はじめようRuby on Rails

フレームワークで作るWebアプリケーション

かずひこ（日本Rubyの会）・香西利衣（日本Rubyの会）

講師紹介（かずひこ）

- オープンソース・プログラマ
- 「はじめようRuby on Rails」著者
- ウェブアプリケーションを中心に、さまざまなオープンソースソフトウェアの開発に参加

講師紹介（香西利衣）

- 京都女子大学4回生
- Rubyでの楽しいプログラミングに魅せられ、関西でのRuby勉強会に参加

今日のテーマ

- シンプルなソーシャルブックマークを題材に、Ruby on Railsによるウェブアプリケーション作りを学びます

はじめに

- Ruby on Railsの概要
- Ruby on Railsのポリシー
- Ruby on Railsのアーキテクチャ
- 想定する環境

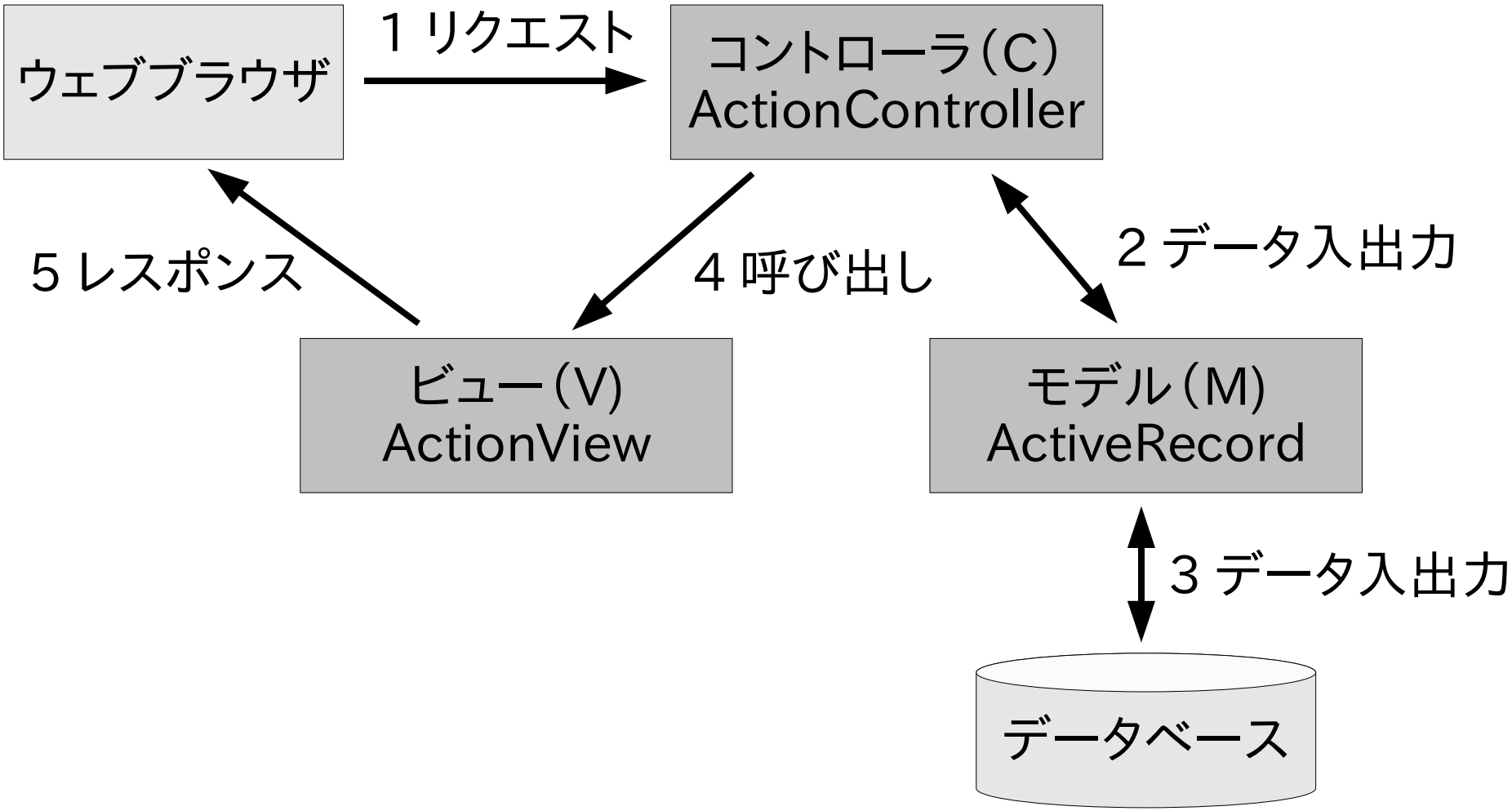
Ruby on Railsの概要

- Rubyで書かれたウェブアプリケーションフレームワークです。
- 自動化プログラムやアプリケーションサーバも含まれます。

Ruby on Railsのポリシー

- DRY (Don't Repeat Yourself)
「重複を避けましょう」
  - 重複する部分を一つにまとめることで、プログラムのメンテナンスがしやすくなります。
- CoC (Convention over Configuration)
「設定より規約」
  - 積極的にデフォルト値を利用することで、プログラムの記述量を減らせます。

Ruby on Railsのアーキテクチャ



モデルの例

```
CREATE TABLE pages (
  id integer,
  uri varchar(1024),
  title varchar(1024)
);

# app/models/page.rb
class Page < ActiveRecord::Base
end
```

モデルの例

```
page = Page.new
page.uri = ”http://jp.rubyist.net/”
page.title = ”日本Rubyの会”
page.save
```

コントローラの例

```
# app/controllers/page_controller.rb
class PageController < ApplicationController
  def list
    @pages = Page.find(:all)
    # ビューを指定しなければ自動的に
    # page/list.rhtmlが呼び出される
  end
end
```

ビューの例

```
<!-- app/views/page/list.rhtml -->
<ul>
  <% for page in @pages %>
  <li><%= h(page.title) %></li>
