12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Grails–in–a–Day

Grails Workshop

Detlef Brendle, Christoph Sperle

Canoo Engineering AG

Basel, Schweiz

Herzlich Willkommen…

- **Detlef Brendle**
  - Software Engineer @ Canoo AG, Basel
  - Grails Erfahrung seit 2008

  > Risikomodellierungs–Applikation

  > CustomerSelfCare Applikation

  > Prototyp für Offertenstellung

- **Christoph Sperle**
  - Software Engineer @ Canoo AG, Basel
  - Grails Erfahrung seit 2008

  > CATS: Canoo Tippspiel zur WM 2010

  > Dozent für Webentwicklung an der Dualen Hochschule Lörrach

Workshop–Überblick

- Session 0 (ca. 60 min) – **Einführung in Grails**
- Pause / offene Diskussion (30 min)

- Session 1 (ca. 90 min) – **Hands on Grails ! *The basics***
- Mittagspause (12.30 – 14.00 Uhr)

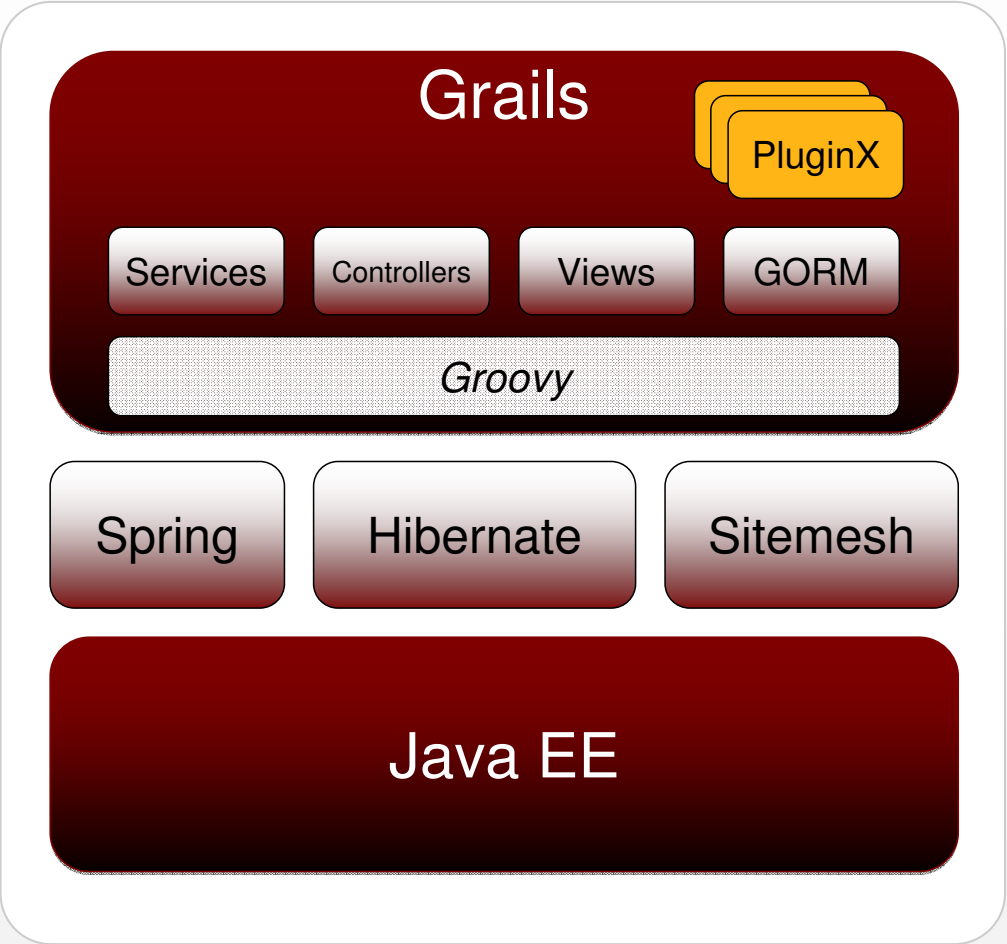
- Session 2 (ca. 60 min) – **Hands on Grails ! *Advanced techniques***
- Pause / offene Diskussion (15 min)

