

WEB+DBシステム(入門編)



第5回(2016年5月12日)

データベース言語SQL

今日のテーマ

- データベースとは何か
- データベース言語SQLについて学ぶ

- 新たにテーブルを一つ追加する。

データベースとは

- コンピュータによって書き込みや読み出しを行えるように構成されたデータの集まり
- データとは？
 - 電子的や磁気的な信号で管理される情報体のことを指し、絵やプログラムや音楽、文章などは全てデータと言える。
 - Dataは、Datumの複数形！

ファイルでいいんじゃないの？

□ ファイル【File】とは何か？

- コンピュータでの管理上、データの集まりをひとつの単位としてまとめたもの

□ データベースとファイルとは、どう違う？

- 呼び名が違う？
- ファイルが集まるとデータベース？

個別ファイルの問題点

- データに変更があった時、関連するファイルすべてを修正しなければならない。
 - 作業が複雑になる。
 - 修正漏れがあると、ファイルによって内容が異なる場合が出て来る。

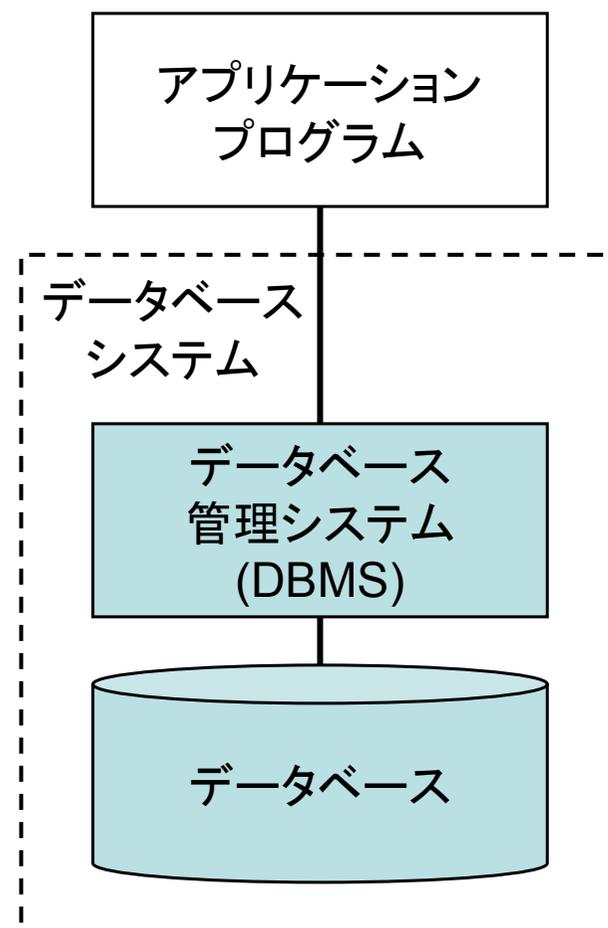
- 記憶容量の無駄
 - それぞれのファイルが独立して存在すると、ファイル管理領域に無駄が生じる

個別ファイルの問題点の解決策

- 本質的な解決
 - データを一つの場所に記録して共有すること
- One fact in one place.
 - (一つの事実は一つの場所に記録)
- One fact in **Only** one place.
 - (一つの事実は一つの場所だけに記録)
- このような設計思想で作られたのがデータベース

DBMS

- DBMSはデータベースを管理する機能を有する。
- Database Management System
- DBMSとデータベースを合わせて、データベースシステムと呼ぶ場合がある。



SQLはDBの操作言語

□ SQL

- Structured Query Language
- SQLは、他のアプリケーションプログラムや人がDBMSにアクセスしてデータベースを操作するための言語
- DML(Data Manipulation Language)という言語分類がある。SQLはDMLの一つ

