



Web-basierte Anwendungen

Praktikumsaufgabe 3: Formulare, HTML5

Studiengänge AI (4140) & WI (4120)

HTML5-Formulare: Vorbereitungen

- Erweitern Sie die Ergebnisse aus Aufgabe 1&2
 - Wechseln Sie in den Basisorder „wba1“ Ihres Rails-Projekts
`$ cd mein/pfad/zu/wba1`
 - Erzeugen Sie nun zwei leere HAML-Seiten für einen Controller „pr03“:
`$ rails generate controller pr03 anmeldung fragebogen`
 - Editieren Sie Datei „./**config/routes.rb**“. Ergänzen Sie die Zeilen
post pr03/anmeldung und post pr03/fragebogen. Beispiel:
`get "pr03/anmeldung" # Bereits vorhanden`
`post "pr03/anmeldung" # Neu hinzufügen`
 - Editieren Sie „./**app/controllers/pr03_controller.rb**“.
Ergänzen Sie in den *actions* „anmeldung“ und „fragebogen“ die Zeile
„raise if request.method == 'POST'“:
`def anmeldung`
 `raise if request.method == 'POST'`
`end`

HTML5-Formulare: Hintergrund

- Hintergrund:
 - Wir haben nun veranlasst, dass auch http-POST-Aktionen möglich sind. In diesem Fall erzeugt der Controller einen Laufzeitfehler und zeigt (im Development-Modus!) Debug-Informationen an.
 - Dieser Fall tritt beim Absenden eines Formulars ein!
 - Die so erzwungene Fehlermeldung zeigt uns die Rückgabewerte der Formulare serverseitig an, d.h. wir können so überprüfen, was unsere Formulare an den Server senden.
- Sinn der Aufgabe
 - *Client*-seitige Eingaben erfolgen meistens über Formulare. Formulare sind daher wichtige Komponenten einer Benutzerschnittstelle im Web
 - Wir wollen den Umgang mit HTML-Formularen üben, d.h. insbesondere mit dem Element „**input**“ und den Varianten seines Attributs „**type**“ vertraut werden sowie mit der Datenübertragung an einen Web-Server.
 - In Vordergrund stehen dabei die neuen HTML5-Möglichkeiten!

HTML5-Formulare: Validierungen

- Validierungen
 - Benutzer-Eingaben sind potenzielle Sicherheitsrisiken (Einschleusung von Schad-Code, z.B. „*SQL injection*“), außerdem sind Anwendungen generell vor Unsinnseingaben zu schützen!
 - Es gibt drei gängige Wege für diese sog. „Validierungen“:
 - *Server*-seitig (unentbehrlich, wird im Kurs aber erst später behandelt)
 - *Client*-seitig mit JavaScript (traditionell)
 - Neu: ***Client*-seitig mit HTML5** (viel einfacher als mit JavaScript)
- Ein Ziel dieser Übung ist die Beschäftigung mit der neuen *Client*-seitigen Validierung mit HTML5.
 - Ermittelt Sie, welche der neuen Möglichkeiten von den installierten Browsern Firefox und Opera unterstützt werden
 - Bauen Sie dazu ein Formular (s.u.), geben Sie Testdaten ein (gültige und nicht gültige), und testen Sie mit beiden Browsern, welche Falscheingaben diese erkennen und was den Server erreicht!



HTML5-Formulare

- **Szenario:** Anmeldung zu einer (nicht-anonymen) Studierendenbefragung
 - Formular 1 simuliert die Eingabe diverser persönlicher Daten sowie die Vergabe eines Passworts Ihrer Wahl im Rahmen einer „Anmeldung“
 - Formular 2 simuliert eine Befragung. Es dient i.w. zum Testen von bisher zu kurz gekommenen Eingabeoptionen.
- **Bemerkung:**
 - In einer realen Anwendung würden Sie erst nach erfolgreichem Login zur Befragungsseite gelangen. Dazu ist u.a. *session management* erforderlich, was aber erst später behandelt wird.
 - Vorerst sind daher beide Seiten „anmeldung“ und „fragebogen“ unabhängig voneinander erreichbar.