- Session 3 (ca. 60 min) – **Hands on Grails ! *Expert corner***
- Pause / offene Diskussion (15 min)

- Session 4 (ca. 80 min) – **Hands on Grails ! *'Spring Security' Plugin***
- Offene Diskussion & Ende (18.00 Uhr)

Grails – Ein Webapplikations–Framework

- Groovy basiertes Web–Framework
  - 100% Java–Integration
- Convention over configuration
- MVC – Ansatz zur klaren Trennung
- Build System
  - Kommandozeile
  - dependency management
- IDE Support
- `'grails console'`



Grails – Setup und Projektstruktur

- `grails create-app <PROJEKT_NAME>`

- Standard–Projektstruktur wird angelegt:
  - <PROJEKT_NAME>
    - grails–app
      - conf
      - controller
      - domain
      - i18n
      - services
      - taglib
      - views
    - lib (Zusätzliche Java Libraries)
    - src (produktiver Code)
    - test (unit + integration)
    - web–app (Web resourcen)

Grails – Kommandozeilenoperationen

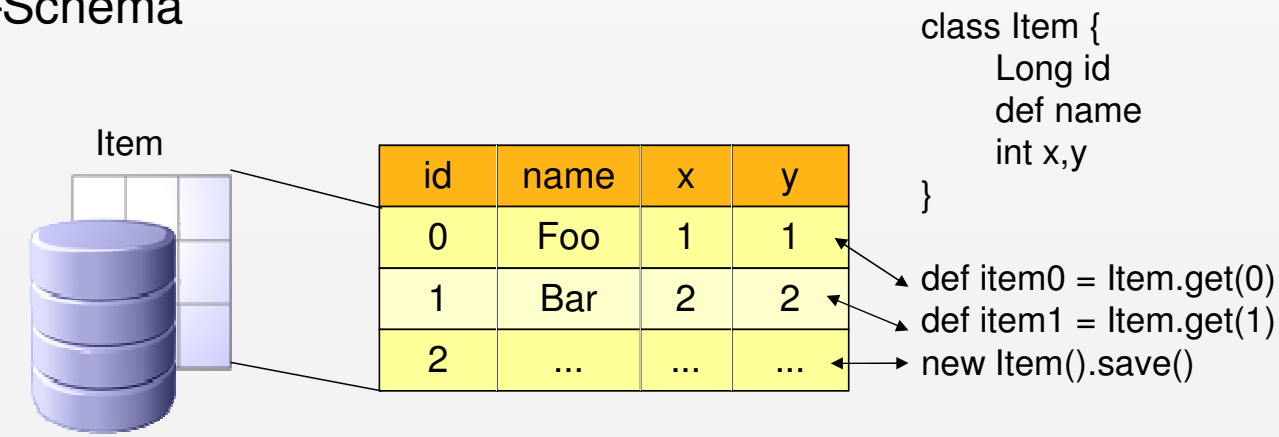
- Operationen auf Kommandozeilenebene
  - **`grails`**
  - > **`createApp`** (erzeugt ein Grails Projekt)
  - > **`create-*`** (Erstellen von Grails Artefakten wie Controller, Domänenklassen, Service)
  - > **`runApp`** (startet die Webapplikation)
  - > **`testApp`** (führt alle Testcases des Projektes aus und erstellt Report)
  - > **`war`** (erzeugt eine WAR Datei)
  - > …
  - > **`help`** (listet alle möglichen Operationen auf)

Hinweis: Plugins können weitere Operationen mitliefern

> z.B. **`run-webtest`**

Grails – Domain Objects

- Domänenobjekte halten Status und implementieren Verhalten
- Domänenobjekte können über Relationen verbunden werden (z.b. one–to–many, many–to–many,...)
- GORM – Grails Object Relational Mapping – basiert auf Hibernate 3 ORM
- Dynamische Methoden werden 'injiziert'
  - **`findBy.., save(), delete(), ...`**
- Implizites DB–Schema