□ Sqlite3は、SQLの実装の一つ

- Oracle, SQL Server, DB2などが有名
- フリーウェアでは、postgresqlやMySQLなどがある。

データのモデリング

□ データのモデル化とは？

■ データ構造を表現する

- 「項目」どうしがどんな関係にあるか。

■ データの整合性制約を記述する

- 「整合性」 -- 部分・部分を合わせても矛盾がないこと

■ データを操作する方法

- 自動的に計算されるものや入力が必要なものなど、区別する。

□ モデリングによって、スキーマを決める。

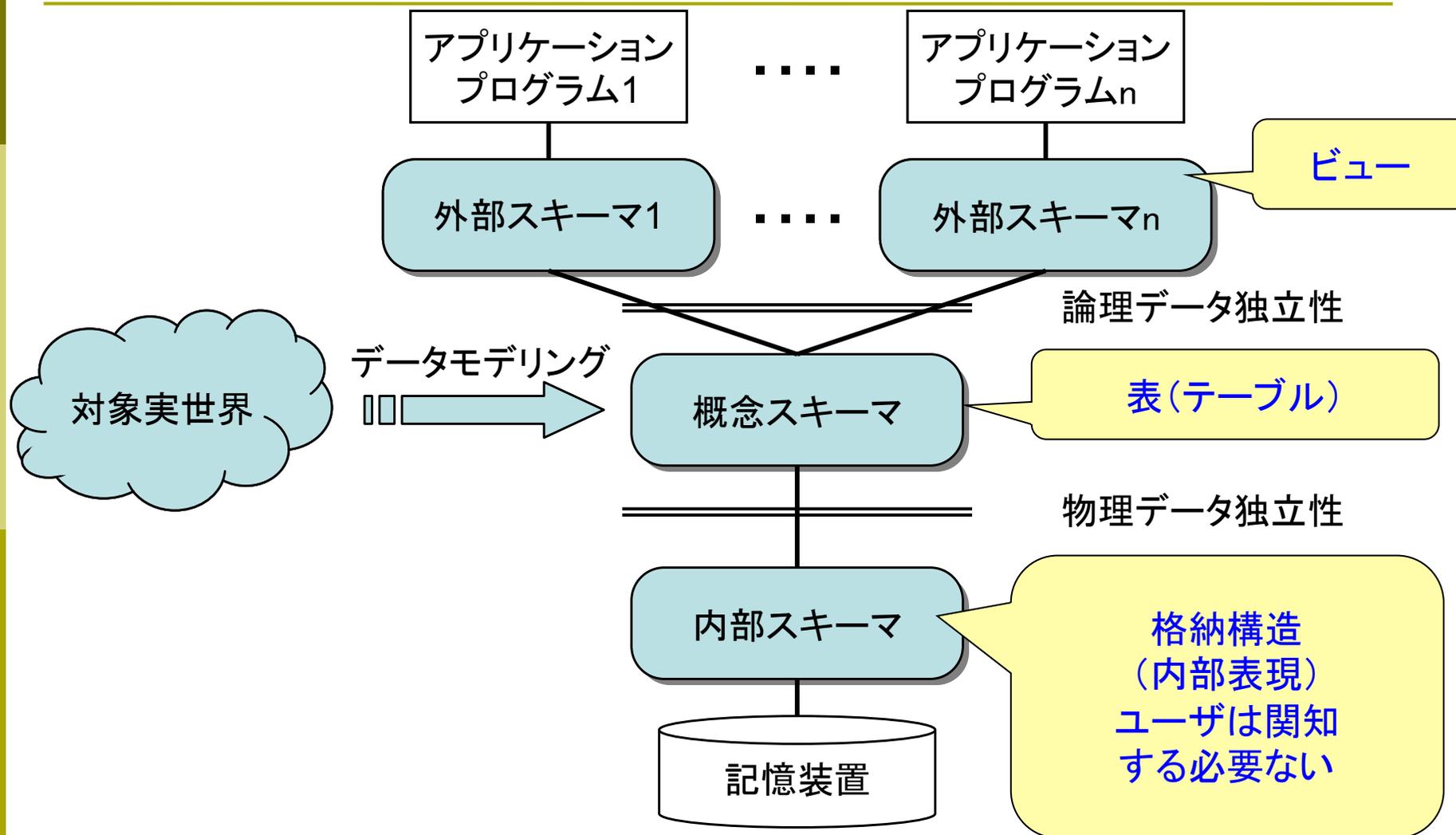
スキーマとは

□ Schema

- アプリケーションデータモデルを検討して得られたデータの構造
- これから扱いたいデータが、どのような関係になっているか。

- スキーマには、外部スキーマや概念スキーマなどがある。

ANSI/X3/SPARC3層スキーマアーキテクチャ



速水治夫著:リレーショナルデータベースの実践的基礎、コロナ社、2008より

ORマッピング

□ Object Relational Mapping

- オブジェクト指向言語で、クラス構造が直接データベースに結びついていること
- Railsなどの開発プラットフォームで利用できる。

□ Ruby on Railsでは、モデル=クラス

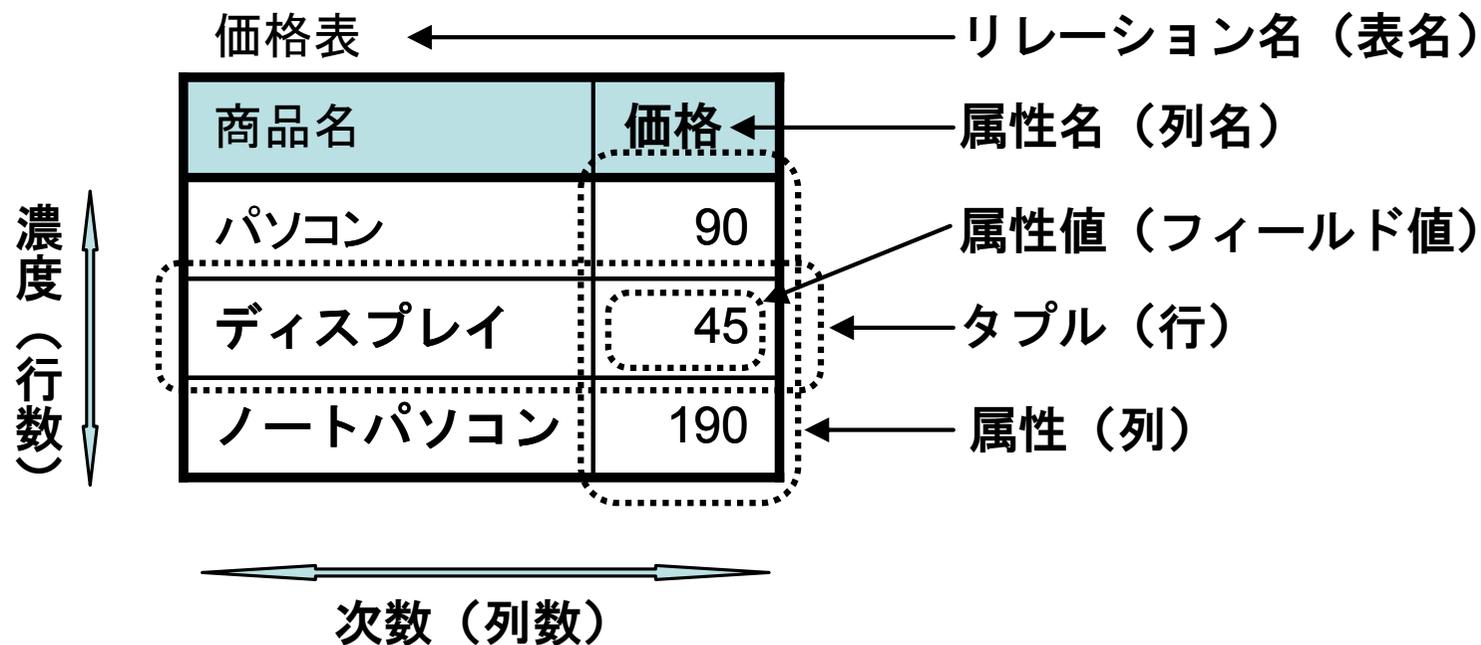
- スキーマ設計が、そのままRubyのクラス設計になる。

DBとテーブル(表)

- 情報は、テーブルに格納される。
- テーブルのことをリレーションと呼ぶこともある。(RDB)
 - 表には「属性(列/Column)」が定義される。
 - 属性は「項目名」である。
 - データの「行(タプル)」には実際のデータの1セットが入る。
