



# Autohändler

Diese Präsentation wird  
Ihnen präsentiert von:  
Alexander Hahn und  
Marcus Bauer

---

# Inhalt

- Ziele
- Entwicklung
  - Quellcode? Datenbank?
  - verwendete Technologien
- Oberfläche
- Konfiguration
- Benutzerverwaltung
- Bedienungsanleitung
- Reflexion

# Ziele

- Gesamten Lagerbestand an Neu- und Gebrauchtwagen verwalten
  - Baujahr, PS, Modell, Hersteller, Farbe, Typ, ...
- Erfassen von Kunden und deren Bestellungen
- Einfache Anwendung, welche das alles ermöglicht

# Entwicklung

- Java 7
- Swing
- Microsoft SQL Server 2008 R2
- jTDS
  - JDBC-Treiber
  - Open Source (LGPL), Komplette in Java (Ausnahme: Windows-Authentifizierung)
  - <http://jtds.sourceforge.net/>

# Entwicklung?

- Entwicklung mit zwei Personen
  - Nur lokale Installation und Entwicklung auf einem Rechner nicht praktikabel
- Quelltext auf github
  - Repository-Hosting für Git
- Realisierung der Datenbank: ?

# Entwicklung: Datenbank Migrationen

- Veränderung der zugrunde liegenden Datenbankstruktur
  - CREATE, ALTER, DROP
- Änderung an Inhalten?
  - Nur, wenn Einfügen bzw. Ändern für das Programm notwendig ist
  - Möglichst wenig Einfügen neuer Datensätze, nur vordefinierte Werte
  - Beispiele
    - Füllen der Tabelle „Hersteller“
    - Mehrwertsteuersatz - 0.19 → 19

# Entwicklung: Datenbank Migrationen

- Gleichzeitige Änderungen
  - Jederzeit möglich, System erkennt „fehlende“
  - Besser als Austausch von SQL-Dateien per Hand oder E-Mails „Ändere mal diese Tabelle“
  - Probleme nur, wenn die verwendeten Objekte nicht mehr in dieser Form existieren (gelöschte Tabelle, umbenannte Felder, ...)
- Konzept an *Ruby on Rails* angelehnt
  - Dateiname ist `201201311200_was_passiert.rb` enthält Datum und Uhrzeit (31. 1. 2012, 12:00)
  - In Rails: normaler Quelltext
    - kann Migrationen beliebig anwenden und rückgängig machen, Stand wird gespeichert

# Entwicklung: Datenbank Migrationen in unserem Projekt

- Selbstgeschriebene Java-Klasse MigrateTool.java
- Prinzip
  - Lädt alle ids aus der Tabelle *migrations*
  - Liest Dateien im Verzeichnis *migrations* nach Name sortiert
  - Durchläuft die Dateiliste
    - wenn Migration nicht in Tabelle, als Transaktion ausführen
    - Bei Fehler: Abbruch, Fehlermeldung
- \*.sql-Dateien
  - Keine „Rückgängig“-Aktion wie in Ruby on Rails
  - Keine Notwendigkeit im Projekt



# Entwicklung: Datenbank Klassen

- Database
  - connectWindowsAuth, connectSQLAuth
  - disconnect
  - prepare(String) → Prepared
  - transaction(Prepared) → boolean
- Result
  - HashMap
  - getInt(feldname) → Integer,  
getString(feldname) → String,  
...

# Entwicklung: Datenbank Klassen

- Results
  - extends Vector<Result>
    - einfach iterierbar
  - Implements TableModel
    - Kann als Modell für JTable genutzt werden
    - Logik, um Datenbank zu aktualisieren
      - (eigentliche Abfragen über die Klassen GenericTable bzw. UserTable realisiert)

# Entwicklung: Datenbank Klassen

- ◉ Prepared
  - ◉ executeWithResult(...)
  - ◉ executeNoResult(...)