Grails – Constraints

- Attribute können durch constraints gesichert werden
- Beispiele: '**nullable**', '**blank**','**size**',...
- Ein Domänenobjekt kann nur gespeichert werden wenn alle constraints erfüllt sind.
- Die dynamische Methode '**validate**' prüft constraints
- Fehler werden am Domänenobjekt abgelegt (dynamische Property '**errors**')
- Custom validation

```
class Item {

    String description
    Date startDate
...
    static constraints = {

        description(blank: false)

        startDate(nullable: false, validator: {newValue, item ->
           // validate here
        })
..    }
}
```

Grails – Datenquellen und Bootstrapping

- Einstellungen in DataSource.groovy
- Umgebung (Environment) wird bei Grails Kommando gesetzt, kann aber übersteuert werden : '**`grails -Dgrails.env=<ENV>`**'
- Klasse Bootstrap definiert Closure
  - **`init`** (startup)
  - **`destroy`** (shutdown)

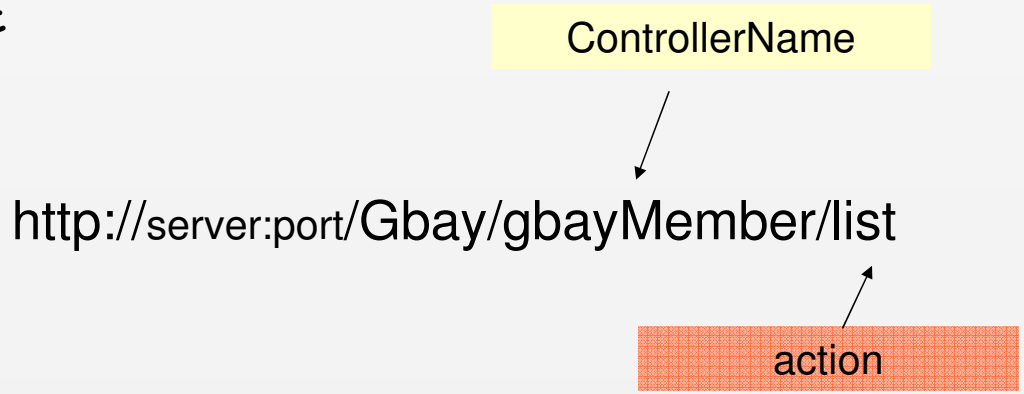
```
class BootStrap {

     def init = { servletContext ->
     }
     def destroy = {
     }
}
```

```
dataSource {
    pooled = true
    driverClassName = "org.hsqldb.jdbcDriver"
    username = "sa"
    password = ""
}
...
// environment specific settings
environments {
    development {
        dataSource {
            dbCreate = "create-drop"
            url = "jdbc:hsqldb:mem:devDB"
        }
    }
    test {
        dataSource {
            dbCreate = "update"
            url = "jdbc:hsqldb:mem:testDb"
        }
    }
    production {
        dataSource {
            dbCreate = "update"
            url = "jdbc:hsqldb:file:prodDb;shutdown=true"
        }
    }
}
```

Grails – Controller

- Controller definiert Actions (Closures) für die request–Verarbeitung
- Mapping per Namenskonvention
- Controller erstellt/modifiziert Model für View
- Requestparameter stehen als Map bereit
- Ebenso Zugriff auf
  - **`javax.servlet.ServletContext`**
  - **`javax.servlet.Session`**
  - **`javax.servlet.Request`**

```
class GbayMemberController {

        def authenticateService


        def index = {
                redirect action: list, params: params
        }

        def list = {
                [personList: GbayMember.list(params)]
        }

        def save = {
                render view: 'create', model: [...]
        }

...
}
```



Grails – Views

- View Technologie: GroovyServerPages (GSP)
- Built–in Tags stehen zur Verfügung (z.b. **<g:each**, **<g:if**,**<g:link**,...)
- Taglibs und Templates für Modularisierung und Wiederverwendung
- Layoutunterstützung mit Sitemesh
- AJAX Support
- I18n Support über Taglib 'message' **<g:message code='messagecode'>**