  <% end %>
</ul>
```

テキストで解説する環境

- OS
  - Linux
- データベース
  - PostgreSQL
- Ruby
  - バージョン1.8.5
- Ruby on Rails
  - バージョン1.1.6をgemコマンドでインストール
- 開発環境
  - エディタ、コンソール

プレゼンで使用する環境

- OS
  - Windows XP
- データベース
  - MySQL
- Ruby
  - バージョン1.8.5
- Ruby on Rails
  - バージョン1.1.6
- 開発環境
  - RadRails（Rails Platform）

簡単インストール

- Instant Rails (Windows XP)
  - http://instantrails.rubyforge.org/
- Locomotive (Mac OS X)
  - http://locomotive.raaum.org/

開発環境

- RadRails
- emacs-rails
- その他の開発環境

RadRails

- http://www.radrails.org/
- Eclipseベースの統合環境
- 日本語化＋サポート＋αのWindows XP向け製品「Rails Platform」もあります
  - http://railsplatform.jp/

emacs-rails

- http://rubyforge.org/projects/emacs-rails/
- ファイルの切替えや補完などいろいろ便利

その他の開発環境

- エディタとコンソールさえあれば開発できる
- UTF-8に対応するもののほうがよい
  （特にエディタ）

Step1 さあ始めよう

- アプリケーションの雛型の作成
- 文字コードの設定
- アプリケーション・サーバの起動
- アプリケーション・サーバの確認
- リファレンスマニュアル

アプリケーションの雛型の作成（RadRails）

- [File]→[New...]→[Rails]→[Rails Project]を選択して[Next]をクリック
- [Project name]に「bookmark」と入力して[Finish]をクリック

誤植の訂正（p.2）

- 2行目

  app/contrillers

  ↓

  app/controllers

文字コードの設定

- Ajaxと相性がいいUTF-8がお薦め
- データベースの文字コード
- Rubyの文字コード
- ウェブサーバのcharset

アプリケーション・サーバの起動（RadRails）

- 画面下の[Servers]タブを選択
- リスト中の[bookmarkServer]を選択
- 右向きの三角をクリック（停止するときは四角）
- [Status]が「Started」に変ったら起動完了

