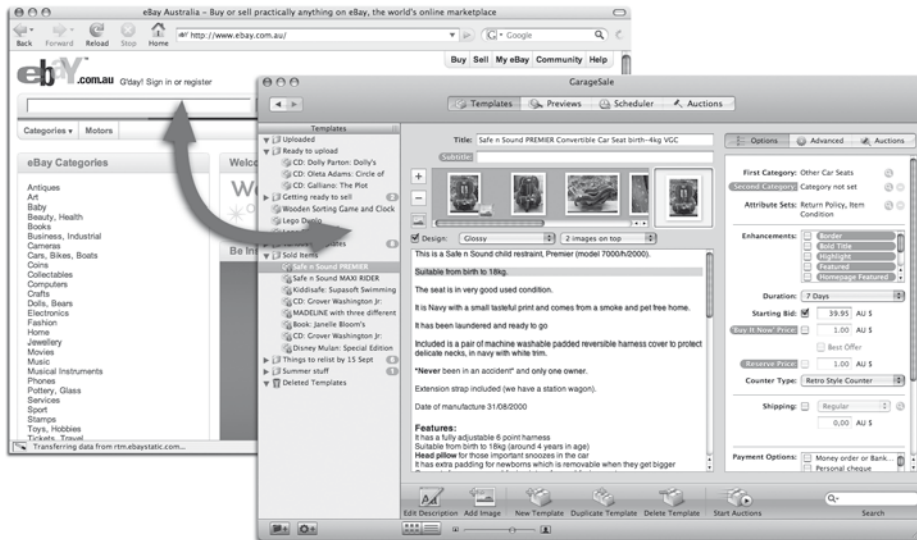


Abbildung 12.1 Die Desktop-Applikation GarageSale unter Mac OS X spricht anhand von Webservices mit der auf XML basierenden API von eBay.

Nach einem Blick auf die Formatierung der an dieser Diskussion beteiligten Daten zwischen den Applikationen lautet die nächste Frage, wie die Applikationen miteinander kommunizieren.

12.1.2 Produktion und Verarbeitung strukturierter Daten

E-Mails dienen als gutes Beispiel dafür, wie Applikationen mittels eines strukturierten Datenformats (Internet-E-Mail-Format) miteinander sprechen können. Dieses Datenformat wird an dem einen Ende produziert, über einen Mailserver (MTA) versendet und dann auf Empfängerseite vom E-Mail-Client (MUA) verarbeitet. Diese „Gespräche“ zwischen den Applikationen, die das Thema dieses Kapitels sind, treiben dieses Grundkonzept einen Schritt weiter, indem durch diesen Austausch auf den jeweiligen Seiten Aktionen ausgelöst werden, und zwar durch einen sogenannten Remote Procedure Call (RPC).

Im weiteren Verlauf dieses Kapitels wird es um `Zend_XmlRpc` gehen, das mit `Zend_XmlRpc_Server` arbeitet, um XML-kodierte RPCs über HTTP zu senden und zu empfangen. Ursprünglich geschaffen von Dave Winer, ist XML-RPC eine überraschend einfache und flexible Spezifikation, die deswegen in ganz verschiedenartigen Situationen eingesetzt wird.

Als weitere Funktionalitäten eingebaut wurden, entwickelte sich XML-RPC zu SOAP (ursprünglich ein Akronym, aber jetzt einfach nur ein Wort), das auch vom W3C adaptiert wurde. Sie brauchen sich nicht lange umzuhören, bis Sie erfahren, zu welchen Beschwerden diese erweiterten Funktionalitäten wegen der höheren Komplexität führten. Das erklärt teilweise, warum trotz der offiziellen Anerkennung von SOAP durch das W3C weiterhin

XML-RPC verwendet wird. In mancherlei Hinsicht ist SOAP selbst schon von anderen Protokollen wie Atom abgelöst worden. Auch das Zend Framework spiegelt den Status dieser Protokolle wider. `zend_soap` selbst dümpelte über zwei Jahre im Inkubator herum, ehe es endlich abgeschlossen wurde – das lag weitgehend am mangelnden Interesse der User, während XML-RPC und Atom beide in den Core aufgenommen wurden.

Wir schauen uns die Komponenten des Zend Frameworks gleich an, doch vorher soll noch erläutert werden, wie Webservices funktionieren und warum wir damit arbeiten wollen.