Grails – Testing

- '**`grails testApp`**' führt alle Tests aus und erstellt Report
- UnitTests und IntegrationTests
- UnitTests benötigen keine Grails Umgebung → schnelle Ausführungszeit
- **`grails.test.GrailsUnitTestCase`** bietet diversen Testsupport
  - mock–Methoden z.B. '**`mockForConstraintsTest`**', '**`mockDomain`**', ...
  - Testen der dynamischen Methoden
- IntegrationTests laufen innerhalb des Grails–Kontextes
- Webtesting, CodeCoverage als PlugIn

Grails – PlugIns

- PlugIn Mechanismus bietet Erweiterbarkeit nach Wahl
- Modularisierung von Grails-Applikationen
- PlugIns sind ebenfalls als Grails Projekte aufgebaut
- PlugIns können andere PlugIns referenzieren
- Installation: '`grails install-plugin <[Name|Pfad|URL]>`'
- Vielzahl von Plugins verfügbar unter: http://grails.org/plugin/home

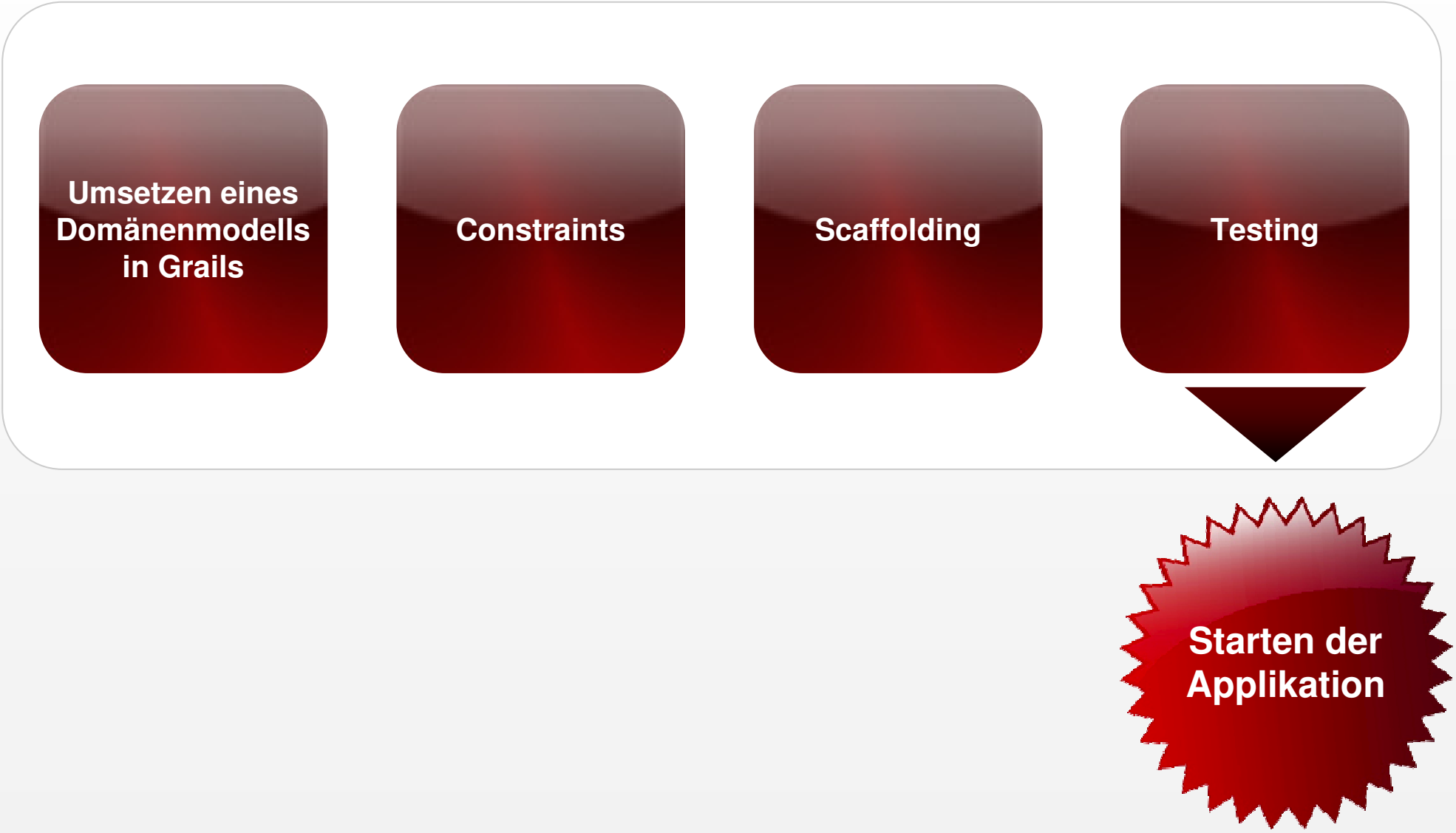
canoo

Fragen?

