Entwickler Magazin 1.12

inkl. CD!

OAuth im praktischen Einsatz >>> S. 72!

S&S

Deutschland € 6,50 Österreich € 7,00 Schweiz sFr 13,40

entwickler magazin

www.entwickler-magazin.de

Januar/Februar 1.2012

CD-Inhalt

GoClipse: Eclipse-Plug-in zur Erweiterung der Eclipse-IDE mit Googles Go

OpenLayers 2.11: JavaScript-Bibliothek zur Anzeige von Geodaten im Browser

Apache CouchDB 1.1.1: Dokumentenbasiertes Datenbanksystem

Talend Open Studio 4.2.3: ETL-Tool

Video von der BASTA! 2011: Sharding – der Turboboost für Ihre Datenbanken von Rainer Stropek

Bonus:
Zusätzlich finden Sie auf unserer aktuellen Magazin-CD ausgewählte Tools sowie alle Quellcodes und Beispiele zu den Artikeln im Heft

▶▶▶ Alle Infos auf Seite 3

Weitere Artikel

Grundlagen und Methoden
Scrum Basics

Der 2. Usability-Test
UML-Editoren

Geodaten im Browser
OpenLayers

CouchDB-Apps ohne Aufwand
CouchApp mit erica

4 194156 606509 01

Datenträger enthält Info- und Lehrprogramme gemäß §14 JuSchG

User Interfaces

Schnittstellen richtig integriert

Der große Überblick

AWS, Azure & die App Engine

Zeit ist Geld. So sparen Sie beides!

Prototyping

Google Go

Parallelisierung mit Googles neuer Sprache

UML • Delphi • OpenLayers • CouchDB • Prototyping • Go • AWS • Azure • App Engine • Cloud • UI • OAuth • Kanban • Scrum

Einführung in die Applikationsentwicklung mit CouchDB

App auf die Couch mit erica

Der Wunsch, eine Webapplikation, die nur ein Backend benötigt, auf einfachste Art und Weise zu entwickeln, wird mit CouchDB und CouchApp erfüllt. Denn CouchDB ist Datenbank, Webserver und Applikationsserver in einem. Das aus dieser Kombination entstehende Stück Software ist auch bekannt als CouchApp.

von Andy Wenk

J. Chris Anderson [1], Core-Entwickler von Apache CouchDB [2] und Mitbegründer der Firma Couchbase, hatte bereits bei der Erscheinung der CouchDB-Version 0.9 die Idee, CouchDB als Applikationsserver zu nutzen. Er beabsichtigte, CouchDB als einziges Backend für das Bereitstellen von Webapplikationen zu nutzen (CouchDB Embedded Applications). Dafür hat er angefangen, ein Tool in Ruby zu schreiben, das als CLI dazu dienen sollte, den Prozess der Übertragung von HTML-, CSS- und JavaScript-Dateien in die CouchDB zu vereinfachen. Darüber hinaus sollte es eine einfache Scaffolding-Funktionalität besitzen (das automatische Erstellen einiger essenzieller Dateien in einer vorgegebenen Struktur). Nachdem Anderson die Zeit fehlte, das Tool weiterzuentwickeln, hat Benoit Chesneau [3] die Aufgabe übernommen und dabei den Quellcode nach Python portiert. CouchApp [4] war geboren. Besuchen Sie unter [5] die offizielle Website von CouchApp. In diesem Artikel lesen Sie eine Einführung in die Entwicklung von CouchApps, unter Zuhilfenahme des hervorragenden Tools *erica* [6], ebenfalls entwickelt von Benoit.

Konzept

Wie immer sollte sich zu Anfang der Entstehung einer Applikation ausführlich darüber Gedanken gemacht werden, was die Software leisten soll. Natürlich kann hier nur ein sehr einfaches Beispiel gezeigt werden, aber es wird genügen, die Prinzipien für die Erstellung von CouchApps zu verstehen. Die CouchApp, die wir erstellen, soll ein einfaches Board für Eventtermine sein. Die Repräsentation eines Dokuments finden Sie in Listing 1. Die Datenbank soll *event_board* heißen und lässt sich