アプリケーション・サーバの確認（RadRails）

- 画面下の[Servers]タブを選択
- リスト中の[bookmarkServer]を選択
- 地球アイコンをクリック

リファレンスマニュアル

- gemコマンドでRailsをインストールすると、Railsを構成する各パッケージのHTML形式のマニュアルもインストールされます。
- このマニュアルは、gem_serverを起動することで、ブラウザからアクセスすることができます。

```
$ gem_server --help
 (略)
$ gem_server &
 (略)
[2006-11-06 11:46:45] INFO
  WEBrick::HTTPServer#start: pid=14875 port=8808
```

リファレンスマニュアル（補足）

- http://techno.hippy.jp/apidoc/（日本語訳）
- http://api.rails2u.com/（全文検索）
- 上記の二つは開発版用マニュアルなので注意

モデリング

- 「○○さん」が
- 「○○というページ」に
- 「○○というコメントでブックマークする」
- 順にそれぞれ、User、Page、Bookmarkというモデルを割り当てます。

各モデルの持つべき情報

- User
  - ログインID, パスワード
- Page
  - URI, タイトル
- Bookmark
  - ユーザID, ページID, コメント, 作成日時

ユーザとページとブックマークの関係

- あるユーザは複数のブックマークを持つ
- あるページは複数のブックマークを持つ

```
User ------ Bookmark ------ Page
       1:多              多:1
```

ユーザとページとブックマークの関係

- あるユーザは複数のページをブックマークする
- あるページは複数のユーザにブックマークされる

```
User -------- Page
       多:多
```

Step2 モデル作成(1)