Hands on Grails

Hands on Grails – The basics

Hands on Grails – The basics



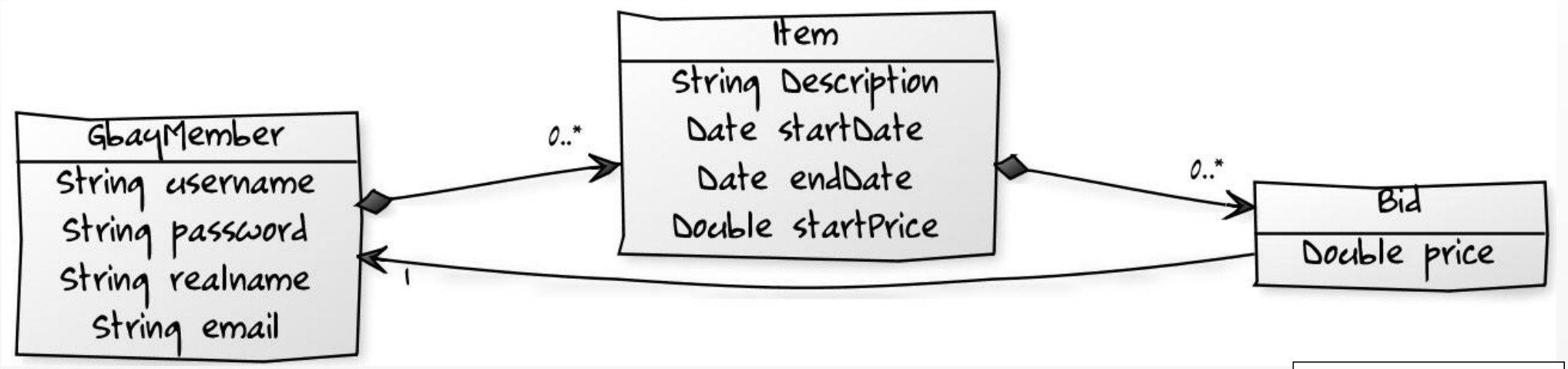
The basics – Installation prüfen

- Voraussetzungen :
  - Java 1.5+ installiert
  - JAVA_HOME Umgebungsvariable gesetzt
- GRAILS_HOME auf <USB-Stick>/grails setzen
- PATH Variable mit GRAILS_HOME/bin erweitern
- Verifikation:
  - '**`grails help`**' Kommando ausführen.

```
detlef@detlef-laptop:~$ grails help
Welcome to Grails 1.1.1 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: /programs/grails-1.1.1

Base Directory: /home/detlef
Running script /programs/grails-1.1.1/scripts/Help_.groovy
Environment set to development

Usage (optionals marked with *):
grails [environment]* [target] [arguments]*
```

Hands on Grails – The basics

- UML Diagramm:



username unique
password more than 4 characters
realname not empty
email valid email format

