15. Light Weight Language に関する知識 II

1. 科目の概要

Light Weight Language によるシステム構築の応用例として Ruby によるアプリケーション構築フレームワークである Ruby on Rails (RoR)を取り上げる。RoR の基本的な仕組みを解説し、RoR を利用したデータベースアプリケーション開発やアプリケーションのカスタマイズ、開発方法などについて解説する。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説 明	シラバスの対応コマ
II-15-1. Ruby on Rails (RoR)の仕組み	RubyによるWebアプリケーション開発フレームワークであるRuby on Rails (RoR)の基本的な概念、構成、仕組みについて解説し、導入と設定方法、簡単な使い方を紹介する。またコードの自動生成や開発方法を述べ、RoRを利用したアプリケーション開発の第一歩に関して説明する。	11
II-15-2. RoRにおけるMVCアーキテクチャ	RoRを用いたWebアプリケーション開発を行う際に基本となるMVCアーキテクチャについて触れ、RoRではMVCアーキテクチャがどのように実現されるか、その構成要素を解説する。	11
II-15-3. Railsにおけるデータベース連携	RoRを用いたWebアプリケーション開発ではどのようにデータベースとの連携が行われるかを解説する。RoRにおけるデータベース連携のコンポーネントであるActiveRecordの概要と構成について説明し、データベース操作に必要な各種の設定について述べる。	12
II-15-4. RoRを利用したデータベースアプリケーション開発	データベースアプリケーションの開発方法を、RoRでどのように実現するかについて具体的な操作手順を交えて説明する。テーブルの定義、テーブル操作、トランザクション処理といったそれぞれの手順がRoRでどう実現されるかを解説する。	12
II-15-5. RoRを利用したWebアプリケーション開発	RoRによるWebアプリケーション開発の概要と作業の流れを解説する。RoRを利用して開発したWebアプリケーションの構成要素を説明し、データベースの設定方法やWebサーバの起動、インタラクションの確認方法などについて解説する。	13
II-15-6. Webアプリケーション開発のカスタマイズ	ルーティングやページ遷移、表示部分のカスタマイズ、RHTMLの利用、リダイレクト処理など、RoRによるWebアプリケーションをカスタマイズする典型的な手法を紹介する。また、他のOSSによるWebアプリケーションについてカスタマイズの事例を紹介する。	13,15
II-15-7. RoRの機能拡張	RoRの機能を拡張するプラグインについて解説する。プラグインとは何か、プラグインの導入と利用方法について述べ、既存のプラグインとしてどのようなものがあるか、どのようにプラグインを利用すると効果的かといった話題について述べる。さらにサンプルプログラムでプラグインの利用例を示す。	14
II-15-8. RoRのプラグイン開発	RoRの機能を拡張するプラグインを新たに開発する方法を説明する。ジェネレータの実行や機能の実装と実行といったプラグイン作成手順の流れを示し、作成したプラグインをテストする手法を示す。また新たに作成したプラグインを利用するサンプルプログラムを示し実際の利用手順を説明する。	14
II-15-9. RoR応用アプリケーションの利用	典型的なWebアプリケーションであるコンテンツ管理システム(Content Management System; CMS)について説明し、CMSのRubyによる実装例としてRubricksやRadiantなどのアプリケーションを紹介する。RubricksやRadiantの構成と特徴について述べ、さらに各アプリケーションの導入と設定方法を解説する。	15
II-15-10. Rubricksのカスタマイズ	Rubricksは様々なコンポーネントで拡張、カスタマイズすることができる。各種コンポーネントの種類と導入方法を示し、Ruburicksを使いやすいようにカスタマイズする手順について説明する。	15

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「15. Light Weight Language に関する知識Ⅱ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）										応用レベル（Ⅱ）				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15. Light Weight Languageに関するスキル	<LightWeight Languageの基本>	<Perlの基本構造>	<PHPの基本構造>	<Pythonの基本構造>	<Rubyの基本構造>	<オブジェクト指向プログラミング>	<組み込みクラス[データ構造]>	<組み込みクラス[データ操作]>	<組み込みクラス[ファイル管理]>	<GUIアプリケーション開発>	<Ruby on Rails>	<データベースアプリケーション開発>	<Webアプリケーション開発>	<プラグイン導入と開発>	<オープンソースシステムのカスタマイズ>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_15.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング [15-6]	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと変換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法 [15-1, 2, 3, 4, 5]	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要 [15-1]						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計 [15-6]	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトウェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング [15-2]	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャ	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CA00. 歴史と概要	CE-CA01. コンピュータアーキテクチャの基礎	CE-CA02. メモリシステムの構成とアーキテクチャ	CE-CA03. インタフェースと通信	CE-CA04. デバイスサブシステム	CE-CA05. CPUアーキテクチャ	CE-CA06. 性能・コスト評価	CE-CA07. 分散・並列処理	CE-CA08. コンピュータによる計算	CE-CA09. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、Ruby on Rails(RoR)に関する知識が挙げられる。RoR の仕組み、開発手順、プラグイン開発、RoR アプリケーションの仕組みを Linux 上での作業を通して習得する。

科目名	第11回	第12回	第13回	第14回	第15回
15.Light Weight Language に関する知識Ⅱ	(1) Ruby on Rails の仕組み (2) コードジェネレートの説明 (3) MVC アーキテクチャ	(1) データベースの仕組み (2) ActiveRecord の説明 (3) データベースアプリケーション開発	(1) Rails によるWeb アプリケーション開発の説明 (2) Web アプリケーションのカスタマイズ	(1) プラグインとは (2) 新規プラグインの開発	(1) Rubricks とは (2) 新規コンポーネントの開発 (3) オープンソースシステムのカスタマイ

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-1. Ruby on Rails(RoR)の仕組み	
対応する コースウェア	第 11 回 (Ruby on Rails)	

II -15-1. Ruby on Rails (RoR)の仕組み

Ruby による Web アプリケーション開発フレームワークである Ruby on Rails (RoR)の基本的な概念、構成、仕組みについて解説し、導入と設定方法、簡単な使い方を紹介する。またコードの自動生成や開発方法を述べ、RoR を利用したアプリケーション開発の第一歩に関して説明する。