Abb. 1: Die frisch erstellte CouchApp

über *curl* oder das CouchDB-Webinterface *Futon* einfach erstellen (**Abb. 1**): *curl -X PUT http://127.0.0.1:5984/event_board*. Natürlich benötigen wir auch ein Layout für unser Event-Board. An dieser Stelle kommen wir bereits zum Einsatz von erica. Sie können den gesamten Quellcode des hier gezeigten Beispiels unter [7] herunterladen oder klonen.

erica

erica ist ein CLI-Tool, das Ihnen das Leben beim Erstellen von CouchApps erleichtern soll. Der Vergleich zu *couchapp* zeigt einen markanten Unterschied: erica verfügt über ein eigenes, webbasiertes User Interface, das im entsprechenden Projekt gestartet werden kann. Darin haben Sie die Möglichkeit, die einzelnen Projektdateien mithilfe des integrierten Editors zu editieren, meiner Ansicht nach ist das ein sehr großer Vorteil gegenüber *couchapp*. Außerdem ist erica in Erlang geschrieben, wohingegen *couchapp* in Python programmiert wurde. Sehen wir uns zuerst die Installation von erica an (beachten Sie bitte, dass Erlang R14 [8] und gcc [9] auf dem System installiert sein müssen):

```
$ git clone git://github.com/benoitc/erica.git
$ cd erica
$ make
```

Nach erfolgreichem Ausführen dieser Befehle finden Sie ein Programm namens erica im aktuellen Verzeichnis. Auf einem Unix-basierten System ist es sinnvoll, es an einen Ort zu legen, der in der Umgebungsvariable *PATH* referenziert wird (auf dem Mac z. B. */Users/username/bin/*). Wie üblich erhalten Sie beim Aufruf des Programms mit der Option „-h“ einen Hilfetext. Sie sollten sich auf alle Fälle die einzelnen zur Verfügung stehenden Kommandos ansehen. Der Aufruf erfolgt durch die Option „-c“ (Tabelle 1). Zu gegebener Zeit werden wir auf die einzelnen Kommandos näher eingehen. Im nächsten Schritt erstellen wir erst einmal das Grundgerüst unserer Event-Board-Applikation.

Die event_board-App erstellen

Eine hervorragende Hilfe für das Erstellen von CouchApps mit erica ist die Nutzung des Kommandos *create* unter Angabe eines Templates. erica bietet von Haus aus drei Templates:

- *simpleapp*: sehr einfache Struktur als leerer Applikationscontainer
- *example*: komplexere Struktur mit einem ersten Template
- *couchapp*: Struktur wie bei der Nutzung von *couchapp*

Wir greifen hier gleich in die Vollen und nutzen als Template *example*. Wechseln Sie in ein Verzeichnis, in dem Sie gleich den Applikationsordner der CouchApp

Listing 1

```
{
  "_id": "50af4493a1b5b9c5731458f834000f80",
  "_rev": "1-4e620424d842528a812d09e7120fb590",
  "type": "event",
  "date": "2011-11-18",
  "location": "Indra",
  "city": "Hamburg",
  "info": "Punk-Rock Show with the Naked Mermaids On Water Ski",
  "price": 7,
  "tags": "music,punk,rock"
}
```

Listing 2

```
==> erica (create)
Writing event_board/_id
Writing event_board/.couchapprc
Writing event_board/language
Writing event_board/README.md
Writing event_board/couchapp.json
Writing event_board/views/recent-items/map.js
Writing event_board/_attachments/index.html
Writing event_board/_attachments/script/app.js
Writing event_board/_attachments/style/main.css
Writing event_board/_attachments/couchapp/jquery.couch.app.js
Writing event_board/_attachments/couchapp/jquery.couch.app.util.js
Writing event_board/_attachments/couchapp/jquery.couchForm.js
Writing event_board/_attachments/couchapp/jquery.couchLogin.js
Writing event_board/_attachments/couchapp/jquery.couchProfile.js
Writing event_board/_attachments/couchapp/jquery.mustache.js
Writing event_board/_attachments/couchapp/md5.js
```

Option	Optionen	Beschreibung
init		initialize a erica
push	[options...] [dir] dest	push a document to couchdb
clone	[option] source dir	clone a document from couchdb
browse		display the erica in the browser.
web	port=Port [dir]	launch the web ui
create-app	appid=AppID lang=Lang	Create a blank couchapp, Default: appid=myapp, lang=javascript
create	template= [vars...]	create an application using a template
help		Show the program options
version		Show version information