HTML5-Formulare

- Technik:

- Es ist zwar möglich, das Formular direkt mit HTML / HAML zu bauen, aber mit Helper-Methoden von Rails ist dies viel leichter! Hier ein View-Fragment:

```
/ In app/views/pr03/anmeldung.html.haml
```

```
%h1 Anmeldung
```

```
= form_tag 'anmeldung' do
```

```
  %table
```

```
    %tr
```

```
      %td Nachname
```

```
      %td= text_field :params, :last_name, size: 20 # , usw.
```

```
    / ... weitere Felder, schließlich:
```

```
    = submit_tag "Absenden"
```

- Fall „date“ gibt es zweimal. Verwenden Sie einmal die HTML5-Lösung mit Rails-Helper **date_field**, einmal den Rails-Helper **date_select**
- Für Fall „hidden“: Ergänzen Sie in Controller „pr03“, Methode „fragebogen“, die neue erste Zeile:

```
params['hidden_info'] = 'Versteckte Angabe' if request.method == 'GET'
```
- Hinweis für Opera 11: Bei nicht ausgefüllten Pflichtfeldern erhält man (leider) keine Warnung – er versendet das Formular einfach nicht...



Vorgaben: „Anmeldung“

M = Muss-Feld, O = Optional,
C = „conditional“ (abhängig von...)

Angezeigter Feldname	Interner Name	M	Input-Typ	Vorgaben
Nachname	last_name	M	text	Breite: 20, Muster: '[\w -]+'
Vorname	given_name	M	text	Breite: 20, Muster: '[\w -]+'
Geschlecht	gender	M	radio	männlich (m), weiblich (f), divers (d)
Geburtsdatum	birthdate	M	date	Format: YYYY-MM-DD Mit Rails-Helper „date_field“,
Familienstand	marital_status	M	radio	Ledig (l), verheiratet (m), geschieden (d), verwitwet (w)
Verheiratet seit	married_since	C	(n.a.)	Mit Rails-Helper „date_select“, Format wie Geb.dat., Jahr ab 2005
E-Mail-Adresse	email	M	email	Breite: 20
Homepage	homepage	O	url	Breite: 20
Telefonnummer	phone	O	tel	Breite: 20, Muster: '((\+\\d+-) 0)\\d+-?\\d*-?\\d+'
Passwort	password	M	password	Breite: 20, mind. 8 Zeichen, dabei keine <i>whitespace</i> -Zeichen
Passwort-Wiederh.	password_confirmation	M	password	Breite: 20

* Vorgaben: „Fragebogen“

Angezeigter Feldname	Interner Name	M	Input-Typ	Vorgaben
(darf <u>nicht</u> zu sehen sein)	hidden_info	M	hidden	(vom Controller aus vorbelegen mit dem Text „Versteckte Angabe“, s.o.)
Schulabschluss	school_merits	M	- (select)	Keiner (-), Hauptschule (H), Realschule (R), Fachhochschulreife (FH), Abitur (A), sonstiges (x)
Abgeschlossene Berufsausbildung	profession_learned	M	radio	Nein (n), Ja (j), Ja/Meisterbrief (M)
Beworben bei	applications	M	- (select)	Frankfurt UAS (UAS-F), Hochschule Darmstadt (HS-D), Hochschule RheinMain (HS-RM), Uni Frankfurt (Uni-F), Uni Mainz (Uni-Mz), andere (x). <u>Mehrere Auswahlen möglich</u> , „HS-RM“ voreingestellt
Angaben <i>vor</i> Klammern: Anzeigewerte Angaben <i>in</i> Klammern: Rückgabewerte (gilt auch für „Anmeldung“)				
Entfernung zur Hochschule (km)	distance	M	number	Minimum: 0
Verkehrsmittel zur Hochschule	commute_by	C	checkbox	Zu Fuß (F), per Fahrrad (R), mit ÖPNV (Ö), mit PKW (P), anders (x)
Tipp: collection_check_boxes				
Farbe des HSRM-Logos	color	M	color	
Zufriedenheit mit dem Studium (0-100%)	satisfied	M	range	Werte von 0 bis 100 in 5er-Schritten