【学習の要点】

* Ruby on Rails(RoR)は Ruby 言語による Web アプリケーション開発フレームワークである。
* RoR では、「設定より規約」「Don't Repeat Yourself」の方針がフレームワーク全体に適用されており、開発に要する作業量を大幅に減らすことが可能である。
* RoR のインストール方法には、各 LinuxOS に付属するパッケージマネージャを用いる方法や Ruby 独自のパッケージ管理システムである RubyGems を用いる方法などがある。
* RoR ではアプリケーションに必要なファイルの雛形を自動生成する仕組みが充実しており、開発の効率化を図ることができる。

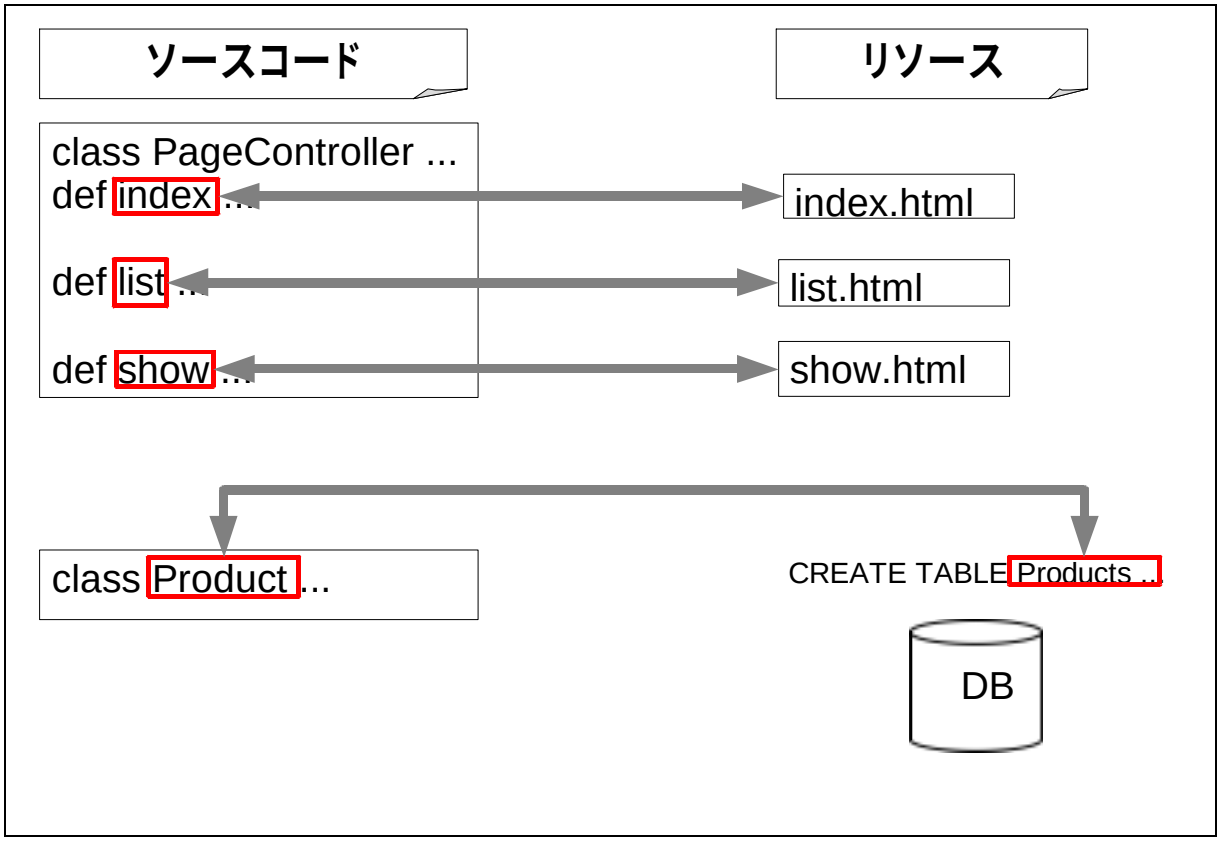

図 II-15-1.　RoR における規約による関連付け

【解説】

1) Ruby on Rails(RoR)

Ruby on Rails(RoR)は Ruby 言語による Web アプリケーション開発フレームワークであり、素早い開発を可能にするフレームワークとして人気を得ている。RoR の設計は「設定より規約:CoC（Convention over Configration）」と「DRY(Don't Repeat Yourself)」の哲学に基づいている。

* 「設定より規約」は、従来のフレームワークでは設定ファイルに記述していた情報を、RoR 側が規約にしたがって自動的に解決することにより、設定ファイル記述の大部分を不要とする考え方である。開発者が RoR の規約に従ったクラス名、変数名、ファイル名などを用いることで開発スピードが大幅に向上する。
* 「DRY」はアプリケーションに関するソースコード、設定などあらゆる情報は、一ヶ所に一度だけ記述する、という考え方であり、作業量が少なくなると共に、保守性が向上する。

2) RoR のインストール

* RubyGems を利用したインストール

  最も基本的な方法である。Ruby のインストール、RubyGems のインストール、RubyGems による RoR のインストール、の手順で行う。
* apt-get を利用したインストール
  debian 系の Linux ディストリビューションでは rails パッケージが用意されており、apt-get コマンドによるインストールが可能である。
* One-Step インストール

  RubyStack などの One-Step インストーラを用いることで、Ruby、RoR、Apache HTTP Server、MySQL、Mongrel などを、事前に設定された状態で一挙にインストールすることができる。

3) RoR の構成

rails コマンドにより、アプリケーションを作成できる。アプリケーションの初期構造は以下のとおり。ソースコードファイルや設定ファイルなどを配置する場所が規約により指定されている。

* README - インストール手順と使い方
* Rakefile - ビルドスクリプト
* app/ - モデル、ビュー、コントローラのファイル
* components/ - 再利用可能なコンポーネント
* config/ - 各種設定ファイル
* db/ - データベーススキーマとマイグレーションの情報
* doc/ - 自動生成されるドキュメント
* lib/ - ライブラリ
* log/ - アプリケーションのログファイル
* public/ - Web からアクセス可能なディレクトリ
* script/ - ユーティリティスクリプト
* test/ - テストプログラム
* tmp/ - 一時的なファイル
* vendor/ - 外部のコード

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-2. RoR における MVC アーキテクチャ	
対応する コースウェア	第 11 回 (Ruby on Rails)	

II -15-2. Ruby on Rails (RoR)の仕組み

RoR を用いた Web アプリケーション開発を行う際に基本となる MVC アーキテクチャについて触れ、RoR では MVC アーキテクチャがどのように実現されるか、その構成要素を解説する。

