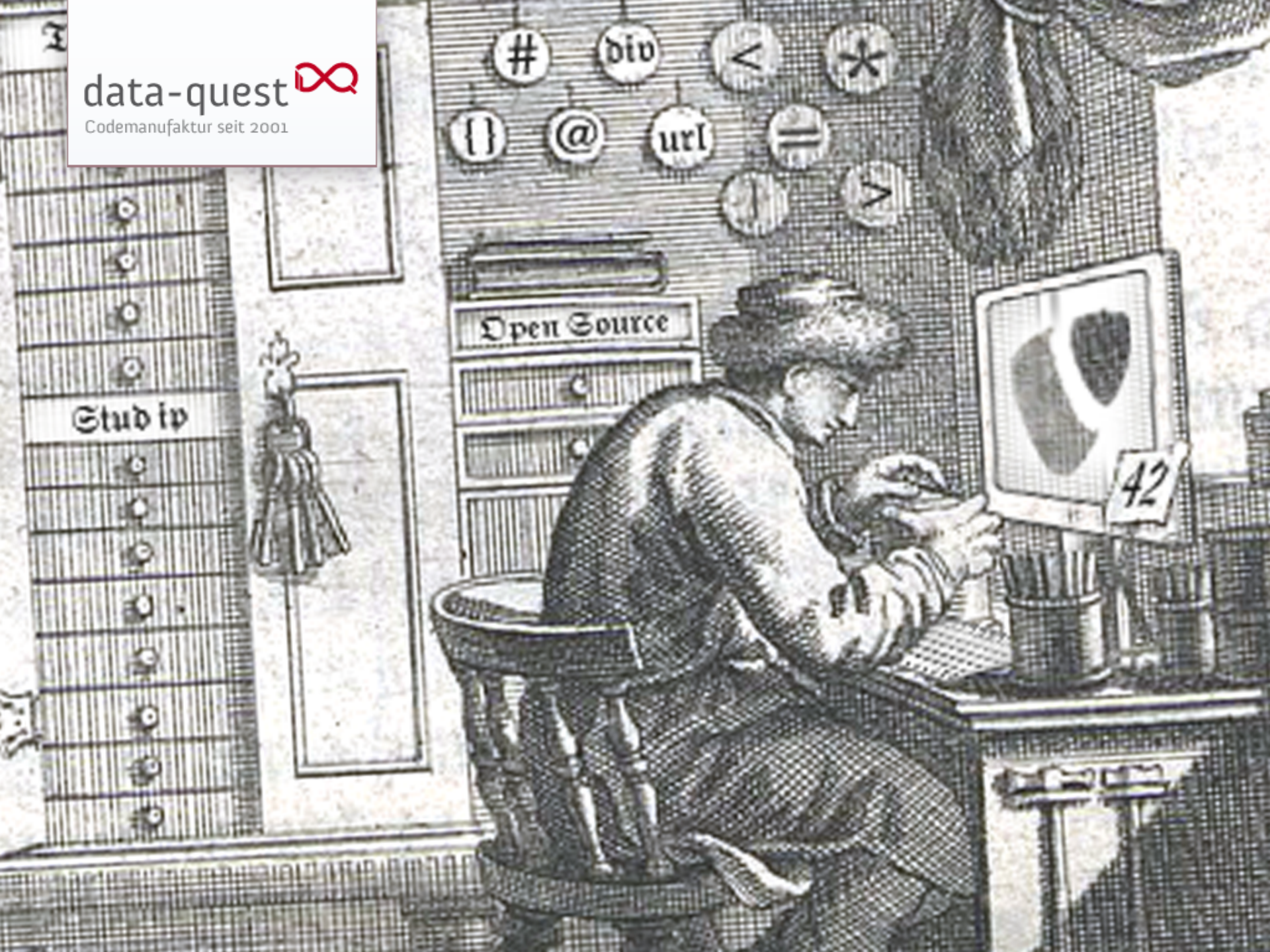
data-quest
Codemanufaktur seit 2001
#
div
<
*
{}
@
url
=
|
>
Open Source
Stud ip
42