Tabelle 1: Die für erica zur Verfügung stehenden Optionen

erstellen wollen. Und dann geht es los: *$ erica create template=example appid=event_board lang=javascript*. Wie sie in Listing 2 sehen, wird mit dem Aufruf eine relativ umfangreiche Struktur erstellt – um genau zu sein fast alles, was für eine einfache Applikation benötigt wird. Leider kann hier nicht auf alle Dateien einzeln eingegangen werden, aber es ist nicht schwer, im Web gute Dokumentationen zu finden. Oder Sie lesen nähere Informationen zu CouchApps im Buch des Autors zu CouchDB [10].

Diese Struktur wird uns also als Grundlage für die *event_board*-CouchApp dienen. Die Hauptaufgabe des CLI erica ist (oder analog das CLI-Tool *couchapp*), wie bereits erwähnt, die Vereinfachung des Transfers der Dateien von der Entwicklungsumgebung in eine CouchDB. Dazu dient das Kommando *push* unter der Angabe des URL der CouchDB-Datenbank, in die die App gelegt werden soll. Diese Datenbank muss übrigens vor dem ersten *push* nicht bestehen, sondern kann von erica erstellt werden. Doch bevor wir die Dateien in die CouchDB pushen, wollen wir uns alles einmal im Onlineeditor ansehen und gegebenenfalls bearbeiten. Dazu wechseln wir in das gerade erstellte Verzeichnis */event_board/* und starten einen Webserver:

```
$ cd event_board
event_board$ erica web port=3001
==> event_board (web)
Erica Web started on "http://127.0.0.1:3001"
```

Öffnen Sie nun einen Browser und rufen Sie den URL http://127.0.0.1:3001 („port=3001“, falls eine Rails-Applikation unter Port 3000 läuft, **Abb. 2**). Sie sehen als Ergebnis die Grundstruktur der Applikation.

Die CouchApp liegt im Ordner */_attachments/*. Das bedeutet, dass hier alle relevanten Dateien wie HTML, CSS und JavaScript zu finden sind. Im Ordner */views/* werden die *map*- und *reduce*-Funktionen in jeweils einer Datei abgelegt: *map.js* und *reduce.js*. Beide Dateien werden dabei in einem Ordner abgelegt, der den Namen der View beinhaltet. In *.couchapprc* werden Konfigurationseinstellungen für diese CouchApp eingetragen. Das könnten zum Beispiel die URLs für Server in unterschiedlichen Umgebungen sein (Development, live etc.). *README.md* ist ein in Markdown [11] geschriebenes Dokument, das allgemeine Informationen über diese CouchApp beinhaltet.