最初に「ページ」のモデルを作成します。

- 雛型の作成
- テーブル定義ファイルの編集
- テーブル定義の実行

雛型の作成（RadRails）

- 画面下の[Generators]タブを選択
- ドロップダウンで[model]を選択
- テキストフィールドに「Page」と入力
- [Go]をクリック

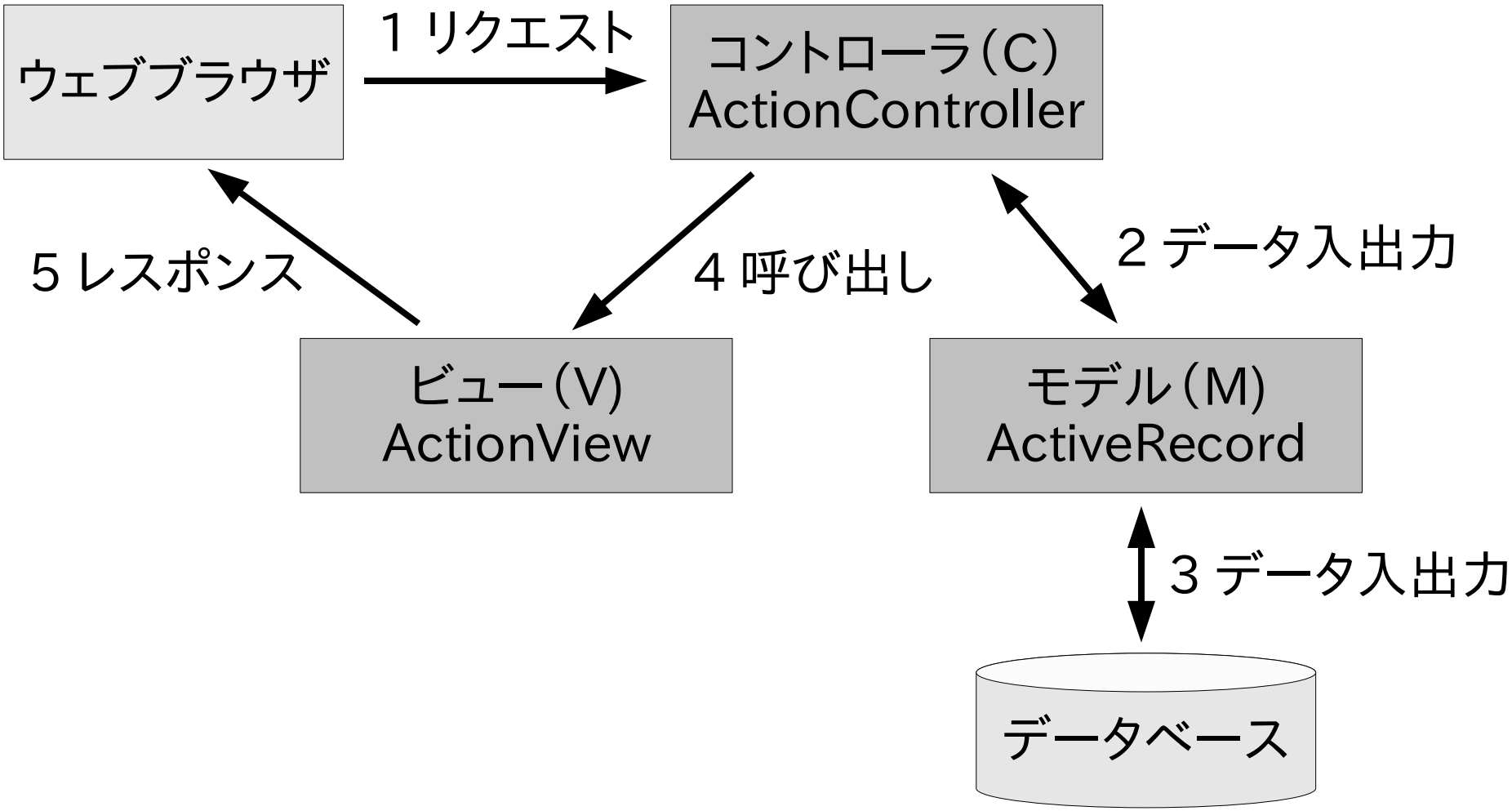
モデルとテーブルの規約

- クラス名はCamelCase
- テーブル名はクラス名の複数形を小文字でアンダーバー区切り
- プライマリキーはidという名のinteger型

テーブル定義の実行（RadRails）

- 画面下の[Rake Tasks]タブを選択
- ドロップダウンで[db:migrate]を選択
- [Go]をクリック

Ruby on Railsのアーキテクチャ



Step3 コントローラ作成（1）

ページ操作のコントローラの雛型を題材に、コントローラの基本を学びます。

- ページ操作の雛型の作成
- ソースを読もう

誤植の訂正（p.6）

- 2行目

  それに続くアクション名を所得し、

  ↓

  それに続くアクション名を取得し、

ページ操作の雛型の作成（RadRails）

- 画面下の[Generators]タブを選択
- ドロップダウンで[scaffold]を選択
- テキストフィールドに「Page Page」と入力
- [Go]をクリック
- 引数にはモデル名とコントローラ名を指定します。