【学習の要点】

* RoR では MVC アーキテクチャが採用されている。
* RoR における Controller は、ApplicationController クラス継承した Ruby クラスとして記述する。
* RoR における View はテンプレートと呼ばれる。テンプレートを記述する際に、各種テンプレートエンジンを利用することで、動的に画面を生成できる。
* RoR における Model は、ActiveRecord::Base クラスを継承した Ruby クラスとして記述する。Active Record は O/R マッピング機能を提供し、開発者がデータベースを直接操作する必要はない。

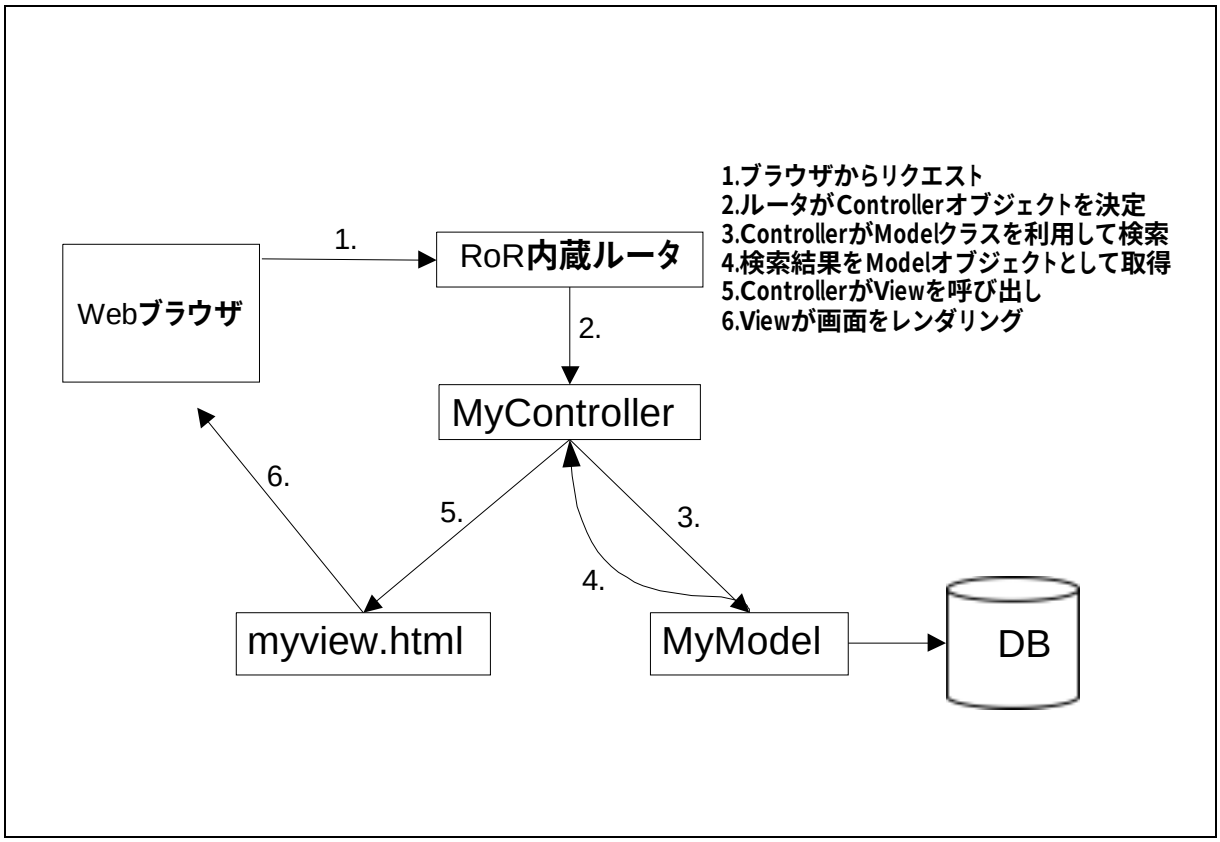

図 II-15-2.　RoR における MVC

【解説】

1) RoR における MVC の配置

RoR では MVC アーキテクチャを採用している。

* app/controllers/
* app/models/
* app/views/

に対応するモジュールが置かれる。

2) RoR における MVC の役割

* Model
  - 通常の MVC では、Model はデータとビジネスロジックをカプセル化したものである。データベースとの関連は実装に任せられている。
  - RoR では Model を ActiveRecord::Base クラスを継承した Ruby クラスとして記述する。Model は ActiveRecord の機能ににより自動的にデータベーステーブルと関連付けられる。
  - RoR においても、ビジネスロジックは通常の MVC と同様に Model に記述することが一般的である。異なる意見として、RoR では Controller が 3 層アーキテクチャにおけるビジネスロジック層、Model がデータアクセス層にあたる、という考え方があり、この場合、ビジネスロジックは Controller に記述する。
* View
  - View はユーザインタフェースを提供する。RoR でも View の役割は通常の MVC と同様である。
  - RoR における View はテンプレートと呼ばれる。テンプレートに対し、builder、ERb、rjs、haml といったテンプレートエンジンを適用することで、クライアントへのレスポンスを動的に生成することができる。
  - RoR では View と Controller の結びつきが強い。そのため、RoR における View と Controller は Action Pack という単一のコンポーネントにまとめられている。
  - RoR では、View にプログラム要素が入り過ぎるのを防ぐために、ヘルパーを利用することができる。ヘルパーとは、表示の整形など、View 用のプログラム要素を、View から簡単に呼び出せる Ruby メソッドとして記述したものである。
* Controller
  - 通常の MVC では、Controller はリクエストの内容を解析し、適した Model、View へ処理を委譲する。
  - RoR における Controller は ApplicationController クラス継承した Ruby クラスとして記述する。
  - 一つのアプリケーション内に複数の Controller が存在することが一般的である。
  - リクエストはまず RoR 内蔵のルータによって受け付けられる。ルータは、開発者が記述した Controller クラスのオブジェクトの内、リクエストに対応するオブジェクトのメソッドを呼び出す。
  - RoR のルータと Controller の役割を合わせたものが、通常の MVC における Controller であると言える。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-3. Rails におけるデータベース連携	
対応するコースウェア	第 12 回 (データベースアプリケーション開発)	

II -15-3. Rails におけるデータベース連携

RoR を用いた Web アプリケーション開発ではどのようにデータベースとの連携が行われるかを解説する。RoR におけるデータベース連携のコンポーネントである ActiveRecord の概要と構成について説明し、データベース操作に必要な各種の設定について述べる。