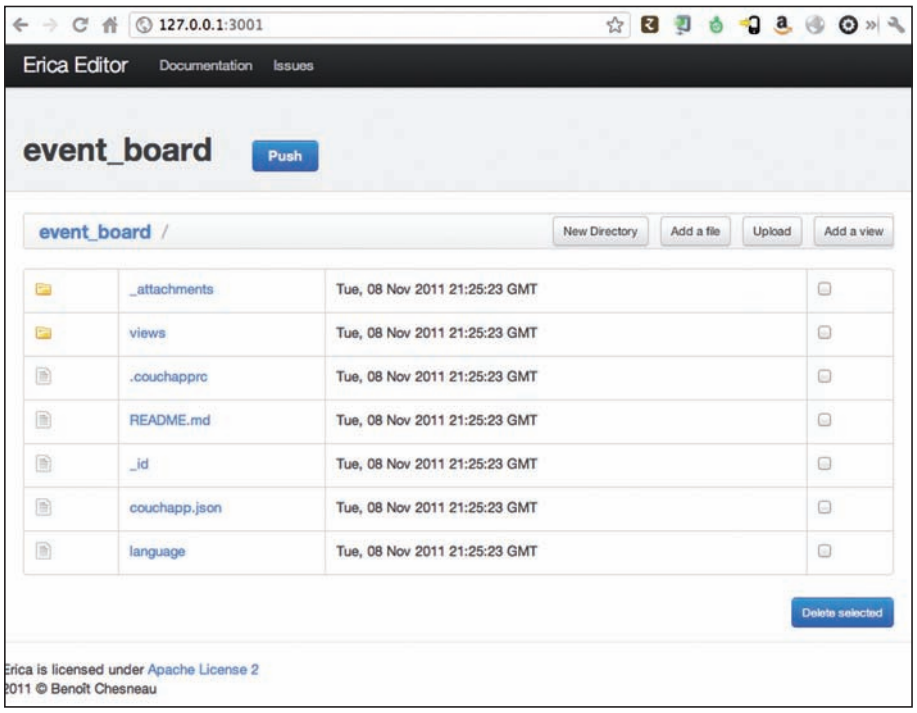

Abb. 2: Die Grundstruktur der Applikation

In *_id* wird der Name des Designdokuments (hier *event-board*) abgelegt. *couchapp.json* ist als Manifest für die CouchApp zu verstehen – ähnlich wie bei Rubygem. Und schließlich wird in der Datei *language* festgehalten, in welcher Sprache die *map*- und *reduce*-Funktionen geschrieben sind. Um zu sehen, wie dieses Template aussieht, muss die CouchApp in die Datenbank übertragen werden. Wir nutzen dabei den *push*-Mechanismus von erica. Um allerdings nicht jedes Mal den Ziel-URL eingeben zu müssen, fügen wir in die Konfigurationsdatei *.couchapprc* ein Ziel namens *development* ein:

```
{
  "env": {
    "development": {
      "db": "http://127.0.0.1:5984/event_board"
    }
  }
}
```