 - インスタンス(instance)

リレーション(テーブル)の構造

- 「リレーショナルDB」での定義とは別に、一般にSQLでは()内の呼び方をする



主キー (primary key)

- 単純に「ID」を設定すると考えると良い。
 - 「会員番号」「学生証番号」など
 - Rails環境では、主キーは常に **id** となる。

- ユニークということ
 - Unique – 「唯一の」
 - その値を持つデータは、一つしかない
 - 「その学生証番号の学生は一人しかいない」という「番号」を与える、ということ。

Unique

1. 【形】ただ一つだけの、ほかに存在しない、唯一の
2. 並ぶものがない、比類のない、優れた
3. 〈話〉変わった、珍しい、ユニークな
4. [場所・もの・人などに]固有の、特有の
5. 《コ》固有の
6. 《数学》一意的な

<http://www.alc.co.jp> から検索

sqlite3の起動と終了

- コマンドプロンプトで、sqlite3を走らせる
 - cd db
 - **sqlite3 development.sqlite3**
- sqlite3から抜ける時は
 - sqlite3 >> **.exit**
- 操作方法を確認する時は、**.help**

```
[root@cisnote memopad]# pwd
/root/Documents/rails4work/memopad
[root@cisnote memopad]# cd db
[root@cisnote db]# sqlite3 development.sqlite3
SQLite version 3.6.20
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> █
```

しつこいようですが...