【学習の要点】

* Active Record は RoR に用意された O/R マッピング層である。
* Active Record を利用した場合、データベーススキーマの情報は動的に取得されるため、Ruby ソースコード中にデータベーススキーマの情報を記述する必要はない。
* RoR ではアプリケーションのデータベーススキーマの変更を管理するために、マイグレーション機能を利用することができる。
* データベースの接続に関する設定は config/database.yml で行う。

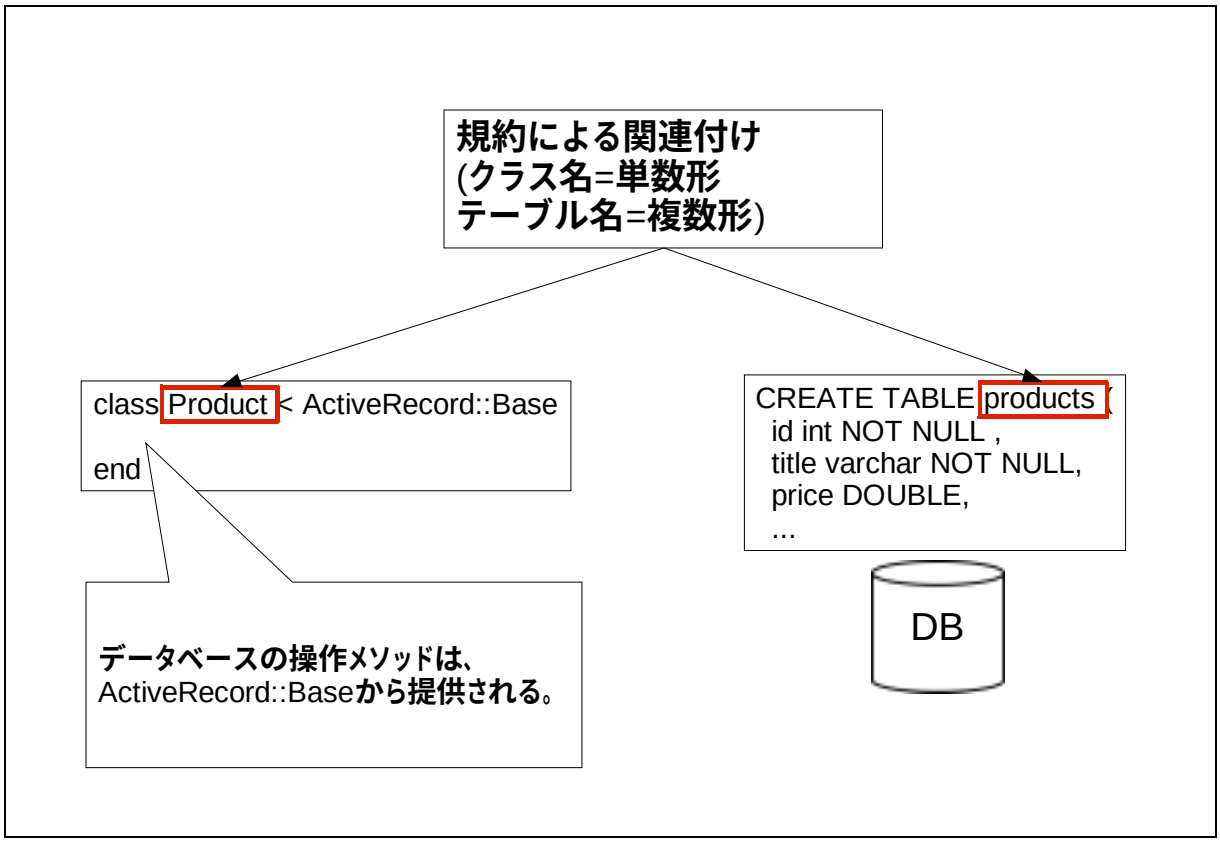

図 II-15-3.　ActiveRecord による O/R マッピング

【解説】

1) ActiveRecord

Active Record は RoR に用意された O/R マッピングの層である。通常の Ruby プログラム内で利用することも可能である。

2) ActiveRecord による O/R マッピング

* ActiveRecored::Base クラスを継承したクラスは、クラス名を元にデータベーステーブルと結び付けられる。
* データベースの行は、Model クラスのオブジェクトにマッピングされる。
* データベースの列は、Model クラスのオブジェクトの属性にマッピングされる。このとき、SQL の型は Ruby のクラスに変換される。
* データベーステーブルの列に関する情報は、ActiveRecord によって動的に取得されるため、ソースコードや設定ファイル内に記述する必要はない。
* ActiveRecord は Model オブジェクトに対し、値の検証を行う機能を提供する。定型的な検証については、あらかじめ用意されているヘルパーメソッドを利用することができる。
* ActiveRecord は、1 対 1、1 対多、多対多のテーブル間の関連をサポートする。

3) データベースへの接続

* ActiveRecord では、データベース操作は共通のインタフェースによって抽象化されている。
* データベース接続処理の詳細は、データベース毎の固有のアダプタによって行われる。
* 各種データベースへの接続に用いるアダプタの多くは、RubyGems によりインストールすることができる。
* RoR 内で ActiveRecord を利用する場合は、config/database.yml にデータベース接続情報を指定する。
* ActiveRecord を単体で用いる場合、データベース接続は ActiveRecord::Base.establish_connection クラスメソッドによって行う。データベース接続情報はメソッドの引数として渡す。

4) マイグレーション

Web アプリケーションをインクリメンタルに開発する場合、開発が進むにつれてデータベーススキーマが変化していく場合が考えられる。RoR では、このようなデータベーススキーマの変更を管理する仕組みとして、マイグレーションを利用することができる。

* マイグレーションは db/migrate ディレクトリに配置される Ruby コードである。
* マイグレーションにはバージョン番号が割り当てられる。
* マイグレーションは ActiveRecord::Migration のサブクラスであり、データベーススキーマの変更を行うクラスメソッド up と、データベーススキーマの変更を取り消すクラスメソッド down を含む必要がある。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-4. RoR を利用したデータベースアプリケーション開発	
対応するコースウェア	第 12 回 (データベースアプリケーション開発)	

II -15-4. RoR を利用したデータベースアプリケーション開発

データベースアプリケーションの開発方法を、RoR でどのように実現するかについて具体的な操作手順を交えて説明する。テーブルの定義、テーブル操作、トランザクション処理といったそれぞれの手順が RoR でどう実現されるかを解説する。