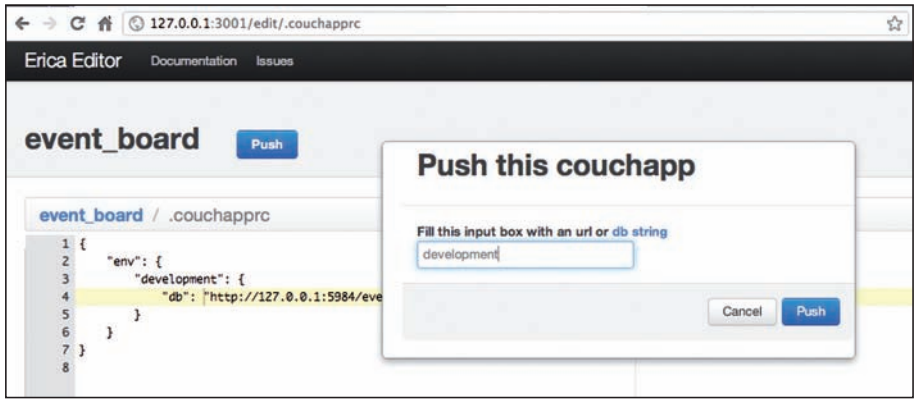

Abb. 3: Der Push-Dialog

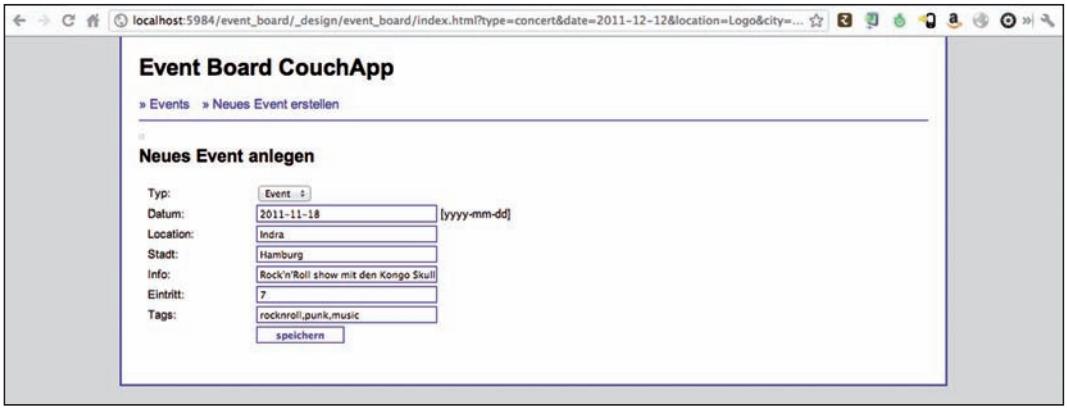

Abb. 4: Formular für neue Events

Abb. 5: Übersicht über erstellte Events

Danach wird der Push-Vorgang gestartet, indem der Knopf Push geklickt wird (**Abb. 3**). In der sich öffnenden Lightbox geben Sie als Ziel einfach den Namen der Umgebung ein: in unserem Beispiel *development*. So sollte im Grunde die Datenbank *event_board* erstellt werden. Allerdings liegt hier in der erica-Version 0.2 ein Bug vor, denn es wird fälschlicherweise eine Datenbank namens *development* erstellt. Entweder Sie geben in der Lightbox den URL direkt ein oder Sie führen den Befehl mit dem CLI aus, *$ erica push development*, oder Sie warten, bis Benoit den Fehler behoben hat, was nicht die beste Wahl ist. Ich habe den Fehler unter [12] an Benoit gemeldet; wahrscheinlich ist er bei Erscheinen dieses Artikels bereits behoben. Sehen wir uns jetzt das Ergebnis im Browser an. Rufen Sie dazu den URL http://127.0.0.1:5984/event_board/_design/event_board/index.html auf.

Listing 3

```
function(doc) {
  if (doc.type === "event") {
    emit(doc.date, {
      id: doc._id,
      rev: doc._rev,
      location: doc.location,
      city: doc.city,
      info: doc.info,
      price: doc.price,
      tags: doc.tags,
      type: doc.type
    });
  }
};
```

Die event_board-App anpassen

Das Ergebnis der Push-Aktion zeigt, wie einfach es ist, eine CouchApp zu erstellen. Der *push*-Befehl ist hier eine zentrale Aktion, die Sie immer wieder ausführen werden, entweder auf der Kommandozeile oder im erica-Webinterface. Im nächsten Schritt passen wir nun einige Dateien an, um schließlich einige Events in unserem Board anzeigen zu können. Beim Ansehen der Dateien der generierten CouchApp fällt auf, dass in der Datei *index.html* ziemlich viele JavaScript-Dateien eingebunden werden. Zudem liegt hier auch noch ein kleiner Fehler vor, der wohl durch die Portierung von *couchapp* zu erica geschehen ist. Der Pfad */vendor/couchapp/* existiert noch nicht. Löschen Sie einfach */vendor*, dann ist das Problem gelöst. Weiterhin brauchen wir von den vorhandenen JavaScript Libraries nur *json2.js*, *jquery.js*, *jquery.couch.js* und *jquery.mustache.js*. Die restlichen können Sie für diese App löschen. Das Ziel der App ist es, eine Liste von Events anzuzeigen und ein Formular zu haben, mit dem neue Events aufgenommen werden können. Das Löschen einzelner Events wäre auch nicht schlecht. Wir brauchen also zwei unterschiedliche Ansichten. In den **Abbildungen 4** und **5** finden Sie einen Vorschlag für beide.

Gesteuert wird die Applikation durch die Datei *event-board/_attachements/script/app.js*. Darin finden Sie ein Objekt namens *Events*. Eine darin enthaltene Methode *init()* wird ausgeführt, sobald der DOM geladen ist. Danach wird auch schon die Methode *show()* aufgerufen, die nach dem Aufruf einer View mithilfe von *mustache.js* eine Liste erstellt. Die *Map*-Funktion für die View befindet sich unter *event_board/views/events/map.js*. Darin werden alle benötigten Daten abgefragt (Listing 3). Diese View liefert alle Dokumente zurück, die vom Typ *event* sind.