Sqlite3から抜け出すコマンドは

.exit

です。また、コマンドがわからない時は

.help

です。

データベースを作る/消す

- データベースを作る(sqlite3では、この操作は×)
 - mysql >> create database データベース名;
 - 例: create database friendDB;
 - friendDBという名前のDBを作る。ここでのデータベースとは、中に複数のテーブルを持つ、大きな入れ物を指す。

- Sqlite3の場合は、
 - コマンドプロンプトで、
sqlite3 データベース名.sqlite3
 - と入力する。
 - 開く場合と同じコマンドで、実際の書き込みがあると、ファイルが作成される。

データベースを消す

- 作成されたデータベースを削除する
 - mysql >> drop database データベース名;
 - 例: drop database friendDB;
 - 内部のテーブル諸共、全部消滅するので、**要注意!**

Sqlite3の場合は、
データベース名.sqlite3
のファイルを削除して下さい。

Rails4の環境とDB

Migrationした結果、データベースが作成されます。

ある時点でのDBをそっくりバックアップする場合には、`development.sqlite3`をコピーして保存しておけば、その時点のデータベースの状態に戻せます。

但し、rails4の側での状態と整合性をとる部分は、各自の責任で行って下さい。

データベースの一覧を見る

□ データベースの一覧を見る

- mysql >> `show databases;`

- sqlite> `.databases`

- Sqlite3では、開く際にデータベースファイルを指定

データベースの使用を宣言する

- `mysql >> use データベース名;`
- 例: `use friendDB;`

`connect データベース名;`

- でも、同様の結果を得る。
- Sqlite3の場合は、開く際にデータベースを指定

テーブルを作る(1)

- `sqlite> create table テーブル名(`
 列名 [データ型] [列属性],
);
 テーブルの要素の数だけ、
 - 列名 [データ型] [列属性] の記述を繰り返す。
- データ型 : int, decimal, text, varchar, datetime, timestamp, blobなど
- 列属性 : primary key, not nullなど

テーブルを作る(2)

会員IDと氏名だけの表を定義してみる。

```
create table MemberT(  
MemberID char(6) primary key,  
MemberName char(16) not null );
```

上記を確認する。(MySQLの場合)

```
show columns from MemberT;
```

Field	Type	Null	Key	Default	Extra
MemberID	char(6)	NO	PRI	NULL	
MemberName	char(16)	NO		NULL	

テーブルを確認する。

- (選んだデータベースの中に)どんなテーブルがあるか確認する
 - mysql >> **show tables;**
 - Sqlite> .tables

- 自分が作ろうと思ったテーブルがあったなら、そのフィールドがどうなっているか確認する。
 - mysql >> **show columns from テーブル名;**
 - sqlite> .schema テーブル名

テーブルの中身を見る

- select文を使う。
 - selectの使い方(select文の書き方)をマスターできるとデータベースのスキルは中級以上！
 - 様々な述語を用いて、複雑な検索でも実行できる。
 - 論理を駆使して、高速な検索を行う。
- 全てのレコードを表示させる場合
 - sqlite> `select * from テーブル名;`
 - 例: `select * from friends;`
- *(アスタリスク)はワイルドカードで、selectの後には通常は列名を並べる。

データを書き込む。(1)

□ `sqlite> insert into テーブル名`

`values (列1の値, 列2の値 ...);`

テーブル定義で記述した「列」の数だけ、テーブル定義で記述した順番に、列挙する。

(標準のSQLの文なので、MySQLでも同様に使うことができる。)

データを書き込む。(2)

少し前のページで定義した、会員IDと氏名だけの表に、直接入力
力でデータを登録する。

```
insert into MemberT values (  
    'A001', 'Aoyama' );
```

```
insert into MemberT values (  
    'B002', 'Konaka' );
```

データの中身を確認する。

```
select * from MemberT;
```

```
sqlite> create table memberT ( id varchar(32) primary key, name varchar(64) );  
sqlite> insert into memberT values ( 'A001', 'Aoyama Aya' );  
sqlite> insert into memberT values ( 'B002', 'Konaka Kanako' );  
sqlite> select * from memberT;  
A001|Aoyama Aya  
B002|Konaka Kanako  
sqlite> █
```

データ更新の方法

- `sqlite> update テーブル名`
`set カラム名='値' [, カラム名='値', ...]`
`where 条件式;`
- 「条件式」を満たすようなレコード(を全部)、set節で記した値に設定する。
(標準のSQLの文なので、MySQLでも同様に使うことができる。)
- **重要な注意:** where節を書かないと、全部のレコードが同じ値にsetされてしまうので、気をつける。
- (しかも、元には戻せない！)

rails 以前 / 以後のプログラム

- sql用の関数コールをスクリプトなどのプログラム中にはめ込む。
- 「お任せ」ではなく、一つ一つのステップごとに、SQLを発行して、データの格納・参照・検索・消去などの操作を行った。
- rubyでも、「手順」それ自体はあまり変わらないが、クラスのインスタンス値を操作すれば、DBに直結させられる。(DBを意識せずに処理できる。)

今日の演習課題

- 今週と来週とで、「メモ帳」のそれぞれのメモに、「分類」コードを入れます。
 - 「至急」、「要連絡」、「宿題」、「訪問客」、「飲み会」、「就活アポ」など
- 今週は、「分類コード」のテーブルを追加し、結果をSQLと画面のそれぞれで確認します。
- 今週はまだ、「メモ帳」と「分類コード」のテーブルの連結は行いません。