【学習の要点】

* テーブルの作成、スキーマの変更はマイグレーションに記述する。
* テーブルに対する新たな行の追加は、対応する Model クラスの new メソッドを利用してオブジェクトを作成し、save メソッドで保存することで実現できる。
* テーブルからのデータの取得は、対応する Model クラスの find メソッドで実現できる。取得したデータは Model クラスのオブジェクトとして返却される。
* 既存の行の更新は、取得した Model クラスのオブジェクトの属性を更新し、save メソッドで保存することによって実現できる。
* テーブルの既存の行の削除は、対応する Model クラスのクラスメソッド、もしくはインスタンスメソッドにより実現できる。
* トランザクションは、Model クラスの transaction メソッドにより実現できる。

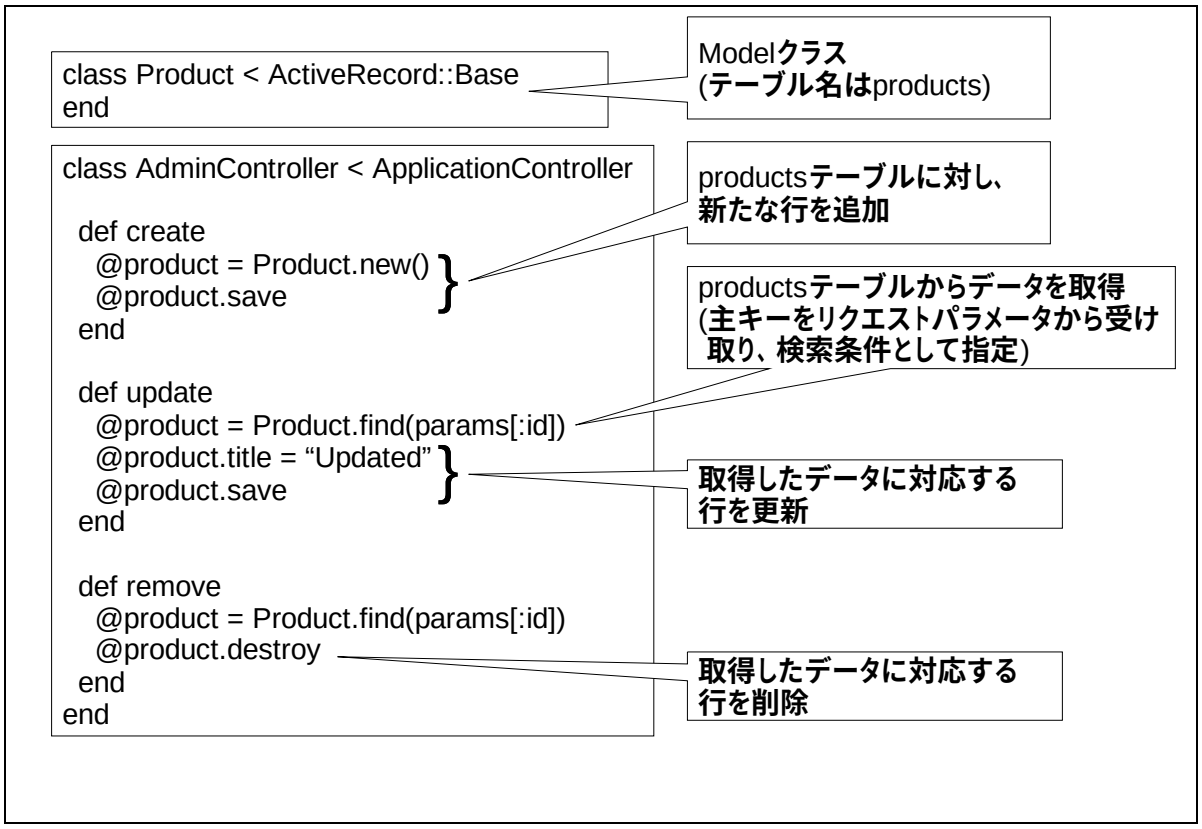

図 II-15-4.　ActiveRecord を利用したデータベース操作

【解説】

1) Model 開発の手順

* config/database.yml ファイルに、データベース接続に関する設定を記述する。database.yml は 3 つのセクションに分かれており、それぞれ開発用データベース、テスト用データベース、本番用データベースの情報を記述する。
* rake タスク「rake db:create」を実行することにより、データベースを作成する。
* スクリプト「ruby script/generate model [Model 名]」を実行することにより、Model クラスとマイグレーションの雛形を生成する。
  - Model クラスの雛形は app/models/ディレクトリに生成される。
  - マイグレーションの雛形は、db/migrate/ディレクトリに生成される。
* マイグレーションにデータベーステーブルの作成処理、および削除処理を記述する。
* rake タスク「rake db:migrate」により、マイグレーションに記述したテーブルの作成処理を実行する。
* アプリケーション仕様に基いて、Model クラスに検証メソッド、およびビジネスロジックを行うメソッドを追加する。

2) ActiveRecord を利用したデータベース操作

* テーブルに対する新たな行の追加
  テーブルに対応する Model クラスの new メソッドを利用してオブジェクトを作成し、save メソッドを呼ぶことでデータベースに新たな行が追加される。また、Model クラスの create メソッドを利用することで、オブジェクトの作成と、データベースへの保存を同時に行うことができる。
* テーブルからのデータの取得
  テーブルに対応する Model クラスの find メソッドによって既存の行データを Model クラスのオブジェクトとして取得することができる。find メソッドには主キーのほか、検索条件やソート条件など、SQL で指定できる検索条件の多くを引数の形で指定できる。
* 既存の行の更新
  取得した Model オブジェクトの属性の値を変更し、save メソッドを呼ぶことで、既存の行を更新できる。また、Model クラスの update、および update_all メソッドを利用することで、行データの取得と更新を一度に行うことができる。
* 既存の行の削除
  テーブルに対応する Model クラスの destroy、および destroy_all メソッドを利用することで、テーブルレベルの行の削除を行うことができる。また、取得した Model オブジェクトの destroy メソッドを利用することで、そのオブジェクトに対応する行を削除することができる。
* トランザクション
  Model クラスの transaction メソッドを利用することで、トランザクション処理を実現できる。トランザクションとして実行される一連の処理はブロックとして渡される。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-5. RoR を利用した Web アプリケーション開発	
対応する コースウェア	第 13 回 (Web アプリケーション開発)	

II -15-5. RoR を利用した Web アプリケーション開発

RoR による Web アプリケーション開発の概要と作業の流れを解説する。RoR を利用して開発した Web アプリケーションの構成要素を説明し、データベースの設定方法や Web サーバの起動、インタラクションの確認方法などについて解説する。