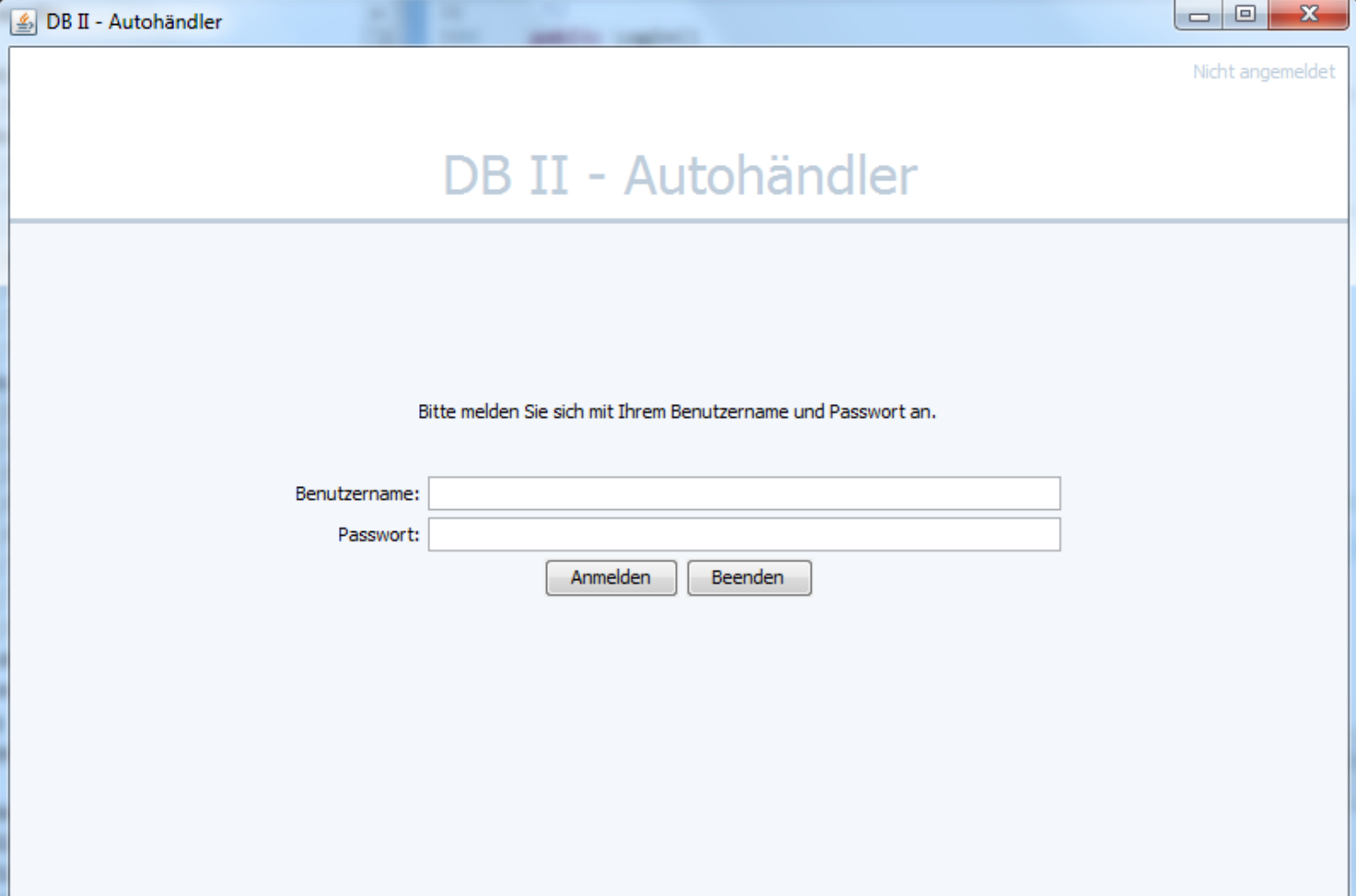
```
Prepared p = db.prepare („SELECT * FROM autos  
    WHERE hersteller_id = ? AND baujahr > ?");  
Results r = p.executeWithResult(herstellerID, 2010);
```

```
for(Result result : results)  
    System.out.println(  
        result.getString („bezeichnung"));  
p.close();
```

# Oberfläche

- mit Swing erstellt
- Grundlegende Idee:
  - Stack mit allen geöffneten Unterfenstern
  - beliebig hoher Verschachtelungsgrad von Fenstern möglich
  - Rückkehr aus aktuellem Fenster mit `goToHome()` oder `goUp()`
  - z.B. Kunden → alle Bestellungen des Kunden → alle Autos in einer Bestellung
- einheitliches Look- and Feel
- geringe Anzahl konfigurierbarer Farben