Categoryモデルのscaffoldを生成

□ Categoryを生成する方法

- rails generate scaffold Category name:string

- 入力用の画面なども、一緒に生成する。
- まだコマンド実行しないで下さい。

□ scaffoldで、画面とモデルの両方を生成する。

□ 先走らないで！

次のページの注意を守らないと、
先週までの作業の一つが、きれいに消えてしまいます！

Scaffoldを生成する場合

□ Categoryのscaffoldを作る。

`rails generate scaffold Category name:string`

- Categoryというmodelを生成し、その際にviewsや、データ処理用の画面(erb)、テスト環境なども一括して生成しろ、というスクリプトをrailsが実行
- 生成するmodelの名前はCategory(大文字)
- 単数形の単語を使います。
- 画面入力用のフィールドは、nameだけ。
- Scaffold.css.scssは「上書きしない」ように
- 上書きすると、先週の作業の一部が消える。

rails generate scaffold Category name:stringの実行画面

```
kobayashi-ikuo-no-MacBook:memopad kobayashi$ rails generate scaffold Category name:string
  invoke  active_record
  create  db/migrate/20120524222048_create_categories.rb
  create  app/models/category.rb
  invoke  test_unit
  create  test/unit/category_test.rb
  create  test/fixtures/categories.yml
  route  resources :categories
  invoke  scaffold_controller
  create  app/controllers/categories_controller.rb
  invoke  erb
  create  app/views/categories
  create  app/views/categories/index.html.erb
  create  app/views/categories/edit.html.erb
  create  app/views/categories/show.html.erb
  create  app/views/categories/new.html.erb
  create  app/views/categories/_form.html.erb
  invoke  test_unit
  create  test/functional/categories_controller_test.rb
  invoke  helper
  create  app/helpers/categories_helper.rb
  invoke  test_unit
  create  test/unit/helpers/categories_helper_test.rb
  invoke  assets
  invoke  coffee
  create  app/assets/javascripts/categories.js.coffee
  invoke  scss
  create  app/assets/stylesheets/categories.css.scss
  invoke  scss
  conflict app/assets/stylesheets/scaffolds.css.scss
  Overwrite /Users/kobayashi/Aptana3Work2/memopad/app/assets/stylesheets/scaffolds.css.scss? (enter "h" for help) [Ynaqdh] n
  skip   app/assets/stylesheets/scaffolds.css.scss
kobayashi-ikuo-no-MacBook:memopad kobayashi$
```

Memosの作成の時の
scaffold.cssを、
上書きするか聞かれたら
上書きしないので、
Nを入力する。

さもない、先週までの
作業の一部が消えます！

migrationの実行

□ コマンド

- rake db:migrate

□ で、作成されたデータベースのテーブルです。
migrationの状態を確認するのは

- rake db:migrate:status

```
[root@cisnote memopad]# rake db:migrate
== 20140515002604 CreateCategories: migrating =====
-- create_table(:categories)
   -> 0.0053s
== 20140515002604 CreateCategories: migrated (0.0054s) =====

[root@cisnote memopad]# rake db:migrate:status

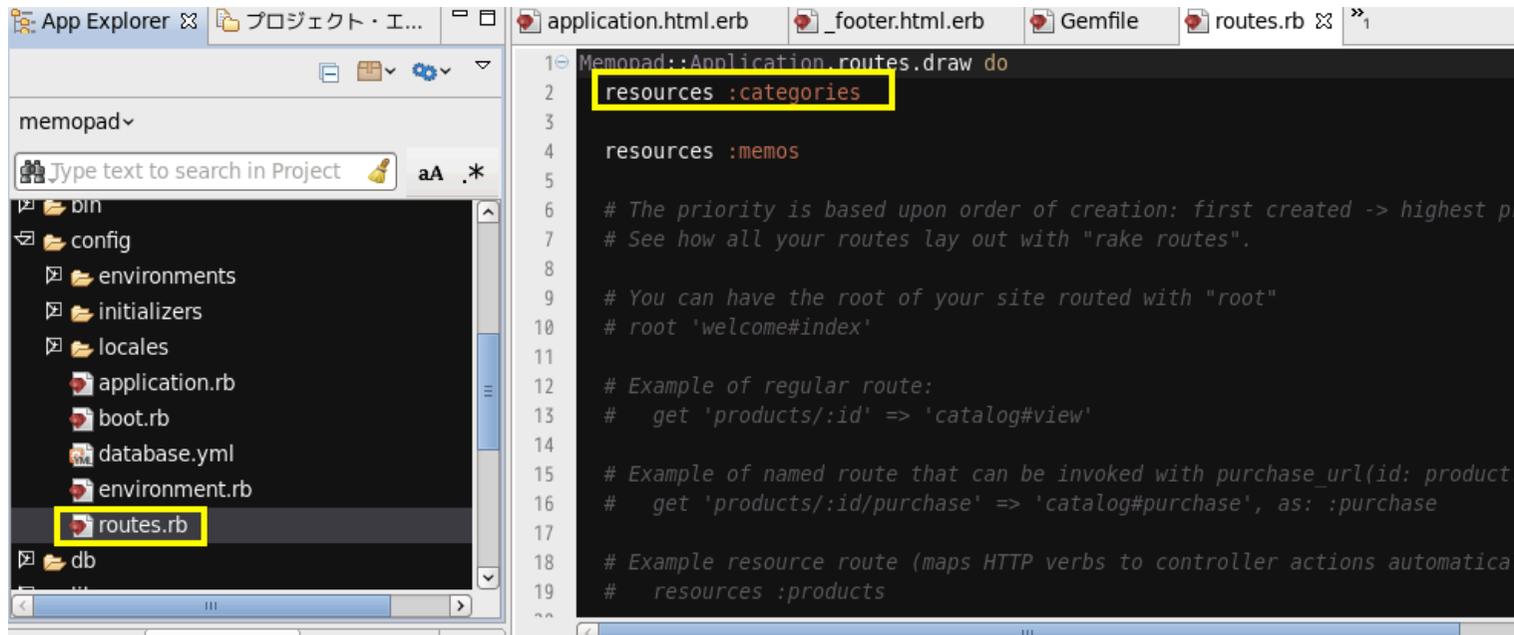
database: /root/Documents/rails4work/memopad/db/development.sqlite3

Status  Migration ID  Migration Name
-----
up      20140419162659  Create memos
up      20140515002604  Create categories

[root@cisnote memopad]#
```

