Migrations (Chapter 23)

The notes in this document are based on the online guide at http://guides.rubyonrails.org/migrations.html and the Agile Web Development with Rails, 4th edition, book.

Migration is a framework that allows us to modify, or migrate, the database schema over time.

Whenever a new model (or scaffold) is generated, it creates a new file in the db/migrate folder. Let's say we create a new model named Book. The migration file is called something like 20111010220648_create_books.rb

Migrations are named with a numeric prefix that represents the exact time when the migration was created – this number uniquely identifies the specific migration in the project.

EXAMPLE:

```
20111010220648_create_books.rb

yyyymmddhhmmss_create_books.rb
```

Open the migration file to see the content.

New migrations can be used to modify or add fields in an existing model.

To specify the appropriate version and the corresponding migration you can invoke the following rake command:

```
> rake db:migrate VERSION=20111010120000
```

These tasks check whether the migration has already run, so for example `db:migrate VERSION=20111010120000` will do nothing if Active Record believes that `20111010120000` has already been run.

Adding more fields

1) Use **generate migration** (similarly to generating controllers, models, etc) and give a descriptive name for the migration following a specific syntax described below.

   Example: Add new fields **edition** and **pagenum** to the Book model

   ```
   > rails generate migration add_edition_and_pagenum_to_books
   edition:integer pagenum:integer
   ```

   ADD syntax: the key to use RAILS' built in functionalities is to use the following syntax for the ADD migration name:

   add_*newfield1*_**and**_*newfield2*_**and**_*newfield3*_**to**_*tablename*

   The generator creates a migration class in db/migrate prefixed by a number identifying when the migration was created.

2) Apply the migration and make changes to the database using the rake software

   > rake db:migrate

Note for scaffold projects, you need to modify the following templates to include the new columns:

- **_form_html_erb:** add form fields for the added columns.
- **show.html.erb:** add code to display values for the new columns
- **index.html.erb:** add code to display values for the new columns

Inside the migration

Open the xxxxxxxxxxx_add_edition_and_pagenum_to_books.rb file to see the content. Rails uses the **add_column** helper to create a new field in the table.

RAILS 3.1.0

```
class AddEditionAndPagenumToBooks < ActiveRecord::Migration
  def change
    add_column :books, :edition, :integer
    add_column :books, :pagenum, :integer
  end
end
```

change method "knows how to migrate your database and reverse it when the migration is rolled back without the need to write a separate down method"

RAILS 3.0.9

```
class AddEditionAndPagenumToBooks < ActiveRecord::Migration
  def self.up
    add_column :books, :edition, :integer
    add_column :books, :pagenum, :integer
  end

  def self.down
    remove_column :books, :edition, :integer
    remove_column :books, :pagenum, :integer
  end
end
```

The down method of your migration reverts the transformations done by the up method.

To revert back to previous version of database you can use

```
> rake db:rollback
```

Removing field/column from database

We can use RAILS built-in functionalities again. We need to generate a migration and use the `remove_`*`fieldname`*`_from_`*`tablename`* syntax.

Example: Remove "pagenum" field from the Book model

1) Use **generate migration** and give a descriptive name for the transformation.

```
> rails generate migration remove_pagenum_from_books pagenum:integer
```

RAILS creates a migration file that uses the remove_column helper to remove the pagenum field.

```
class RemovePagenumFromBooks < ActiveRecord::Migration
  def up
    remove_column :books, :pagenum
  end

  def down
    add_column :books, :pagenum, :integer
  end
end
```

2) Apply the migration and make changes to the database using the rake software

```
> rake db:migrate
```

Renaming existing fields (columns)

There is no magic keyword for renaming columns. We need to generate a migration and use the `rename_column` method defined in RAILS.

```
rename_column :tablename, :oldname, :newname
```

Example: rename "title" field as "btitle" in the Book model

1) Use **generate migration** and give a descriptive name for the transformation.

```
> rails generate migration rename_title_in_books
```

2) Edit rename_title_in_books.rb migration file in db/migrate

```
class RenameTitleInBooks < ActiveRecord::Migration
  def up
  end

  def down
  end
end
```

3) Add rename_column method

```
class RenameTitleInBooks < ActiveRecord::Migration
  def up
      rename_column :books, :title, :btitle
  end

  def down
       rename_column :books, :btitle, :title
  end
end
```

4) Apply the migration and make changes to the database using the rake software

```
> rake db:migrate
```

NOTE: the down method should revert the change if needed

Changing column type

There is no magic keyword for changing column types. We need to generate a migration and use the change_column method defined in RAILS.

```
change_column :tablename, :fieldname, :newtype
```

Example: Change "price" type into a :float type in the Book model

1) Use **generate migration** and give a descriptive name for the transformation.

```
> rails generate migration change_price_in_books
```

2) Edit change_price_in_books.rb migration file in db/migrate

```
class ChangePriceInBooks < ActiveRecord::Migration
  def up
  end

  def down
  end
end
```

3) Add change_column method

```
class ChangePriceInBooks < ActiveRecord::Migration
  def up
      change_column :books, :price, :float
  end

  def down
       change_column :books, :price, :decimal
  end
end
```

4) Apply the migration and make changes to the database using the rake software

```
> rake db:migrate
```

NOTE: the down method should revert the change if needed

Note for scaffolded projects.

The modifications will not migrate to the templates in app/views/books/.

We need to manually modify the code in the following files:

1) **_form_html_erb,**
2) **show.html.erb,**
3) **index.html.erb**

since table columns have changed.