# Oberfläche: Anmeldung



The screenshot shows a web application window titled "DB II - Autohändler". In the top right corner of the window, it says "Nicht angemeldet". The main heading in the center is "DB II - Autohändler". Below this, a message reads: "Bitte melden Sie sich mit Ihrem Benutzernamen und Passwort an." There are two input fields: "Benutzername:" and "Passwort:". Below the input fields are two buttons: "Anmelden" and "Beenden".

DB II - Autohändler

Nicht angemeldet

DB II - Autohändler

Bitte melden Sie sich mit Ihrem Benutzernamen und Passwort an.

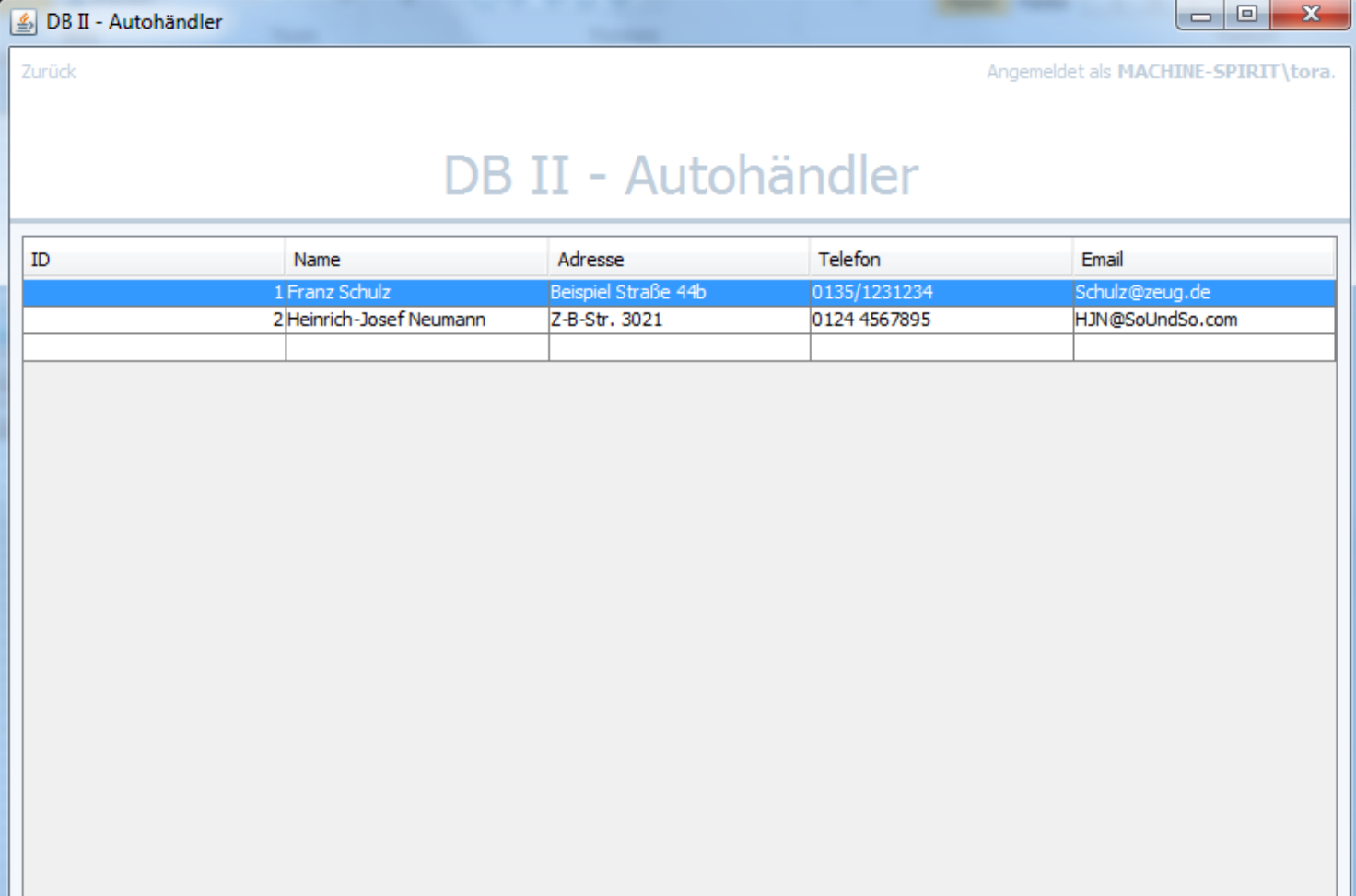
Benutzername:

