hci+d lab.

Ruby on Rails Workshop

Ruby on Rails 03

Joonhwan Lee
human-computer interaction + design lab.

오늘 다룰 내용

- 추가 Model 생성 및 데이터의 입력
- 모델 간의 관계 생성

온라인 주소록의 확장

Rails 실습

- 온라인 주소록 확장
  - 목표
    - 개별 contact 에 그룹을 할당한다
      - 예: 홍길동 (친구), 홍만석 (가족), 조성희 (동료)
    - group model 생성
    - group 입력/수정 form 생성
    - person model 수정
    - person controller/view 수정

데이터 베이스 설계

- Person Table
  - name: string
  - cell_no: string
  - email: string
  - address: string

name	cell_no	email	address
박OO	010-123-1234	park@...	...
강OO	010-222-2222	kang@...	...
전OO	010-321-4321	jun@...	...

데이터 베이스 설계

- Person Table
  - name: string
  - cell_no: string
  - email: string
  - address: string
  - group: string

name	cell_no	email	address	group
박OO	010-123-1234	park@...	...	동료
강OO	010-222-2222	kang@...	...	친구
전OO	010-321-4321	jun@...	...	가족

테이블의 연결

- Person Table 을 확장해서 group 정보 입력
- 장점: 새로 테이블을 생성하지 않아도 됨
- 단점: 자료의 수정, 추가가 복잡
  - 예: 동료 → 회사동료 (person 테이블 전체 수정하지 않고, group 만 수정하면 됨)

데이터 베이스 설계

- Person Table
  - name: string
  - cell_no: string
  - email: string
  - address: string
  - group_id: integer

name	cell_no	email	address	group_id
박OO	010-123-1234	park@...	...	3
강OO	010-222-2222	kang@...	...	2
전OO	010-321-4321	jun@...	...	1

데이터 베이스 설계

- Group Table
  - `id: integer`
  - `name: string`

id	name
1	가족
2	친구
3	동료

데이터 베이스 설계

- Group Table
  - `id: integer`
  - `name: string`

id	name
1	가족
2	친구
3	동료

name	cell_no	email	address	group_id
박OO	010-123-1234	park@...	...	3
강OO	010-222-2222	kang@...	...	2
전OO	010-321-4321	jun@...	...	1

Model 생성

- group 모델 생성

  `> rails generate model Group`

- db/migrate/

  2015xxxxxxxx_create_groups.rb 내용 수정

```
def change
  create_table :groups do |t|
    t.string      :name
    t.timestamps
  end
end
```

- id 는 자동으로 생성되므로 생략

Model 수정

- person 모델 수정

  > rails generate migration add_group_id_to_person

- db/migrate/

  2015xxxxxx_add_group_id_to_person.rb

  내용수정

```
def change
  add_column :people, :group_id,
  :integer
end
```

- model migration: rake db:migrate

Model 간 관계 설정

- has_many 와 belongs_to
  - 한 레코드가 다른 테이블에서 참조되는 방식으로 관계가 맺어질 때, one-to-many 혹은 one-to-one 의 관계가 성립된다
- 예: 블로그 아티클과 커맨트의 관계
  - blog_article_01
    -- comment_01
    -- comment_02
    -- comment_03 ...
  - an article has many comments; a comment belongs to an article

Model 간 관계 설정

- User & Post & Comment
  - a user has many posts; a post belongs to a user
  - a user has many comments; a comment belongs to a user
- User & Profile
  - a user has one profile; a profile belongs to a user
- Person & Group
  - a group has many people; a person belongs to a group

Model 간 관계 설정

- app/models/person.rb 수정

```
class Person < ActiveRecord::Base
  belongs_to :group
end
```

- app/models/group.rb 수정

```
class Group < ActiveRecord::Base
  has_many :people
end
```

- 모델의 단/복수형에 주의

Model 간 관계 설정

- 관계 설정하는 이유?
  - 복잡한 SQL 구문을 작성하지 않고 양쪽테이블의 레코드를 손쉽게 검색하고 수정하기 위해서

```
@post = Post.find(2)
@post.comments

@group = Group.find(1)
@group.people

→ select * from people where group_id == 1
```