リンク情報のマップを更新する

- config/routes.rbに
resources :categories
 - を追加されていることを確認する。(自動生成済み)
- この記述がないと、pathが定義されていない(routing error)が発生する。



```
Memopad::Application.routes.draw do
  resources :categories
  resources :memos

  # The priority is based upon order of creation: first created -> highest priority
  # See how all your routes lay out with "rake routes".

  # You can have the root of your site routed with "root"
  # root 'welcome#index'

  # Example of regular route:
  # get 'products/:id' => 'catalog#view'

  # Example of named route that can be invoked with purchase_url(id: product.id)
  # get 'products/:id/purchase' => 'catalog#purchase', as: :purchase

  # Example resource route (maps HTTP verbs to controller actions automatically)
  # resources :products
end
```

Routing情報の確認

コマンドプロンプトで

`rake routes`

と入力する。URLの一部となるパスがわかります。

```
[root@cisnote memopad]# rake routes
  Prefix Verb  URI Pattern          Controller#Action
categories GET    /categories(.:format) categories#index
          POST   /categories(.:format) categories#create
new_category GET    /categories/new(.:format) categories#new
edit_category GET    /categories/:id/edit(.:format) categories#edit
  category GET    /categories/:id(.:format) categories#show
          PATCH  /categories/:id(.:format) categories#update
          PUT    /categories/:id(.:format) categories#update
          DELETE /categories/:id(.:format) categories#destroy
  memos GET    /memos(.:format)      memos#index
          POST   /memos(.:format)      memos#create
  new_memo GET    /memos/new(.:format)  memos#new
  edit_memo GET    /memos/:id/edit(.:format) memos#edit
  memo GET    /memos/:id(.:format)  memos#show
          PATCH  /memos/:id(.:format)  memos#update
          PUT    /memos/:id(.:format)  memos#update
          DELETE /memos/:id(.:format)  memos#destroy
[root@cisnote memopad]#
```

第2のテーブルの準備完了！

- 修正がうまくいくと、
 - <http://127.0.0.1:3000/categories>
- のURLで、第2のテーブル(categories)の表示、編集、追加ができるようになる。



SQLでの結果確認

- sqlite3 に接続して、結果を表示する。
 - cd db
 - sqlite3 development.sqlite3
 - テーブルの一覧を見る
 - `sqlite> .tables`
 - 出力のファイルを指定する
 - `sqlite> .output output.sql`
 - データを出力する
 - `sqlite> .dump`
 - `sqlite> .exit`
- Linux上で、db/output.sqlを開いてみる。