Fragen & Aufgaben für die Abnahme

- Analysieren und kommentieren Sie den von Rails erzeugten HTML-Code
 - Wie werden die Vorgaben aus den Helper-Methoden umgesetzt?
 - Erklären Sie diese Umsetzungen während der Abnahme
- Wie hängt die am Server eingetroffene Information aus den Debug-Ausgaben mit den Formularinhalten zusammen?
 - Achten Sie unter „Request“ auf „Parameters“!
 - **ALLE** Ihre Formularinhalte sollten in „params“ **enthalten** sein!
- Welche (der hier verwendeten) neuen HTML5-Funktionen beherrscht Ihr Browser noch nicht?
 - Testen Sie mindestens zwei aktuelle Browser, z.B. Chrome, Firefox, Safari, Edge, Opera
 - Beschreiben Sie das Ersatz-Verhalten bei nicht vollständiger Implementierung!
- Was passiert bei Eingabe nicht zugelassener Daten?
 - Unterscheiden Sie zwischen den beiden Browsern
 - Ist das Verhalten jeweils korrekt?
 - Gelangen nicht zugelassene Daten bis zum Server?
- **Sie sollten Ihre Antworten beim Abnahmegespräch geben können**

Anmerkungen zum „Fragebogen“

- Mehrfachauswahlen
 - Die Erfüllung aller Vorgaben bei „applications“ und „commute_by“ ist deutlich komplizierter als beim Rest
 - Wenn Ihnen dies nicht gelingt, führt das zu keinem Punktabzug. Versuchen Sie aber, eine Lösung zu finden
- Herausforderungen:
 - „applications“ und „commute_by“ sollten Arrays (mit allen ausgewählten Angaben) als Rückgaben erhalten
 - Rails erzeugt standardmäßig auch einen leeren String in diesen Rückgaben. Versuchen Sie, dies abzuschalten
- Tipp: Die Lösungen lassen sich durch sorgfältiges Lesen der Rails-Dokumentationen finden, zusammen mit etwas Ausprobieren.
 - ½ Sonderpunkt bei Gelingen der vollständigen Umsetzung



Bedingungen

- Abgabe
 - Wegen des Feiertags am 13.5. erst in zwei Wochen (KW 20)
 - Jeweils vor Beginn Ihrer Praktikumsgruppe in der Abgabewoche
 - Hinweis: In KW 19 beginnt bereits die Bearbeitung des Rails-Tutorials (Übung 04)
- Art des Leistungsnachweises
 - Zu vergeben: **1 Punkt**
 - **Einzel-Arbeit – keine Teams!**
 - Abgabe der Dateien:
 - Im Rails-Verzeichnis „wba1“ ausführen:

```
$ rake log:clear  
$ rake tmp:clear  
$ cd ..
```
 - Sie sind nun im Elternverzeichnis von „wba1“. Jetzt noch ausführen:

```
# Ordner ./wba1 verpacken, ohne unnötige Unterordner:  
$ tar czf 03-wba1-<matnr>.tar.gz --exclude wba1/tmp \   
    --exclude wba1/node_modules --exclude wba1/log ./wba1
```

Datei „03-wba1-<matnr>.tar.gz“ abgeben
 - Achten Sie auf die Entfernung unnötiger Daten – Dateigröße von ca. 250 kB ist ok.
Dateien > 1 MB werden nicht angenommen!
 - Kurzes Abnahmegespräch / Online-Demo der korrekten Funktion
 - Online, per Breakout-Session.

Fortsetzungszeile

- **Zu HTML5-Formularen:**
 - **www.w3schools.com**
 - Tutorial „Learn HTML“, darin Sektion „HTML forms“ (incl. HTML5)
 - **<https://www.w3.org/TR/html52>**
 - Kapitel 4.10 „Forms“, insb. 4.10.5 „The **input** element“
- **Zu HAML**
 - **haml-lang.com**
 - „Tutorial“ und „Documentation“ genügen völlig
- **Zu SASS/SCSS**
 - **sass-lang.com**
 - „Tutorial“ und „Documentation“ genügen völlig
- **Zu Formular-Helfern in Rails**
 - <http://api.rubyonrails.org/>
 - dann Stichwortsuche oben links nach „text_field“ (liefert Seite mit allen FormHelper-Methoden)
 - bzw. nach „form_tag“ (liefert Seite mit FormTagHelper-Methoden)

- **Weiterführende Wünsche / Ausblick**
 - „session“-Technik zur Verbindung der Seiten
 - Persistenz, z.B. Speichern der Eingaben in einer Datenbank
 - Aktivierung abhängiger Felder
 - Feld „married_since“ nur einblenden im Fall „verheiratet“
 - Prüfung, ob password == password_confirmation
 - Fehlermeldung, falls nicht
 - Server-seitige Validierungen
- **Erst mit JavaScript und serverseitiger Software lösbar!**
 - Teile davon werden wir später nachholen