Passwort:

# Oberfläche: Dashboard



# Oberfläche: Anzeige einer Tabelle



DB II - Autohändler

Zurück

Angemeldet als MACHINE-SPIRIT\tora.

## DB II - Autohändler

ID	Name	Adresse	Telefon	Email
1	Franz Schulz	Beispiel Straße 44b	0135/1231234	Schulz@zeug.de
2	Heinrich-Josef Neumann	Z-B-Str. 3021	0124 4567895	HJN@SoUndSo.com

# Konfiguration

- config/app.xml ⇔ Config.java mit get...()
- Wichtigste Eigenschaften konfigurierbar
  - Name der Anwendung
    - In der Praxis: Name des Autohändlers
  - Serveradresse, Datenbankname, unterstützte Anmeldemethoden (Windows/SQL)
  - Farben



# Benutzerverwaltung

- Jeder Benutzer der Software ist ein Datenbankbenutzer
- Benutzergruppen
  - lesen – nur Lesen
  - schreiben – Lesen und Schreiben
  - db\_owner – Lesen, Schreiben, Löschen, Datenbankstruktur verändern
- Anlegen neuer Benutzer – Serverrecht, keine von uns neu definierte Rolle
  - db\_owner erhält automatisch sysadmin

# Benutzerverwaltung: Rechtevergabe

- CREATE LOGIN usw. unterstützt keine ? als Parameter
  - Namen und Passwörter auf a-zA-Z0-9 beschränkt
- Hinzufügen neuer Rechte (ähnlich löschen)
  - EXEC sp\_addrolemember role, user
  - EXEC sp\_addsrvrolemember user, role
  - Also: fast konsistent implementierbar
    - ALTER (SERVER) ROLE ... ADD MEMBER ... ist erst ab SQL Server 2012 verfügbar

# Benutzerverwaltung: Authentifizierung

- Windows
  - aktueller Benutzer wird angemeldet
  - kein Passwort erforderlich
  - Login-Bildschirm nur bei Fehlern
- SQL-Server
  - Login-Bildschirm
- Konfigurierbar
  - Eine oder beide Optionen aktivieren

# Bedienungsanleitung

- Bedienung ist intuitiv
  - Einfüge: Füllen der leeren Zeile am Ende
  - Bearbeiten: Auf einen Wert klicken, neuen Wert einfügen
  - Löschen: Zeilen markieren, „Ausgewählte Löschen“

# Reflexion:

## Benötigte Zeit

- 2 Tage: funktionierende Datenbank-anbindung, Migrationen, Aufbau der Fenster, Anmelden
  - ≈ 1.100 Zeilen Java (+ Kommentare, Leerzeilen, ...)
- 3 Tage: Anzeigen, Einfügen, Löschen und Verändern von Daten
- 2 Tage: Rechte-/Benutzerverwaltung
- 1 Tag: kleinere Tweaks, Verwalten von Bestellungen
  - ≈ 2.200 Zeilen Java insgesamt

# Reflexion:

## Wiederverwendbarkeit

- Datenbankklassen können komplett wiederverwendet werden
  - Nur ggf. andere JDBC-Treiber
- Migrationen
  - offensichtlich: \*.sql-Dateien für andere Projekte müssen erstellt werden
  - ohne Codeänderung wiederverwendbar
  - enorme Zeitersparnis selbst bei zwei Entwicklern in Hinsicht auf konsistente Datenbanken
  - Selbst bei einer Datenbank: Struktur in Versionsverwaltung



# Ende Autohändler

Diese Präsentation wurde  
Ihnen präsentiert von:  
Alexander Hahn und  
Marcus Bauer

---