12.1.3 Wie Webservices arbeiten

Die Einfachheit von Webservices präsentiert man am einfachsten über eine Illustration wie in Abbildung 12.2. Ganz einfach ausgedrückt arbeiten Webservices wie viele andere Methoden zur Datenübermittlung: Daten werden am einen Ende formatiert (z. B. mit XML-RPC ins XML-Format), dann über ein Protokoll übertragen (in diesem Fall HTTP) und schließlich am anderen Ende weiterverarbeitet.

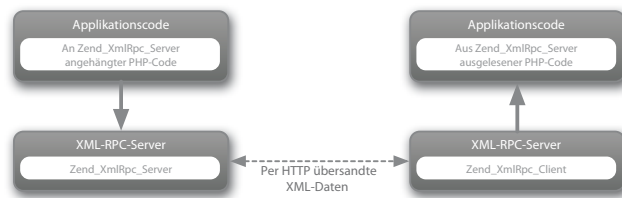


Abbildung 12.2 Die grundlegende Transaktion eines Webdienstes zwischen zwei Systemen mit XML-RPC

Natürlich ist diese Erklärung so allgemein gehalten, dass sie praktisch wertlos ist. Um also noch besser zu veranschaulichen, wie Webservices arbeiten, wollen wir den Schritten eines XML-RPC-Beispiels folgen, bei dem eine Desktop-Applikation aktualisierte Preise von einem Onlinedienst bezieht:

1. Eine Desktop-Applikation holt die Daten, die für einen Procedure-Aufruf benötigt werden (einschließlich der ID des angeforderten Artikels und der Remote Procedure, die die Preise für die Artikel holt). Die XML-RPC-Client-Komponente der Desktop-Applikation kodiert diesen Remote Procedure Call ins XML-Format und sendet ihn wie folgt an den Onlinedienst:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>onlineStore.getPriceForItem</methodName>
  <params>
    <param>
      <value><i4>123</i4></value>
    </param>
  </params>
</methodCall>
```

- Der XML-RPC-Server des Onlinedienstes empfängt den XML-kodierten Procedure-Aufruf und dekoriert ihn in ein Format, das der Systemcode verarbeiten kann, z. B. `$store->getPriceForItem(123)`. Der Systemcode gibt den angeforderten Preis an den XML-RPC-Server zurück, der dies als XML-Response kodiert und ihn an die anfordernde Desktop-Applikation zurückschickt:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><double>19.95</double></value>
    </param>
  </params>
</methodResponse>
```

- Die Desktop-Applikation empfängt die Antwort, dekodiert sie in ein Format, das sie verarbeiten kann, und aktualisiert den Preis für Artikel 123 auf 19,95 €.

So haben Sie eine ungefähre Vorstellung von der Funktionsweise von Webservices, doch die Frage bleibt, warum wir damit arbeiten sollten.

12.1.4 Aufgabengebiete für Webservices

Wofür brauchen wir Webservices? Die einfache Antwort ist auch die ironischste: Wir brauchen Webservices, damit Applikationen, die auf unterschiedlichen Plattformen oder Frameworks laufen, miteinander auf standardisierte Weise sprechen können. In diesem Kapitel wurde bereits die Ironie dieses Konzepts angesprochen, indem wir Sie mit einer kleinen Auswahl der verschiedenen Protokolle verwirrt haben, aus denen diese „Standards“ bestehen.

Lassen wir die Ironie mal beiseite und kehren zum Beispiel unserer Desktop-Applikation zurück, die aktualisierte Preise von dem Onlinedienst abholt: Ihnen wird auffallen, dass es nur wenige Details darüber gab, wie jedes Ende der Transaktion diese Procedure-Aufrufe tatsächlich durchführt. Der Grund ist, dass es egal ist, denn solange jede Applikation in der Lage ist, ihre internen Prozesse in eine Standardform der Kommunikation umzuwandeln (das XML-RPC-Protokoll), kann die Transaktion abgeschlossen werden.

Das kann eindeutig zu sehr leistungsfähigen Interaktionen führen wie z. B. einer solchen zwischen GarageSale und eBay. In diesem Kapitel und dem nächsten werden wir uns mit Beispielen beschäftigen, wie Komponenten des Zend Frameworks einige der Komplexitäten von Webservices umgehen können, damit solche Interaktionen vorteilhaft eingesetzt werden können.

Wir beginnen mit einem Beispiel, das den meisten Lesern wahrscheinlich vertraut sein wird: Web-Feeds und die Vorteile von `Zend_Feed`, um Feeds zu produzieren und zu verarbeiten.