Das Formular ist sehr einfach aufgebaut und bedarf keiner großartigen Erklärung. Die Navigation ist ebenfalls sehr einfach, wobei die Click-Events in der Methode *register_event_handler()* abgefangen werden und dann die entsprechende Ansicht generiert wird. Hier werden auch die Methoden zum Speichern und Löschen von Einträgen getriggert. Diese Arbeitsweise ist für kleine Applikationen absolut legitim, da die Arbeit komplett im Client passiert. Solch eine Art Applikation nennt man auch One-Page-App. Natürlich können diese Methoden auch durch *Show*- und *List*-Funktionen (siehe meine vorhergehenden Artikel zum Thema CouchDB) realisiert werden. Um mit der CouchDB zu kommunizieren, wird hier auf das mitgelieferte API aus */_utils/script/jquery.couch.js* zurückgegriffen. Dieses API ist

zwar schon etwas in die Tage gekommen, erfüllt aber den Zweck. Jedes Mal, wenn Sie Anpassungen an der App vornehmen, müssen Sie sie in die CouchDB pushen. Dazu rufen Sie am einfachsten das Webinterface von erica auf. Alternativ können sie natürlich auch das CLI nutzen. Das Ergebnis ist identisch.

Überblick

Wir haben bis hier ziemlich wenige Schritte ausgeführt, um zu einer einfachen CouchApp zu gelangen. Allerdings ist das Ergebnis doch ganz beachtlich. Sie könnten aus dieser Vorlage und mit ein paar Erweiterungen sehr einfach ein richtig nettes Event-Board bauen. Eine gute Sache wäre ja zum Beispiel eine Registrierung für Benutzer, damit diese eigene Boards erstellen können. Letztendlich haben wir nur die Dateien *app.js*, *index.html*, *map.js* und *main.css* angepasst, um zum Ergebnis zu kommen.

Fazit

CouchDB ist cool. Und CouchApp ist es auch. Und weil alle guten Dinge drei sind, erica ist auch cool. Sie haben in diesem Artikel gesehen, wie einfach es ist, eine CouchApp zu bauen. Natürlich gibt es wie immer ein paar Voraussetzungen, zum Beispiel das relativ neue Erlang R14. Aber diese Hürde sollte auf allen Systemen zu nehmen sein. Die hier gezeigte Einfachheit bei der Entwicklung von dynamischen und datenbankgestützten Applikationen legt nahe, dass die prototypische Entwicklung von Projekten damit sehr gut realisierbar ist. Versuchen Sie das doch einmal in einem Ihrer nächsten Probleme. Aber immer relaxt!

Andy Wenk ist Software Developer bei SinnerSchrader in Hamburg, hat dort viel Spaß mit JavaScript und Ruby on Rails und findet CouchDB cool. Zusammen mit Till Klampaeckel (@klimpong) aus Berlin hat er gerade eins der ersten deutschsprachigen CouchDB-Bücher beim Galileo Computing Verlag fertig gestellt. Sie erreichen ihn unter andy@nms.de und @awenkhh.

Links & Literatur

[1] http://jchrisa.net/

[2] http://couchdb.apache.org

[3] https://github.com/benoitc

[4] https://github.com/couchapp/couchapp

[5] http://couchapp.org/

[6] https://github.com/benoitc/erica

[7] https://github.com/andywenk/event_board_couchapp_erica

[8] http://www.erlang.org/

[9] http://gcc.gnu.org

[10] http://www.couchdb-buch.de

[11] http://daringfireball.net/projects/markdown/

[12] https://github.com/benoitc/erica/issues/15

Vorschau auf die Ausgabe 2.2012

Von Magiern, Mathematikern und Informatikern

Auch vor der Codewelt macht das Rechnen mit großen Zahlen nicht halt. Mit externen Bibliotheken verschafft man sich Zugang zur Kalkulation großer Zahlen, und viele Systeme wie Kryptologie, Astrologie oder Umweltsimulation nutzen vermehrt große und in der Regel auch ganze Zahlen. Lassen Sie sich im Folgenden vom Umfang der schieren Unendlichkeit überzeugen und die hohe Flexibilität der Algorithmen zeigen.

RIA goes Mobile

Wer nicht für jede mobile Plattform eine eigene native App entwickeln möchte, kommt an Webapplikationen für Smartphones nicht vorbei. Rich Internet Applications (RIA) bieten den von nativen Apps gewohnten Bedienungskomfort und sind nach dem Start gegen Netzabbrüche gefeit, da der Client lokal auf dem Endgerät abläuft. Auch der Entwickler profitiert von diesem Ansatz, denn RIA-Frameworks wie Sencha Touch bieten hohe Qualität und Produktivität in Kombination mit einem Programmiermodell, das einfach Spaß macht. Dieser Artikel beschreibt das Konzept und die Implementierung einer Sencha Touch App und bietet einen leichten Einstieg in die Arbeit mit dem Framework.