【学習の要点】

* RoR を利用した Web アプリケーション開発は、Model、View、Controller に該当する各要素の作成を通じて行われる。
* RoR では Model、View、Controller の各要素を、scaffold と呼ばれる仕組みにより一挙に自動生成することができ、開発を開始するにあたっての雛形とすることができる。
* scaffold を利用して生成したアプリケーションは、データベースの設定、Web サーバの起動を行うことで即座に動作させることが可能である。

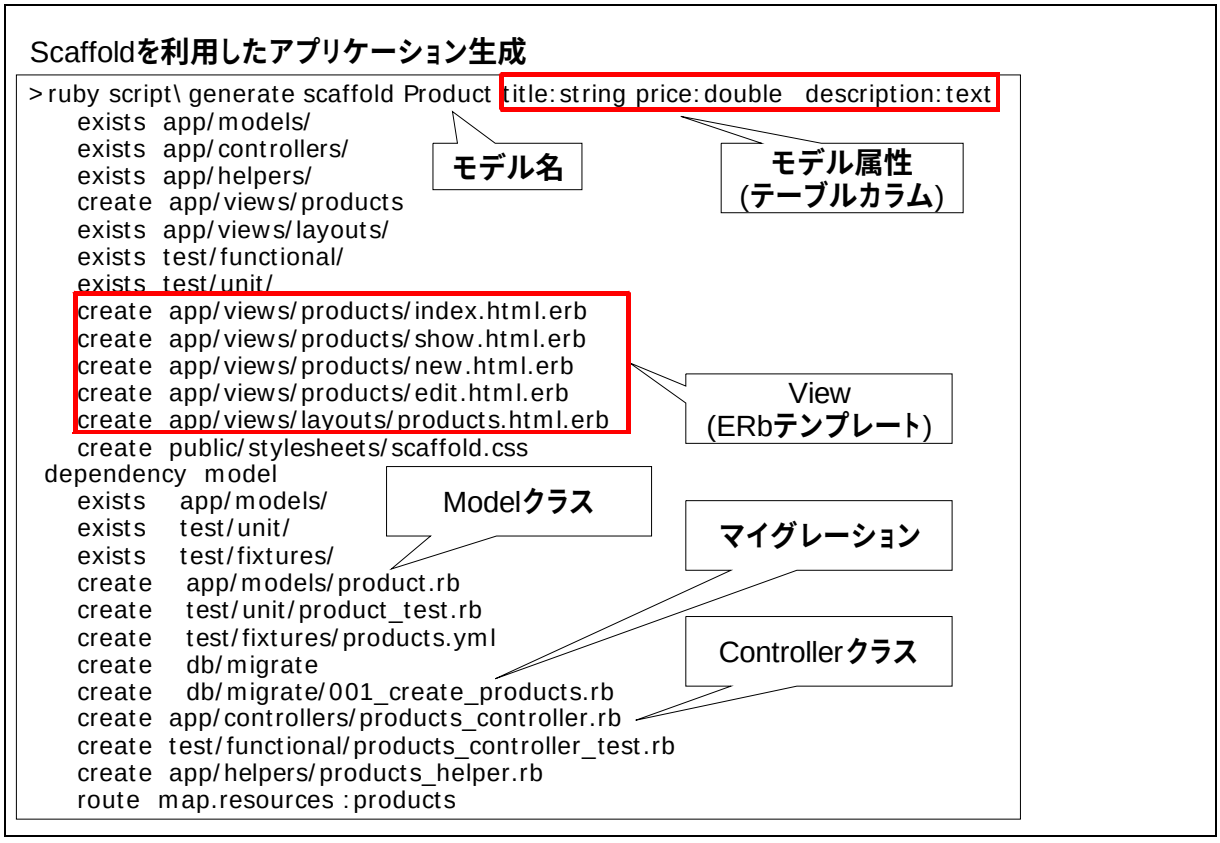

図 II-15-5.　Scaffold を利用したアプリケーションの生成例

【解説】

1) RoR による Web アプリケーション開発

* Model の作成
  「Ⅱ-15-4. RoR を利用したデータベースアプリケーション開発」を参照のこと。
* Controller の作成
  スクリプト「ruby script/generate controller [Controller 名]」を実行することにより、Controller の雛形を生成することができる。生成される要素は以下のとおり。
  - app/controller/[Controller 名]_controller.rb
    Controller の本体となるクラスを記述するソースコードファイル。初期状態では Controller クラスが、メソッドを持たない状態で記述されている。Controller クラスにアクションメソッドを追加することによって Contoroller の開発を行う。
  - test/functional/[Controller 名] _controller_test.rb
    Controller の機能テストを記述するソースコードファイル。
  - app/views/[Controller 名]/
    Controller に対応した View を配置するためのディレクトリ。このディレクトリにテンプレートファイルを置くことで、規約により Controller と View が関連付けられる。
  - app/helpers/[Controller 名]_helper.rb
    Controller、および Controller に関連付けられた View 内から呼び出し可能なヘルパーメソッドを記述するソースコードファイル。
* View の作成
  View はテンプレートによって表現される。動的なレスポンスを実現するテンプレートを記述する際には、テンプレートエンジンとして以下のようなライブラリを利用できる。
  - builder‐ 主に、XML レスポンスの構築に用いられる。
  - erb‐ 主に、動的な HTML ページの構築に用いられる。
  - rjs‐JavaScript の生成に用いられる。

  テンプレートの配置ディレクトリ、およびファイル名は規約によって定められている。
  - テンプレートは、それを呼び出す Controller と規約によって関連付けられる。テンプレートは、「app/views/[関連 Controller 名]/」ディレクトリに配置する。
  - テンプレートのファイル名は「[アクションメソッド名].[フォーマット].[テンプレートエンジン名]」とする。

2) scaffold によるアプリケーション生成

RoR では、scaffold と呼ばれる仕組みにより、簡単なアプリケーションを一挙に作成することができる。scaffold を利用した場合の、アプリケーションの動作を確認するまでの手順は以下のとおり。

* rails コマンドによってアプリケーションのディレクトリ構造を生成する。
* 「ruby script/generate scaffold」スクリプトによってアプリケーションの雛形を生成する。
* config/database.yml ファイルの編集によってデータベース接続設定を行う。
* 「rake db:create」タスクにより、データベースの作成を行う。
* 「rake db:migration」タスクにより、データベースのマイグレーションを行う。
* 「ruby script/server」スクリプトによって Web サーバを起動する。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-6. Web アプリケーション開発のカスタマイズ	
対応する コースウェア	第 13 回 (Web アプリケーション開発) 第 15 回 (オープンソースシステムのカスタマイズ)	