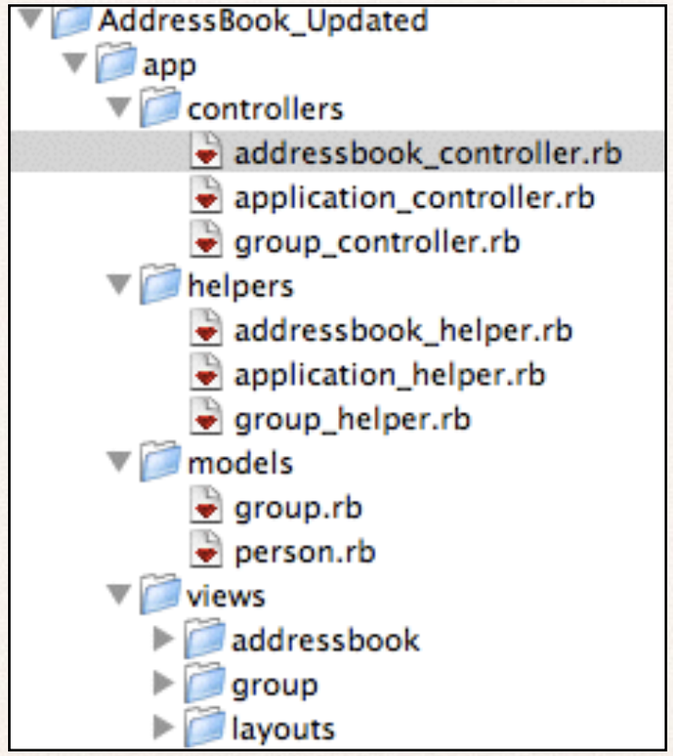
Controller 생성

- group을 관리하기 위한 컨트롤러 생성
- 생성할 컨트롤러의 구조
  - groups (그룹 컨트롤러)
    - list : 그룹 리스트를 보여줌
    - view: 그룹의 상세 정보를 보여줌
    - new : 새로운 그룹의 입력
    - edit : 그룹의 수정

Controller 생성

- 컨트롤러 생성

```
> rails generate controller groups
list view new edit
```



Controller 수정

- Route 수정
  - config/routes.rb
    add following line:
    ```
    resources :groups
    ```
- View 수정
  - new.html.erb, list.html.erb, edit.html.erb, view.html.erb 파일 추가됨
    - new.html.erb, list.html.erb, edit.html.erb → people 로 부터 복사

Controller 수정

- open group_controller.rb

```
def new
  @group = Group.new
end
```

```
def create
  @group = Group.new(group_params)
  @group.save
  redirect_to groups_list_path
end

private
def group_params
   params.require(:group).permit(:name)
end
```

View 수정

- open new.html.erb (copy new.html.erb from people controller)

```
<h1>새로운 그룹등록</h1>
<%= render 'form' %>
<%= link_to '돌아가기', groups_path %>
```

- open _form.html.erb (copy _form.html.erb from people controller)
- name 과 관련한 필드만 제공하고 나머지는 삭제

```
<%= form_for(@group) do |g| %>
  <div>
     <label>Name: </label>
     <%= g.text_field    :name %>
  </div>
  <%= g.submit %>
<% end %>
```

Controller 수정

- open group_controller.rb

```
def list
  @groups = Group.all
end
```

- list view 의 수정 (copy list.html.erb from people controller)

```
<% for g in @groups %>
  <tr>
    <td><%= g.name %></td>
  <td><%= link_to '수정',
          groups_edit_path(:id => g) %></td>
  </tr>
<% end %>
<%= link_to '새 그룹 등록', groups_new_path %>
```

Controller 수정

- open groups_controller.rb (copy actions from people controller)

```
def edit
  @group = Group.find(params[:id])
end

def update
  @group = Group.find(params[:id])
  @group.update(group_params)
  redirect_to groups_list_path
end
```

View 수정

- open edit.html.erb (copy edit.html.erb from people controller)

```
<h1>그룹 수정</h1>
<%= render 'form' %>
<%= link_to '돌아가기', groups_path %>
```

Controller 수정

- open groups_controller.rb

```
def view
  @group = Group.find(params[:id])
end
```

View 수정

- create a new file: view.erb
  - 다음의 내용을 추가

```
<h3>
  <%= @group.name %> (총 <%=
@group.people.size
  %> 명)
</h3>
<hr>
```