Aus aktuellem Anlass kann es zur Verschiebung von Artikeln kommen.

Die nächste Ausgabe erscheint am 15. Februar 2012

Querschau

PHPmagazin — Ausgabe 1.2012 | www.phpmagazin.de

The final Countdown: Neues von FLOW3

Wo bin ich: Geolocation und Geocoding mit PHP

android360 — Ausgabe 4.2011 | www.android360.de

Sicher mit A: Das Android Backup API

Lecker Ice Cream Sandwich: Android 4.0 unter der Lupe

Inserenten

Impressum

Verlag:
Software & Support Media GmbH

Anschrift der Redaktion:
Entwickler Magazin
Software & Support Media GmbH
Geleitsstraße 14
D-60599 Frankfurt am Main
Tel. +49 (0) 69 6300890
Fax. +49 (0) 69 63008989
redaktion@entwickler-magazin.de
entwickler-magazin.de

Chefredakteur: Sebastian Meyen
Redaktion: Thomas Wießeckel, Diana Kupfer, Nicole Bechtel
Chefin vom Dienst/Leitung Schlussredaktion: Nicole Bechtel
Schlussredaktion: Katharina Klassen, Frauke Pesch, Lisa Pychlau
Leitung Grafik & Produktion: Jens Mainz
Layout, Titel: Tobias Dorn, Flora Feher, Pobporn Fischer, Karolina Gaspar, Dominique Kalbassi, Laura Keßler, Maria Rudi, Petra Rüth, Franziska Sponer
CD/DVD-Erstellung: Daniel Zuzek, Özkan Peksan

Autoren dieser Ausgabe:
Dr. Stefan Bente, Dr. Mario Deilmann, Carsten Eilers, Christopher Ezell, Sebastian P. R. Gingter, Stefan Göppert, Christian Himpel, Stefan Janisch, Rudolf Jansen, Jurij Kalina, Dr. Veikko Krypczyk, Bernd Ott, Frank Pientka, Andreas Pihan, Andy Transchel, Andy Wenk, Torsten Zimmermann

Anzeigenverkauf:
Entwickler Magazin
Patrik Baumann
Software & Support Verlag GmbH
Tel. +49 (0)69 630089-20
pbaumann@entwickler-magazin.de

Open Source
Verlagsbüro Ohm-Schmidt
Osmund Schmidt
Schneckenburger Str. 22
30177 Hannover
Tel. +49 (0)511 2354164
Fax: +49 (0)1805 06033695669
E-Mail: osmund@ohm-schmidt.de

Es gilt die Anzeigenpreisliste Mediadaten 2011

Pressevertrieb:
DPV Network GmbH
Tel.:+49(0)40 23711 0
www.dpv-network.de

Druck: PVA, Landau

Abonnement und Betreuung:
Leserservice Entwickler Magazin
65431 Eltville
Tel.: +49 (0)6123 9238-239
Fax.: +49 (0)6123 9238-244
entwicklermagazin@vuservice.de

Abonnementpreise der Zeitschrift (inkl. Leser-CD):

Inland:	6 Ausgaben	€ 29,50
Studentenpreis:	6 Ausgaben	€ 23,60
europ. Ausland:	6 Ausgaben	€ 39,50
Stud. europ. Ausland:	6 Ausgaben	€ 33,60

Abonnementpreise der Zeitschrift
(inkl. Leser-CD plus Profi-CD):

Inland:	6 Ausgaben & CD	€ 99,-
Studentenpreis:	6 Ausgaben & CD	€ 80,-
europ. Ausland:	6 Ausgaben & CD	€ 109,-
Stud. europ. Ausland:	6 Ausgaben & CD	€ 90,-

Abonnementpreise der Profi-CD:

Inland:	6 CD-ROM	€ 72,-
Studentenpreis:	6 CD-ROM	€ 62,-
europ. Ausland:	6 CD-ROM	€ 82,-
Stud. europ. Ausland:	6 CD-ROM	€ 72,-

ISSN: 1619-7941

Erscheinungsweise: zweimonatlich