II -15-6. Web アプリケーション開発のカスタマイズ

ルーティングやページ遷移、表示部分のカスタマイズ、リダイレクト処理など、RoR による Web アプリケーションをカスタマイズする典型的な手法を紹介する。

【学習の要点】

* アプリケーションのルーティング(URL のマッピング)の動作は、config/routes.rb ファイルを修正することで変更できる。
* ページの表示内容は、該当するテンプレートファイルを修正することで変更できる。
* リダイレクト処理は、該当する Controller のアクションメソッド中で redirect_to メソッドを呼び出すことで実現できる。

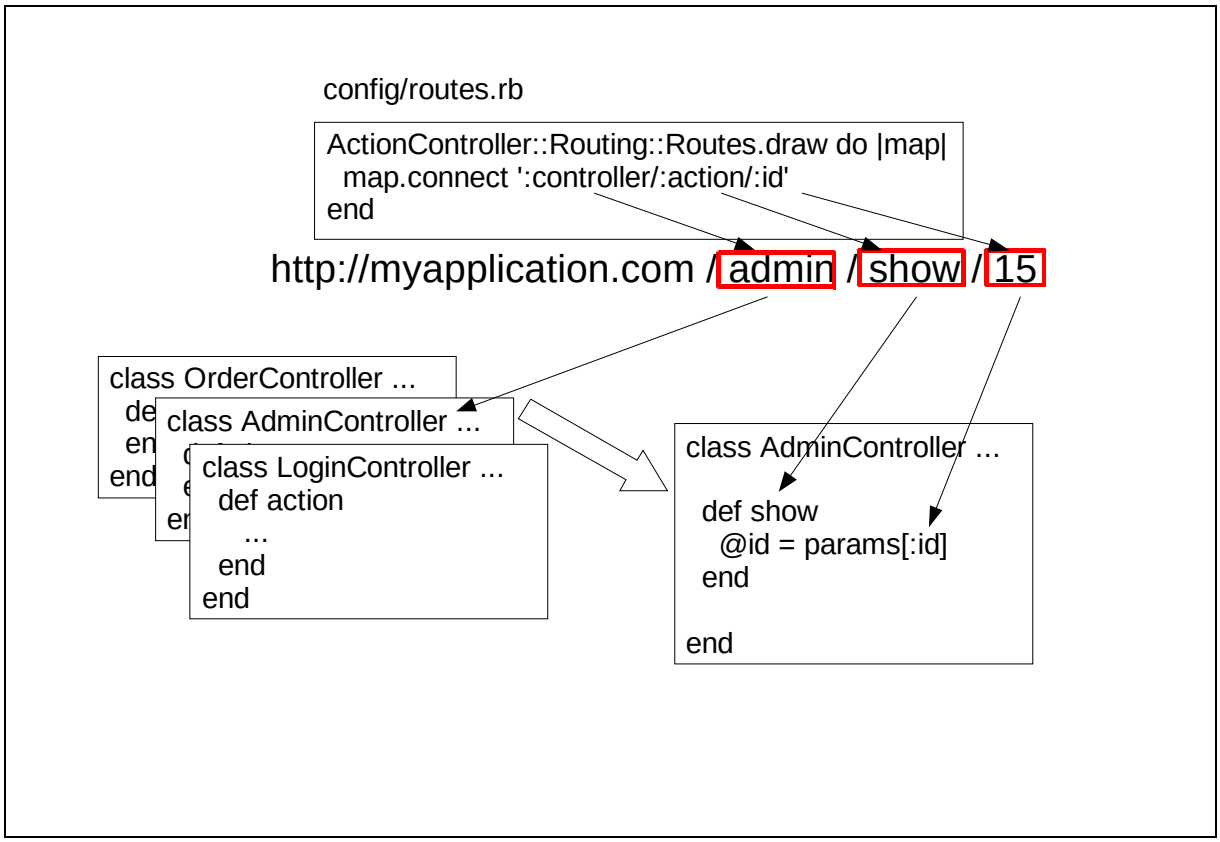

図 II-15-6.　RoR のルーティング

【解説】

1) RoR におけるルーティングの変更

RoR の、URL とメソッドのマッピングの仕組みは、config/routes.rb ファイルを修正することで変更できる。config/routes.rb ファイル内には、ActionController::Routing::Routes.draw メソッドの呼び出しが記述される。draw メソッドの引数として渡すブロックの中で、map オブジェクトのメソッドを呼び出すことで、ルーティングを変更することが可能である。rails コマンドでプロジェクトを作成した場合、config/routes.rb ファイルには次の記述がある。

```
map.connect ':controller/:action/:id'
```

* connect メソッドへの引数は、パターンと追加パラメータに分けられる。上記の例では、パターンのみが渡されている。
* パターンはスラッシュ、またはピリオドでコンポーネントにわけられる。
* 「:[パラメータ名]」の形式のコンポーネントは、リクエストされた URL パスの該当部分の値を[パラメータ名]に対応する値として設定する。
* 上記の例のパターンに対し、「/admin/show /1」という URL パスが与えられた場合、
  - controller => 'admin'
  - action => 'show'
  - id => '1'

  という設定が行われる。
* controller パラメータは Controller オブジェクトの決定に、action パラメータは、アクションメソッドの決定に利用される。その他のパラメータはアクションメソッド内で利用することができる。
* controller パラメータ、および action パラメータは、必ずしもパターン内で指定する必要はなく、追加パラメータとして指定することも可能である。以下の例の場合「foo/bar/baz」というパスに対し、AdminController オブジェクトの show メソッドが呼び出される。

```
map.connect 'foo/bar/baz', :controller => 'admin', :action => 'show'
```

* connect メソッドの呼び出しはブロック内に複数記述することが可能であり、呼び出された順にパターンに対するマッチングが行われる。マッチした時点で、パターンに対応するアクションが実行される。

2) RoR アプリケーションの表示内容の変更

RoR アプリケーションの表示は、テンプレートによって行われる。例として、「ruby script/generate scaffold」スクリプトで生成されるテンプレートの一部を以下にあげる。これらを修正することで、生成したアプリケーションの修正を行うことができる。