View 수정

- list.html.erb 수정

```
<td><%= g.name %></td>
<td>
 <td><%= link_to '수정',
groups_edit_path(:id => g) %> | <%=
link_to '보기', groups_view_path(:id
=> g) %></td>
</td>
```

View 수정

- <hr> 다음에 다음의 내용을 추가

```
<% if @group.people.size != 0 %>
<table>
  <tr>
    <th>name</th>
    <th>cellphone number</th>
    <th>email address</th>
    <th>address</th>
  </tr>
<% for p in @group.people %>
  <tr>
    <td><%= p.name %></td>
    <td><%= p.cell_no %></td>
    <td><%= p.email %></td>
    <td><%= p.address %></td>
  </tr>
<% end %>
</table>
<% end %>
```

Controller 수정

- people controller 에 group 을 추가하고 표시할 수 있도록 수정

```
def new
  @person = Person.new
  @groups = Group.all
end
def edit
  @person = Person.find(params[:id])
  @groups = Group.all
end
```

Controller 수정

- people controller 에 group 을 추가하고 표시할 수 있도록 수정

```
private
def person_params
    params.require(:person).permit
    (:name, :cell_no, :email, :addre
    ss, :group_id)
end
```

View 수정

- _form.html.erb 수정 (people 콘트롤러)
- `<% form_for... %>` 아래에 다음의 내용 추가

```
<label>group</label>
<%= f.select(:group_id, @groups.map
{|g| [g.name, g.id]}) %>
```

View 수정

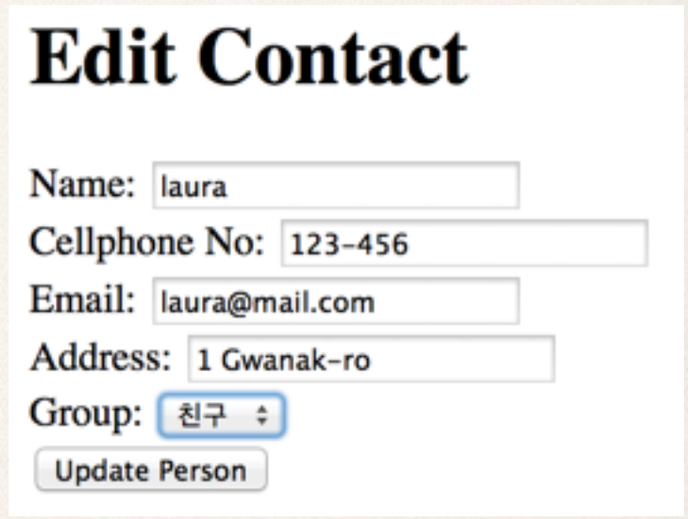
- list.html.erb 에 컬럼 추가
- `<th>이름</th>` 위에
  `<th>그룹</th>` 추가
- `<td><%= person.name %></td>` 위에
  `<td><%= person.group.name %></td>` 추가

→ 에러 발생: group이 지정된 person이 하나도 없어서.

→ 일단 `<%= person.group.name %>` 삭제

People Controller 수정

- person_params 에 group_id 추가
  - params.require(:person).permit(:name, :ce
    ll_no, :email, :address, :group_id)
- 모든 컨택트를 선택하여 edit
  - group 을 하나 선택
- 다시 list.html.erb 로 가서 다음 추가
  - <%= p.group.name %>



nil 에러를 피하는 방법

- 저장된 데이터를 불러오려는데 데이터가 없는 경우가 발생할 수 있음. (`nil`)
  - 예: `<td><%= person.group %></td>`
- 이런 경우, 당연히 데이터를 입력하면 에러는 사라지지만, 다음과 같이 에러 처리를 하는게 바람직함.

```
<td>
  <% if person.group != nil %>
    <%= person.group.name %>
  <% else %>
    no data
  <% end %>
</td>
```

Summary

- group 모델 생성
- person 모델 수정
- 두 개의 모델의 관계 설정
- 수정된 모델의 migration
- person 의 controller 와 view 를 카피하여 group 의 controller/view 생성
- group 추가를 위해 person 의 controller 와 view 수정

Questions?