data-quest
Codemanufaktur seit 2001

Stud.IP - ActiveRecord

aka SimpleORMap

Entwicklertagung 2013 Wiesbaden, Freitag 22.03.2013

André Noack, data-quest GmbH

data-quest
Codemanufaktur seit 2001

Gliederung

- Active Record? ORM?
- SimpleORMap Features
- Einfache Nutzung, CRUD
- Nutzung mit Relationen
- neue Klassen erstellen
- Probleme, offene Enden

data-quest
Codemanufaktur seit 2001

Active Record? ORM?

"An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data."

Martin Fowler, P of EAA page **160**

ORM = object-relational mapping

Eine Technik der Softwareentwicklung, mit der ein in einer objektorientierten (!) Programmiersprache geschriebenes Anwendungsprogramm seine Objekte in einer relationalen Datenbank ablegen kann.

data-quest
Codemanufaktur seit 2001

SimpleORMap

- inspiriert durch RoR ActiveRecord
- in Stud.IP 1.4 in einer ersten Variante (PHP4) aufgetaucht
- ab 2.0 mit ArrayAccess und statischen find Methoden (PHP5)
- ab 2.4 mit Abbildung von Relationen, magischen statischen find methoden und umfangreichen Umbauten
- Grundgedanke "simple", möglichst nur eine Klasse, nichts konfigurieren müssen, was auch anders in Erfahrung zu bringen ist
- im Zweifel geringere Funktionalität
- Stud.IP Spezialitäten (z.b. md5 Schlüssel, chdate) sollten schon dabei sein

```php
class seminare extends SimpleORMap {}
$s = new Seminare($id);
echo $s->getValue('name');
//ab 2.0
echo $s->name;
echo $s['name'];
```

data-quest
Codemanufaktur seit 2001

SimpleORMap II

- alle Spalten der Tabelle werden zu Eigenschaften des Objektes
- Groß-Kleinschreibung ist egal
- Metadaten der Tabelle werden ausgelesen und gecached
- ArrayAccess, kann (fast) wie ein Array angesprochen werden
- Alias-Spalten (z.B. id als Alias für den Primärschlüssel)
- virtuelle (berechnete Spalten)
- 1:n und n:m Relationen zu anderen SimpleORMap Objekten
- callbacks für den Lebenszyklus eines Objektes
- automatische Schlüsselerzeugung Stud.IP Style / Autowert
- Standardwerte für Spalten, chdate,mkdate Automatik
- find(), findMany(), findBySQL(),exists(),create(),import(),countBySQL()
- magische finder für findBy<Spaltenname>
- Collection Klasse für Relationen, SimpleORMapCollection
  - Methoden zum Iterieren, Daten extrahieren
  - gelöschte Objekte vorhalten

data-quest
Codemanufaktur seit 2001

Einfache Nutzung C(REATE)

- neues Objekt instanzieren
- Eigenschaften setzen, mehrere als Array an setData()
- store() speichert Änderungen, erzeugt bei Bedarf neue Datenzeile
- Bei einwertigem Schlüssel: 32 Zeichen Schlüssel oder autowert
- ::create() gibt neues Objekt zurück, Daten werden bereits gespeichert

```php
$course = new Course();
$course->name = 'Neue Veranstaltung';
$course->store();

//alternativ
$course->setData(array('name' => 'Neue Veranstaltung'));
$course->store();

//oder
$course = Course::create(array('name' => 'Neue Veranstaltung'));
```

data-quest
Codemanufaktur seit 2001

Einfache Nutzung R(EAD)