Dumpの結果



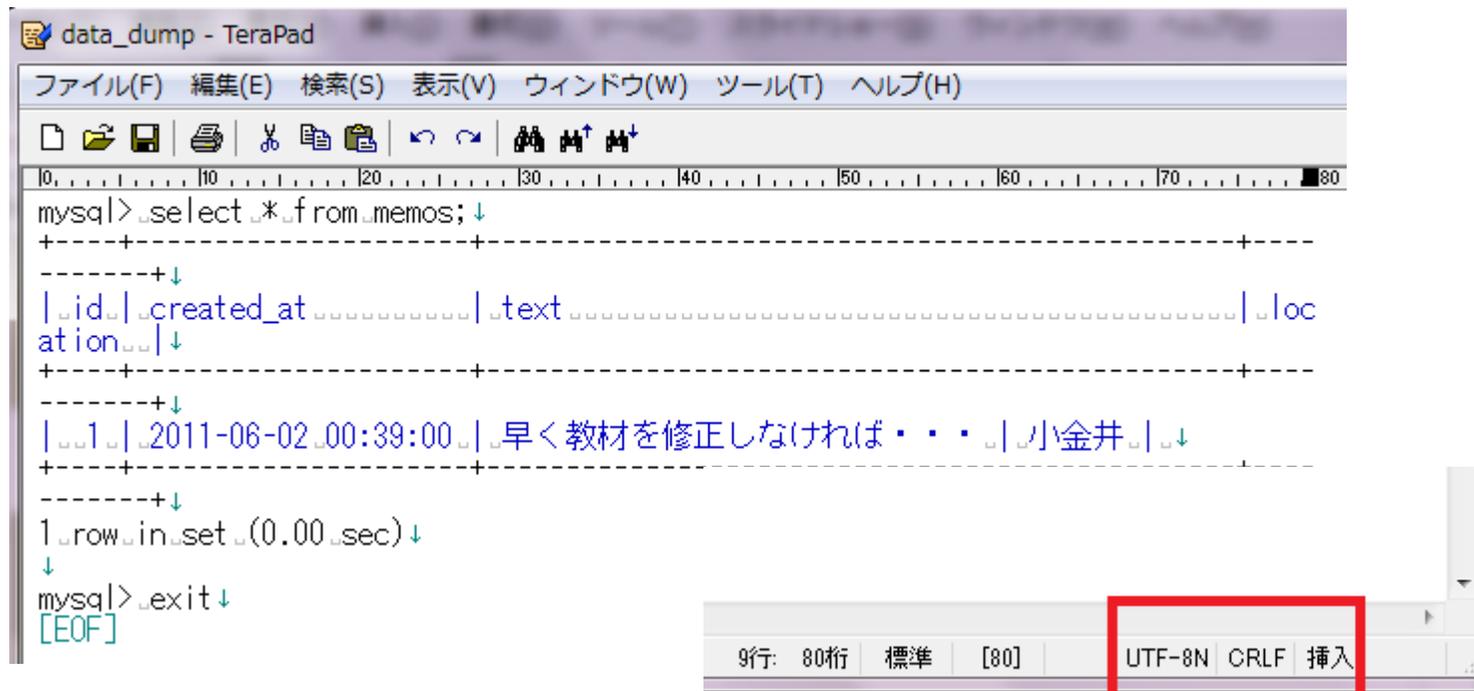
```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE "schema_migrations" ("version" varchar(255) NOT NULL);
INSERT INTO "schema_migrations" VALUES('20140419162659');
INSERT INTO "schema_migrations" VALUES('20140515002604');
CREATE TABLE "memos" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "title" text,
"name" varchar(255), "created_at" datetime, "updated_at" datetime);
CREATE TABLE "categories" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "name"
varchar(255), "created_at" datetime, "updated_at" datetime);
CREATE UNIQUE INDEX "unique_schema_migrations" ON "schema_migrations" ("version");
COMMIT;
```

Dumpとは？

- dump … 画面表示や、データの中身を、そのままの形で吐き出すこと。
- Output.sqlの中身は、SQLの言語でデータの中身を他のシステムに移せるプログラムになっているので、移植が用意になります。
- teeは、Linuxなどで使える中間経過保存用のコマンド
 - tee … コマンドの実行経過をダンプする。
 - Mysqlでは、teeのコマンドが使える。

output.sqlの中身を見る。

- WindowsはShift-JISの文字コード体系
 - テキストエディタが、UTF-8の文字コードとして認識しないと正しく表示されないことを確認して下さい。



The screenshot shows a TeraPad window titled "data_dump - TeraPad". The menu bar includes "ファイル(F)", "編集(E)", "検索(S)", "表示(V)", "ウィンドウ(W)", "ツール(T)", and "ヘルプ(H)". The toolbar contains icons for file operations and editing. The main text area shows a MySQL command prompt session:

```
mysql> select * from memos; ↓
+-----+-----+-----+-----+-----+
| id | created_at | text | location |
+-----+-----+-----+-----+
| 1 | 2011-06-02 00:39:00 | 早く教材を修正しなければ・・・ | 小金井 |
+-----+-----+-----+-----+
1 row in set (0.00 sec) ↓
mysql> exit ↓
[EOF]
```

At the bottom of the window, the status bar shows "9行: 80桁 標準 [80] UTF-8N CRLF 挿入". The "UTF-8N CRLF 挿入" part is highlighted with a red box, indicating the current encoding and line ending settings.