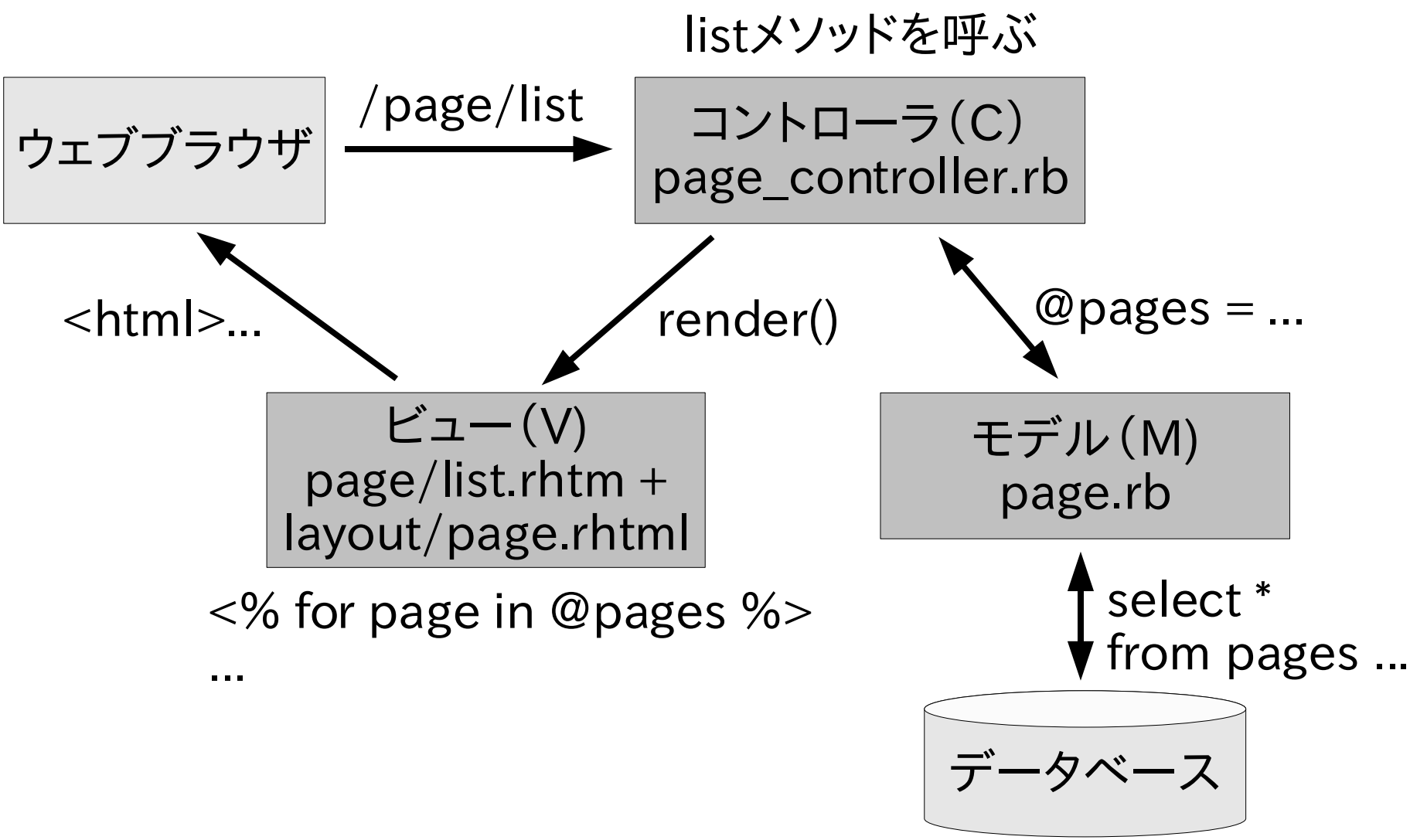
ソースを読もう

- コントローラ
- ビュー
- テスト

コントローラとビューの規約

- /page/index（または/page）にアクセスすると、PageControllerクラス中のindexというメソッドが呼ばれます。
- アクションのメソッドの中でrenderメソッドが呼ばれない場合は、「app/views/コントローラ名/アクション名.rhtml」が呼ばれます。
- レイアウトのファイル名は「app/views/layouts/コントローラ名.rhtml」なければ「app/views/layouts/application.rhtml」が使われます。

アプリケーションの流れ



テストの実行（RadRails）

- 右三角と四角が組み合わさったアイコンをクリックします。
- 終了すると、Test::Unitのタブに切り替わって、結果が表示されます。
- バックトレースの中の行をダブルクリックすると、該当する行がエディタに表示されます。

Step4 モデル修正

- validationの追加
- テスト駆動開発
- アクセサのオーバライド

validationの追加

- モデルのカラムの値が、期待した値かどうか判定する機能がvalidationです。
- モデルのクラス内でvalidateメソッドを定義することで機能します。
- validateメソッドを定義する方法の他に、いくつかのvalidate用メソッドが用意されています。
  - validates_presence_of
  - validates_uniqueness_of
  - validates_format_of
  - など

テスト駆動開発

- テストを先に書き、その後に目的のコードを書くスタイルで開発します。
  - テストを書く→失敗→実装する→成功
    （→リファクタリング→成功）
- テストのコードを見れば、仕様や使い方がよくわかります。
- 何度も同じテストを実行できるので、思わぬエンバグを防ぐことができます。

アクセサのオーバライド

- カラムを取得するメソッドをオーバーライドする際は、元になる値をself.column_nameではなく、self[:column_name]のように取得します。