- Objekte finden mit find Methoden
- neues Objekt mit PK als Parameter sucht auch passende Zeile
- toArray() liefert alle Eigenschaften als Array, auch rekursiv

```
$id = 1;
$course = new Course($id);
if (!$course->isNew()) {
    echo $course->name;
} else {
    //durch Aufruf über Konstruktor wird id gesetzt
    //$course->id ist 1
}

$course = Course::find($id);
if (!is_null($course)) {
    echo $course->name;
}

if (!Course::exists($id)) {
    $course = new Course($id);
    $course->store
}
```

data-quest
Codemanufaktur seit 2001

Einfache Nutzung R(EAD)

- viele Objekte finden mit findBy Methoden
- ::findBySQL() nimmt Klausel mit Platzhaltern und Parameter entgegen
- ::findMany() nimmt Array von Schlüsseln entgegen
- ::findBy<Spalte> ist Abkürzung für ::findBySQL('<Spalte>=?')
- es wird immer ein Array zurückgegeben
- ::findEach... nimmt als ersten Parameter ein callback

```
$courses = Course::findBySql('start_time = ? ORDER BY Name', array($semester_start));

$courses = Course::findBystart_time($semester_start, 'ORDER BY Name');

$courses = Course::findMany($course_ids);

$course_names = Course::findEachMany(function($s) {return array($s->name, $s->nummer);},
$course_ids);
```

data-quest
Codemanufaktur seit 2001

Einfache Nutzung U(PDATE)

- Eigenschaften ändern, ::store() aufrufen
- speichert nur, wenn Änderungen vorgenommen wurden (Problem!?)
- veränderte abhängige Objekte werden auch gespeichert
- ::setData() zum Ändern von mehreren Eigenschaften
- ::import() um ein Array in eine Objektstruktur zu überführen

```
$course = Course::find($id);
$course->name = 'Testveranstaltung 2';
$changed = $course->store();

$data = array('name' => 'Testveranstaltung 2', 'nummer' => '12345');
$course->setData($data);

$data['members'][] = array('status' => 'dozent', 'id' => 5);

$course = Course::import($data);
$course->store();
//speichert neuen Kurs mit einem zugewiesenen Dozenten
```

data-quest
Codemanufaktur seit 2001

Einfache Nutzung D(ELETE)

- ::delete() Methode
- löscht Datenzeile, und wenn konfiguriert abhängige Daten
- das Objekt bleibt zwangsläufig erstmal erhalten, wird aber geleert
- ::isDeleted() gibt Auskunft

```
$course = Course::find($id);
$course->delete();
if ($course->isDeleted()) {
  $course->setNew();

}
```

data-quest
Codemanufaktur seit 2001

Nutzung mit Relationen

- Relation zugreifbar über getValue(), also als virtuelle Eigenschaft
- ist immer eine Instanz von SimpleORMapCollection
- SimpleORMapCollection erbt von ArrayObject
- stellt diverse Methoden für die Collection bereit
- findBy(), each(), map(),filter(), first(), val(), unsetBy(). toGroupedArray()

```
$course = Course::find($id);
$course->members[0]->user_id;
$course->members->val('user_id');
//findBy() liefert Collection
$course->members->findBy('status', 'dozent')->pluck('user_id');
$course->members->findBy('status', 'dozent')->toGroupedArray('user_id', 'username vorname
nachname');
unset($course->members[0]); //geht zwar
$course->member->unsetBy('username', 'noack'); //sinnvoller
$course->member->getDeleted();
$course->member->refresh();
$courses_collection = SimpleOrMapCollection::createFromArray(Course::findbySQL("name LIKE ?",
array('Test%')));
$courses_collection->each(function($c) {$c->visible = 0; $c->store();});
$courses = $courses_collection->pluck('name');
```