お疲れさまでした。

- 今日は、データベースとは何かから始めて、
- WEB+DBの演習環境に第2のテーブルを
- 追加しました。

- 2度目の作業もありますが、かなりのボリュームが
- ありました。

演習環境の発展課題

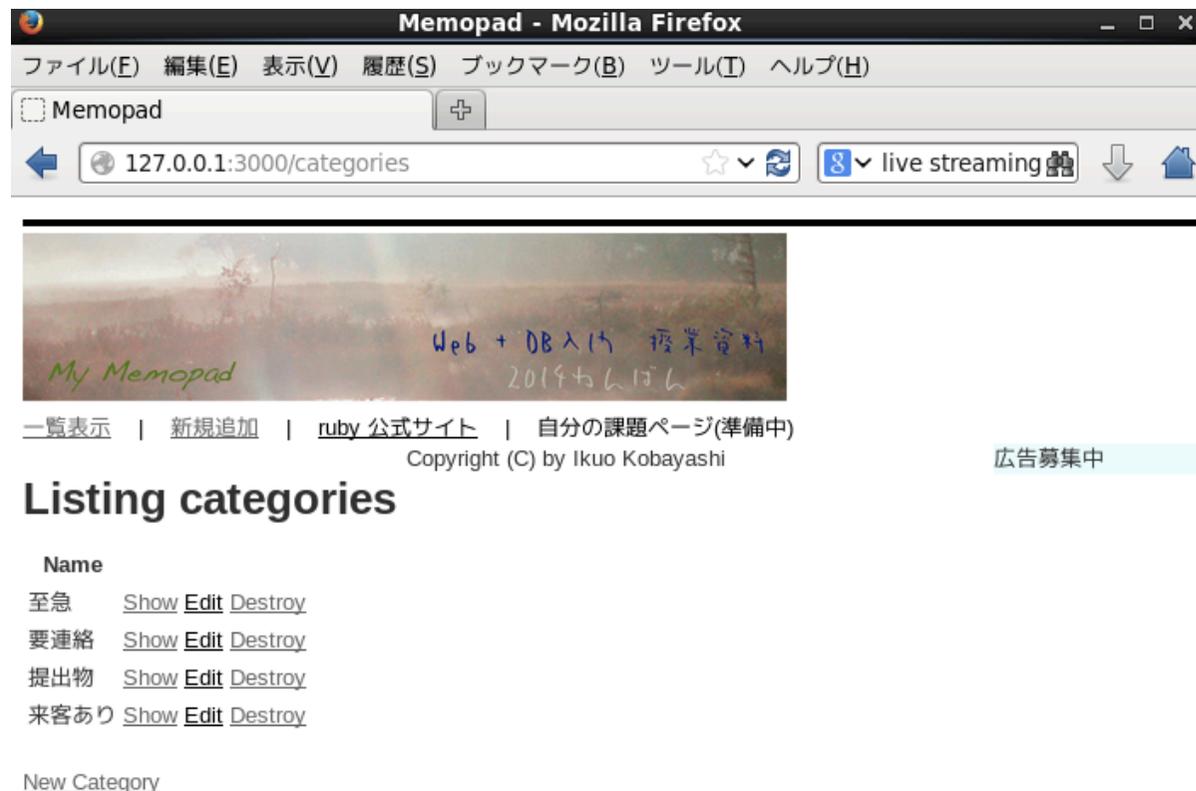
- レポート提出の対象ではありませんが・・・
- 今回の演習は「メモ」に、「分類」をくりつける準備作業でした。
- 分類ばかりではなく、「相手」(「誰と共有するメモか」)を登録して、メモ帳ではその「相手」ごとの一覧表示などが可能となるための情報を追加するとしたら、どうなるか、そのためのテーブル追加などできれば行ってみてください。
 - (演習の中での意味づけは、授業中説明します。)

レポート課題

- 今回は、課題はありません。

今日の授業を休んだ人は・・・

- 自分の演習環境上で、「分類用」コードの一覧を表示した画面を報告して下さい。



Memopad - Mozilla Firefox

ファイル(E) 編集(E) 表示(V) 履歴(S) ブックマーク(B) ツール(T) ヘルプ(H)

Memopad

127.0.0.1:3000/categories

live streaming

My Memopad

Web + DB入り 授業資料
2019おんげん

一覧表示 | 新規追加 | [ruby 公式サイト](#) | 自分の課題ページ(準備中)

Copyright (C) by Ikuo Kobayashi

広告募集中

Listing categories

Name
至急 Show Edit Destroy
要連絡 Show Edit Destroy
提出物 Show Edit Destroy
来客あり Show Edit Destroy

[New Category](#)