Step5 モデル作成(2)

- acts_as_authenticatedプラグインのインストール
- generateで自動生成
- その他のプラグイン

プラグインのインストール（RadRails）

- 画面下の[Rails Plugins]タブを選択
- リスト中の[acts_as_authenticated]を選択
- [Go]をクリック

generateの実行（RadRails）

- 画面下の[Generators]タブを選択
- 左のフィールドに「authenticated」と入力
- 右のフィールドに「User Account」と入力
- [Go]をクリック
- 引数にはモデル名とコントローラ名を指定します。

その他のプラグイン

- http://www.agilewebdevelopment.com/plugins にさまざまなプラグインが登録されています。
- http://blog.netswitch.jp/articles/tag/rails_plugin に日本語による紹介があります。

Step6 モデル作成（3）

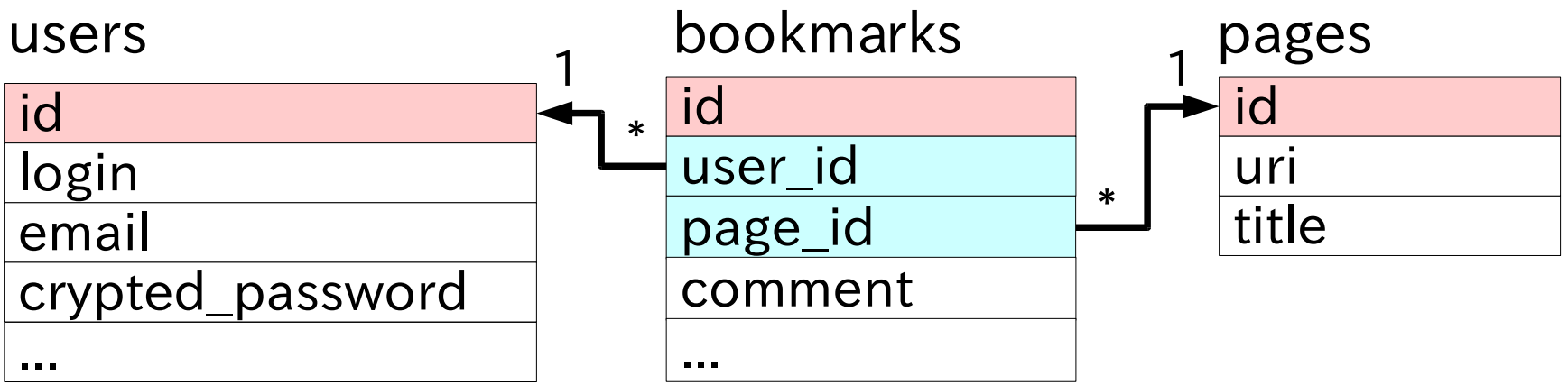
- ブックマークのモデル
- モデル間のリレーション
- テスト

ブックマークのモデル

- テーブル定義

```
create_table :bookmarks do |t|
  t.column :user_id, :integer, :null => false
  t.column :page_id, :integer, :null => false
  t.column :comment, :string, :limit => 1024
  t.column :created_at, :datetime
end
```

モデル間のリレーション



- [規約] リレーション先のプライマリキーを「テーブル名_id」という名前にする

モデル間のリレーション

- リレーションの定義(user.rb)

```
class User < ActiveRecord::Base
  has_many :pages, :through => :bookmarks
  has_many :bookmarks, :order => "created_at desc"
end
```

モデル間のリレーション

- リレーションの定義(page.rb)

```
class Page < ActiveRecord::Base
  has_many :users, :through => :bookmarks
  has_many :bookmarks, :order => "created_at desc"
end
```

モデル間のリレーション - 誤植の訂正(p.13)

- リレーションの定義(bookmark.rb)

```
class Bookmark < ActiveRecord::Base
  belongs_to :user
  belongs_to :page
  validates_uniqueness_of :page_id,
                          :scope => :user_id
end
```

