Notes Apps auf iPad & Co

Wie native Apps mit Notes synchronisieren können.

Julian Buß, jbuss@julianbuss.net

http://julianbuss.net

Heute:

Beispiele mit echtem Code,
Erfahrungen aus dem
echten Leben,
Überlegungen zur Performance

Beispiel 1: Ansicht lesen & in App speichern

Live-Demo

was wir gesehen haben

- JSON Daten via SSJS erzeugen und mittels XPage ausgeben
  Stichworte: rendered=false, AfterRenderResponse Event, Java Faces Writer
- Domino eigenes JSON erhält diverse Daten, die uns nichts nützen,
  deswegen ist ein eigenes Format effizienter
- Lesen der JSON Daten via HTTP in Titanium, speichern in SQL Tabelle

Beispiel 2: Dokumente lesen, in App speichern & verwenden

Live-Demo

was wir gesehen haben

- Notes-Dokument zu XML „konvertieren“
- XML mittels XPage ausgeben und in Titanium via HTTP lesen
- XML in SQL DB speichern
- XML aus SQL DB lesen und Notes-Items (Felder) extrahieren

Beispiel 3: HTML Repräsentation eines Dokuments lesen, speichern, anzeigen

Live-Demo

was wir gesehen haben

- Notes-Dokument nach HTML konvertieren, Layout abhängig von Maske
- HTML in Titanium via HTTP lesen und speichern
- HTML in der App darstellen

Titanium wie LotusScript

- Arbeit mit SQL ist für viele Notes Entwickler neu...
- ...und häufig auch irgendwie unbequem :-)
- Wie wäre es mit Notes-Klassen wie in LotusScript?

Beispiel aus Session 1

Standard Titanium Code

```
1  var db = Ti.Database.open("contacts");
2  var dbRows = db.execute("SELECT * FROM view_people");
3  while (dbRows && dbRows.isValidRow()) {
4
5      var tableRow = {
6          id: dbRows.fieldByName("id"),
7          title : dbRows.fieldByName("name")+", "+dbRows.fieldByName("email")
8      };
9      tableData.push(tableRow);
10
11     dbRows.next();
12 }
13 dbRows.close();
14 db.close();
15 table.setData(tableData);
```

Gleiche Logik, Notes-Style

„Fremde“ SQL Logik in Notes-artigen Klassen verpackt

```
1  var table = Ti.UI.createTableView();
2  var tableData = [];
3  var notesdb = new NotesDatabase("contacts");
4  var notesView = notesdb.getView("people");
5  var vec = notesView.allEntries();
6  var ve = vec.getFirstEntry();
7  while (ve) {
8      var tableRow = {
9          id: ve.universalID,
10         title : ve.getColumnValue("name")+", "+ve.getColumnValue("email")
11     };
12     tableData.push(tableRow);
13     ve = vec.getNextEntry();
14 }
15 table.setData(tableData);
```

Gleiche Logik, Notes-Style

```
var table = Ti.UI.createTableView();
var tableData = [];
var db = Ti.Database.open("contacts");
var dbRows = db.execute("SELECT * FROM view_people"
while (dbRows && dbRows.isValidRow()) {

    var tableRow = {
        id: dbRows.fieldByName("id"),
        title : dbRows.fieldByName("name")+", "+dbR
    };
    tableData.push(tableRow);

    dbRows.next();
}
dbRows.close();
db.close();
table.setData(tableData);
```

```
var table = Ti.UI.createTableView();
var tableData = [];
var notesdb = new NotesDatabase("contacts");
var notesView = notesdb.getView("people");
var vec = notesView.allEntries();
var ve = vec.getFirstEntry();
while (ve) {
    var tableRow = {
        id: ve.universalID,
        title : ve.getColumnValue("name")+", "+ve.getColu
    };
    tableData.push(tableRow);
    ve = vec.getNextEntry();
}
table.setData(tableData);
```

„Titanium wie LotusScript“

+ erleichtert dem typischen Notes Entwickler die Eingewöhnung
+ ermöglicht kompakteren, übersichtlicheren Code
+ spart Aufwand bei wiederkehrenden Aufgaben
+ spart dem Entwickler, sich täglich mit SQL, XML Interpretation etc. auseinanderzusetzen
+ schafft einen Standard für die Arbeit mit Notes Daten in einer App

„Titanium wie LotusScript“

+ erleichtert dem typischen Notes Entwickler die Eingewöhnung
+ ermöglicht kompakteren, übersichtlicheren Code
+ spart Aufwand bei wiederkehrenden Aufgaben
+ spart dem Entwickler, sich täglich mit SQL, XML Interpretation etc. auseinanderzusetzen
+ schafft einen Standard für die Arbeit mit Notes Daten in einer App

- erfordert einmalig zusätzlichen Aufwand
- nimmt den Zwang, sich „täglich“ mit SQL, XML Interpretation etc. auseinanderzusetzen

Einige Tipps zur Performance

- sicherstellen, dass HTTP Traffic von Domino komprimiert übertragen wird
- Caching kann einfach mit applicationScope implementiert werden
- In Ansichten nur NotesViewEntry Objekte nutzen, niemals auf das Backend-Dokument zugreifen
- NotesViewNavigator nutzen (nicht NotesView.getAllEntries())
- NotesViewNavigator.setBufferMaxEntries() nutzen
- NotesView.setAutoUpdate(false)
- In Schleifen nur Methodenaufrufe, die wirklich notwendig sind, z.B. Länge eines Arrays vorher in einer Variablen ablegen
- Java statt SSJS nutzen
- String Verkettung mittels java.util.StringBuffer

Einige Tipps zur Performance

Beispiel aus der Praxis:

JSON Daten aus 75.000
Ansichts-Einträgen erzeugen
vorher: 4 Minuten

Einige Tipps zur Performance

Beispiel aus der Praxis:

JSON Daten aus 75.000
Ansichts-Einträgen erzeugen
vorher: 4 Minuten

nachher: 18 Sekunden

Details unter http://julianbuss.com/blog/1600-faster-building-json-data-out-of-a-view-with-75.000-documents-in-26-seconds-instead-of-4-minutes-htm/

Fazit

- mit einem Framework wie Titanium wird die Entwicklung nativer Apps wesentlich erleichtert
- Synchronisation, Speicherung und Verarbeitung von Notes-Daten in der App ist gut machbar
- Verpacken von Logik in eigenen, LotusScript nachempfundenen Klassen vereinfacht das Leben
- Security ist gegeben
- native Apps haben Vorteile bzgl. offline, Look & Feel

Danke!
Fragen?

Mein Framework für Sync & Verarbeitung von Notes-Daten in Titanium:
http://youatnotes.com/dominotogo

Titanium gibt es kostenlos unter:
http://appcelerator.com

Blog-Einträge zum Thema:
http://julianbuss.com/blog/category/dominotogo/

Julian Buß
http://julianbuss.net

jbuss@julianbuss.net