data-quest
Codemanufaktur seit 2001

Neue Klassen erstellen

- Optionen müssen im Konstruktor gesetzt werden
- Ohne Optionen, wird der Name der Klasse als Tabellenname benutzt
- $this->db_table = Name der Tabelle in der Datenbank
- Spalten/PK und default Werte werden ausgelesen
- eigene default Werte über $this->default_values[<spalte>] = ''
- Alias Spalten über $this->alias_fields[<spalte>] = <andere spalte>
- zusätzliche berechnete über $this->additional_fields[<spalte>][get|set]

```
class Course extends SimpleORMap
{
   function __construct($id = null)
   {
      $this->db_table = 'seminare';
      $this->default_values['admission_endtime'] = -1;
      $this->alias_fields['number'] = 'veranstaltungsnummer';
      $this->additional_fields['end_time']['get'] = function($course) {
         return $course->duration_time == -1 ? -1 : $course->start_time + $course-
>duration_time;
      };
      parent::__construct($id);
```

data-quest
Codemanufaktur seit 2001

Neue Klassen erstellen

- Typen von Relationen ('type' =>)
  - has_one: 1:1
  - has_many: 1:n
  - belongs_to: n:1/umgedrehtes 1:1
  - has_and_belongs_to_many: n:m
- class_name => Name der anderen SimpleORMap Klasse
- foreign_key => Name des Fremdschlüssels in der eigenen Klasse
- assoc_foreign_key => Name des zug. Schlüssels in der anderen Klasse, default PK
- assoc_func => alt. Angabe der stat. Methode in der anderen Klasse
- assoc_func_params_func => alt. Closure für Parameter von assoc_func
- thru_table => Verknüpfungstabelle für n:m
- thru_key => Name der linken Verknüpfungsspalte
- thru_assoc_key => Name der rechten Verknüpfungspalte
- on_delete => delete bei Löschweitergabe, alt. Closure
- on_store => store bei Aktualisierungsweitergabe, alt. Closure

data-quest
Codemanufaktur seit 2001

Neue Klassen erstellen

```
$this->has_many = array(
        'members' => array(
            'class_name' => 'CourseMember',
            'on_delete' => 'delete', 'on_store' => 'store'),
    );
    $this->belongs_to = array(
        'start_semester' => array(
            'class_name' => 'Semester',
            'foreign_key' => 'start_time',
            'assoc_func' => 'findByTimestamp',
            'assoc_foreign_key' => 'beginn'),
        'home_institut' => array(
            'class_name' => 'Institute',
            'foreign_key' => 'institut_id',
            'assoc_func' => 'find')
    );
    $this->has_and_belongs_to_many = array(
        'study_areas' => array(
            'class_name' => 'StudipStudyArea',
            'thru_table' => 'seminar_sem_tree',
            'on_delete' => 'delete', 'on_store' => 'store'),
);
```

data-quest
Codemanufaktur seit 2001

Neue Klassen erstellen

- callbacks: before_create before_update before_store before_delete after_create after_update after_store after_delete after_initialize
- callback ist Closure oder Instanz Methode, bekomt den Typ übergeben
- mehrere Callbacks eines Typs können registriet werden
- Rückgabewert false beendet die Ausführung von weiteren Callbacks
- Bei before_ callbacks wird bei false die Methode beendet

```
if ($this->hasAutoIncrementColumn()) {
    $this->registerCallback('before_store after_create', 'cbAutoIncrementColumn');
} elseif (count($this->pk) === 1) {
    $this->registerCallback('before_store', 'cbAutoKeyCreation');
}
```

data-quest
Codemanufaktur seit 2001

Probleme, offene Enden

- Performance, n+1 Problem
- Kein Identity Mapper
- Enge Bindung an Datenbank, Testbarkeit
- getValue('test') -> $this->test Problem

Eigene Erfahrungen, Anregungen?

data-quest

Codemanufaktur seit 2001

data-quest

Codemanufaktur seit 2001

data-quest
Codemanufaktur seit 2001

Text 2-Spaltig

data-quest
Codemanufaktur seit 2001

Foto vertikal