モデル間のリレーション

- 以下のようにリレーションを参照できます。
  - あるユーザがブックマークしている全ページ
    a_user.pages
  - あるユーザの全ブックマーク
    a_user.bookmarks
  - あるページをブックマークしている全ユーザ
    a_page.users
  - あるページの全ブックマーク
    a_page.bookmarks
  - あるブックマークのユーザ
    a_bookmark.user
  - あるブックマークのページ
    a_bookmark.page

Step7 コントローラ作成（2）

- ページのブックマーク一覧の表示
- ユーザのブックマーク一覧の表示
- ブックマークの追加
- 関連するモデルの保存
- トランザクション

コントローラの雛型の作成（RadRails）

- 画面下の[Generators]タブを選択
- ドロップダウンで[controller]を選択
- テキストフィールドに「User」と入力
- [Go]をクリック
- 引数にはコントローラ名を指定します。

ブックマークの追加

- あるページへのブックマークの追加の処理
  - すでにデータベースに存在するページへのブックマーク
    → ブックマークのみ作成
  - データベースに存在しないページへのブックマーク
    → ページとブックマークを同時に作成
- つまり、Pageモデルのオブジェクトの管理は、Bookmarkモデルのオブジェクトを介して行います。

関連するモデルの保存

- 関連するモデルが新規オブジェクトの場合、同時に保存されます。

```
$ ruby script/console
>> b = Bookmark.new
>> b.page = Page.new(:uri =>
                ”http://notexisting.example.com/”)
>> b.user = User.find(1)
>> b.save
```

トランザクション

- 関連するオブジェクトを保存する際は、自動的にトランザクションにラップされます（今回はこちら）。
- そうでない場合はActiveRecord::Base#transactionメソッドで明示的にトランザクションを使います。

Step8 コントローラ作成（3）

- 不要なアクションメソッドとビューの削除
- 人気順ページ一覧の表示

不要なアクションメソッドとビューの削除

- テストのメソッド、コントローラのメソッド、ビューのファイルをそれぞれ削除します。
- 削除した後で、テストを実行します。

findのオプション - 誤植の訂正(p.20)

- SQLで書くと...

```
SELECT page_id, count(user_id) AS count
  FROM bookmarks
  GROUP BY page_id
  ORDER BY count DESC
```

- ActiveRecordのfindメソッドで書くと...

```
Bookmark.find(:all,
              :select => "page_id, count(user_id) as
                          count",
              :group => "page_id",
              :order => "count desc")
```

Step9 パフォーマンスの改善

- インデックスの追加
- 関連するオブジェクトを同時に取得する

インデックスの追加

- findする際の検索条件や並び替えによく使われるカラムに、インデックスを追加します。
  - pages:uri
  - users:login
  - bookmarks:user_id, page_id, created_at
- インデックスの追加も、migration機能でできます。

migrationの雛型の作成（RadRails）

- 画面下の[Generators]タブを選択
- ドロップダウンで[migration]を選択
- テキストフィールドに「AddIndex」と入力
- [Go]をクリック
- 引数はmigrationスクリプトのクラス名になるので、モデルのクラス名などとかぶらないような名前を指定します。
- スクリプトの編集後、Rakeの「db:migrate」タスクを実行します。

関連するオブジェクトを同時に取得する

- 通常、関連するオブジェクトのロードは遅延されます。
- ActiveRecordのfindメソッドの:includeオプションを使えば、関連するオブジェクトを同時に取得することができますので、SQLの発行回数を減らせます。
- app/views/user/show.rhtml（一部）

```
<% for bookmark in @user.bookmarks -%>
   ↓
<% for bookmark in @user.bookmarks.find(:all,
                        :include => [:page]) -%>
```

Step10 URIのルーティングの変更

- ルーティングの概要
- トップページを人気順ページ一覧に
- ブックマーク追加のURIを短くする
- ユーザのブックマーク一覧のURIを、ID番号ではなくログイン名にする
- ページのブックマーク一覧のURIを、ID番号ではなくページのURIにする

ルーティングの概要

- 規約では、以下のようなURIにルーティングされています。

  /コントローラ名/アクション名

  /コントローラ名/アクション名/idの値

- config/routes.rbを変更することで、URIのルーティングを変更することができます。

トップページを人気順ページ一覧に

- トップページの静的ファイルの削除
- ルーティングの変更

```
map.connect "", :controller => "page",
              :action => "top"
```