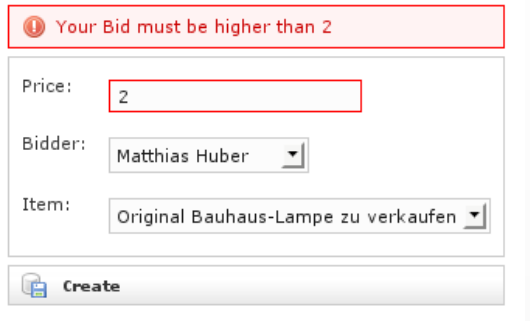
description: not empty
startPrice >= 1

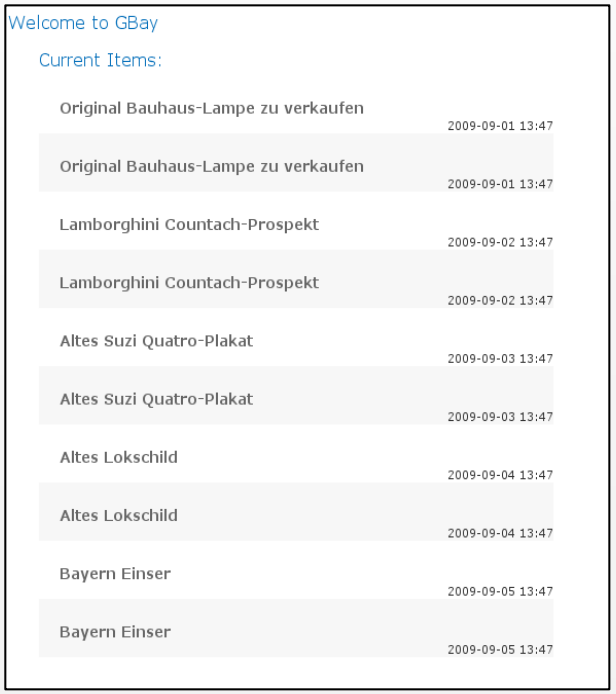
price >=1

Hands on Grails
Advanced techniques

Hands on Grails – Advanced techniques

- Custom validation
  - Item: startDate > now ; startDate < endDate
  - Bid: price >= startPrice ; price > highestBid
- Views & Templates
  - Neue Einstiegsseite mit Liste aller Items
- Layout



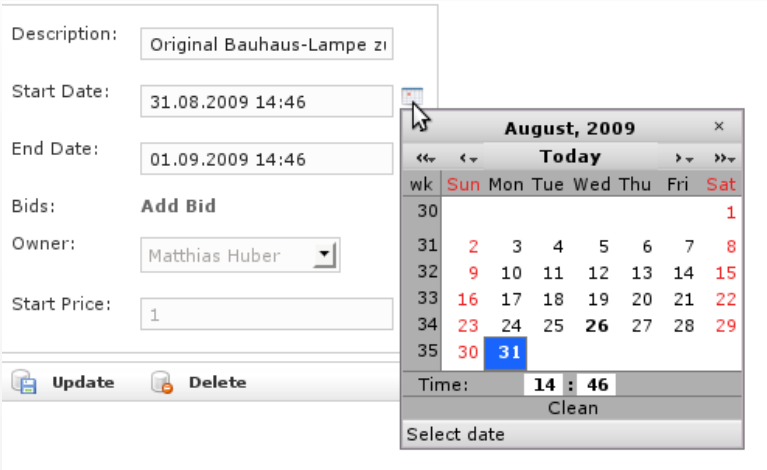


Hands on Grails
Expert Corner

Hands on Grails – Expert Corner

- PlugIns
  - Demo: code-coverage PlugIn
  - Einsatz des calendar PlugIn

  > DatePicker für Datumseingabe



Hands on Grails – Spring Security Plugin

Hands on Grails – Spring Security Plugin

Spring Security Features

- Advanced Authentication/Authorization für J2EE
- Support für Java Authentication (JAAS)
- OpenID Support
- CAS Support (Single-sign-on)
- LDAP Authentication

Hands on Grails – Spring Security Plugin

Spring Security and Grails

- Settings in Config.groovy
- **@Secured**(['<ROLE']) Annotation Support
- Spring Bean 'springSecurityService,
- Taglib <sec:{feature}>

```
<sec:ifLoggedIn>
   Welcome Back<sec:username/>!
</sec:ifLoggedIn>
```

canoo

Vielen Dank!

canoo