ブックマーク追加のURIを短くする

- ブックマーク追加のURIを、/bookmark/addから、はてなブックマークのように/addに変更します。
- ルーティングの変更

```
map.connect ”add”, :controller => ”bookmark”,
                   :action => ”add”
```

URIを、ID番号ではなくログイン名にする

- ユーザのブックマーク一覧を表示するURIを、/user/quentinのように変更します。
- テストの変更
- ルーティングの変更

```
map.connect ”user/:login”, :controller => ”user”,
                :action => ”show”
```

- コントローラとビューの変更

URIを、ID番号ではなくページのURIにする

- ページのブックマーク一覧のURIを、はてなブックマークのように/entry/http://example.comのように変更します。
- テストの変更
- ルーティングの変更

```
map.connect ”entry/*uri”, :controller => ”page”,
                :action => ”show” # ←追加
```

- コントローラとビューの変更
  - request.pathでパス全体を取得してから、必要な部分を取り出して、「%xx」という表記をデコードします。

Step11 その他の変更

- AccountControllerのリダイレクト先の変更
- 共通レイアウトの作成
- Javascriptによるブックマークレット

共通レイアウトの作成

- 規約では、app/views/layout/コントローラ名.rhtmlがあればそれを使い、「なければ」app/views/layout/application.rhtmlが使われます。

Javascriptによるブックマークレット

- 外部から、http://localhost:3000/add?title=this_is_title;uri=http://example.com/page.rhtmlのようなURIにアクセスすると、あらかじめパラメータに値をセットした状態でブックマークの追加画面に行くことができます。

Javascriptによるブックマークレット

- ページの移動
  - window.location='http://...'
- titleの取得
  - encodeURIComponent(document.title)
- uriの取得
  - encodeURIComponent(location.href)

Javascriptによるブックマークレット

- ブックマークレットのリンクは以下になります（一行につなげてください）。

```
javascript:window.location='http://localhost:3000/add?title='+encodeURIComponent(document.title)+';uri='+encodeURIComponent(location.href);
```

セキュリティ

- XSS
- CSRF
- SQLインジェクション
- パラメータの改竄

XSS

- 動的にページを生成するシステムを利用し、サイト間を横断して悪意のあるスクリプトが混入される事をXSS（Cross Site Scripting）といいます。
- 外部からの入力値をもとに動的にHTMLを出力する際に、適切にエスケープ処理をする必要があります。
  - hまたはhtml_escape:HTML上特別な文字（<, >, &, "）をエスケープします。
  - uまたはurl_encode:URIに使えない文字をエスケープします。

CSRF

- 外部のページからのHTTPリクエストを受け付けるよう仕向け、ユーザーの意図しない操作をWebアプリケーション上で行わせる事をCSRF（Cross Site Request Forgeries）といいます。
- リクエストの偽造を難しくする対策を施します。
  - security_extensionsというプラグインが便利です

SQLインジェクション

- SQL文を含む引数を持つHTTPリクエストを使って、データーベースシステムを不正に操作することをSQLインジェクションと言います。
- ActiveRecordが提供するプレースフォルダの機能を使います。
- find_by_nameのようなメソッドでは、内部で自動的にエスケープされます。

パラメータの改竄

- `@page = Page.new(params[:page])`のようなコードだと、本来フォームから入力されないはずのフィールドまで設定されてしまう可能性があります。
- ActiveRecord::Base.attr_protectedメソッドで、保護したいフィールドを指定します。
- ActiveRecord::Base.with_scopeメソッドで、あらかじて制限をかけます。

これからの学習に

- http://www.fdiary.net/dev/sns/にて「Rubyist SNS」開発中
- 今回紹介したコードをもとに、ソーシャルブックマーク機能を実装予定
- その他さまざまな機能をできるだけシンプルに実装します

おしまい

- http://kazuhiko.tdiary.net/tmp/iw2006.pdf でこのスライドを配布します。
- ご来場ありがとうございました。