* app/views/[コントローラ名]/index.html.erb – 一覧画面のテンプレート
* app/views/[コントローラ名]/edit.html.erb – 編集画面のテンプレート
* app/views/[コントローラ名]/show.html.erb – 詳細画面のテンプレート

3) RoR アプリケーションにおけるリダイレクト

リダイレクト処理は、該当する Controller のアクションメソッド中で redirect_to メソッド呼び出すことにより実現できる。リダイレクト先は引数として指定し、controller、action をパラメータとして指定することができる。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-7. RoR の機能拡張	
対応する コースウェア	第 14 回 (プラグイン導入と開発)	

II -15-7. RoR の機能拡張

RoR の機能を拡張するプラグインについて解説する。プラグインとは何か、プラグインの導入と利用方法について述べ、既存のプラグインとしてどのようなものがあるか、どのようにプラグインを利用すると効果的かといった話題について述べる。さらにサンプルプログラムでプラグインの利用例を示す。

【学習の要点】

* RoR は、プラグインを導入することでフレームワークに存在しない機能を追加することができる。
* プラグインの検索・インストールは script/plugin スクリプトで行うことができる。
* プラグインは vender/plugins ディレクトリに保存される。

プラグインの利用例(calendariffic プラグイン)

(1)プラグインのインストール

```
> ruby script/plugin install http://opensvn.csie.org/calendariffic/calendariffic/ vendor/plugins/calendariffic
```

(2)プラグインの呼び出し(View テンプレート)

```
<h1> calendariffic プラグイン </h1>
<%= javascript_include_tag 'calendariffic/calendar.js',
                           'calendariffic/calendar-setup.js',
                           'calendariffic/lang/calendar-en.js' %>
<%=
  stylesheet_link_tag 'calendariffic/calendar-win2k-cold-1.css'
%>
<%= calendariffic_input(false,
                        'start_date',
                        'calendariffic/date.png',
                        'start_cal',
                        '%m/%d/%y',
                        nil,
                        {:class => 'myfavoriteclass', :readonly => 'true'},
                        {:class => 'borderless'}) %>
```

(3)プラグインの実行(ブラウザ)

図 II-15-7.　プラグインの利用例

【解説】

1) プラグイン

プラグインは、RoR のフレームワークが提供する機能の拡張、もしくは機能の修正を行うためのプログラムである。プラグインを利用することにより、既存のコードを修正すること無く新たな機能を付け加えることができる。また、プラグインの提供者は独自に修正やアップデートをリリースすることができる。

2) プラグインのインストール

プラグインは、script/plugin スクリプトを利用して、Web 上で公開されているリポジトリからインストールすることができる。スクリプトの詳細は以下のコマンドで確認できる。

```
ruby script/plugin -h
```

* プラグインのリスト
  入手可能なプラグインのリストは以下のコマンドにより確認することができる。

```
ruby script/plugin list
```

* リポジトリの追加
  以下のコマンドにより、リポジトリの追加を行うことができる。あらかじめリポジトリを source として登録しておくことで、プラグイン名でのインストールが可能になる。

```
ruby script/plugin source [URL]
```

* プラグインのインストール
  以下のコマンドにより、指定したプラグインをインストールすることができる。リポジトリが source として登録されている場合はプラグイン名、登録されていない場合は URL を指定する。URL として、subversion や git のリポジトリを指定することができる。

```
ruby script/plugin install [プラグイン名 or プラグインの URL]
```

* インストールしたプラグインは vendor/plugins ディレクトリに配置される。

3) プラグインの例

現在公開されているプラグインに関する情報は、Web ページ (http://wiki.rubyonrails.org/rails/pages/Plugins)などから入手できる。公開されているプラグインの例を以下にあげる。

* CSS Graphs(http://nubyonrails.com/pages/css_graphs)
  グラフを作成するためのヘルパーを提供するプラグイン。
* Calendariffic(http://opensvn.csie.org/calendariffic/calendariffic/)
  ポップアップカレンダーによる日付入力インタフェースの実現を容易にするプラグイン。
* TextAreaWithStatus
  (http://text-area-with-status.googlecode.com/svn/tags/text_area_with_status)
  文字数制限のあるテキスト入力欄の作成を容易にするプラグイン。
* Restful Authentication(git://github.com/technoweenie/restful-authentication.git)
  ユーザ登録機能、およびユーザ認証機能の作成を容易にするプラグイン。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-8. RoR のプラグイン開発	
対応する コースウェア	第 14 回 (プラグイン導入と開発)	

II -15-8. RoR のプラグイン開発

RoR の機能を拡張するプラグインを新たに開発する方法を説明する。ジェネレータの実行や機能の実装と実行といったプラグイン作成手順の流れを示し、作成したプラグインをテストする手法を示す。また新たに作成したプラグインを利用するサンプルプログラムを示し実際の利用手順を説明する。

【学習の要点】

* プラグインの雛形はスクリプトにより自動生成することができる。
* プラグインの実装は vender/plugins/[プラグイン名]/lib/[プラグイン名].rb ファイルに記述する。
* プラグインのテストは vender/plugins/[プラグイン名]/test/[プラグイン名]_test.rb ファイルに記述する。
* プラグインのテストは rake test:plugins コマンドによって実行する。

プラグインの開発例(Hello Worldプラグイン)

(1)プラグインの雛形の生成

```
> ruby script/generate plugin hello_world
```

(2)プラグイン本体の記述
(vendor/plugins/hello_world/lib/hello_world.rb)

```
module HelloWorld
  def helloWorld
    '<font size="7">Hello World!</font>'
  end
end
```

(3)プラグインの組み込み処理の記述
(vendor/plugins/hello_world/init.rb)

```
ActionView::Base.send :include, HelloWorld
```

(4)プラグインの呼び出し
(Viewテンプレート)

```
<%= helloWorld %>
```

(5)プラグインの実行(ブラウザ)

Hello World!

図 II-15-8.　プラグインの開発例

【解説】

1) プラグインの種類

RoR のほとんどの機能はプラグインとして開発可能である。以下に拡張例をあげる。

* ActiveRecord::Base クラスに新たなメソッドを追加することで、Model クラスに新たな機能を加える。多くの場合、Model クラス内で「acts_as_」で始まるメソッドを呼び出すことで、機能が有効になるように作成される。
* View テンプレート内で利用可能なヘルパーを追加する。View テンプレート内でメソッドとして呼び出すことで利用することができる。

2) プラグインの開発

プラグインの開発にあたっては、「ruby script/generate plugin [プラグイン名]」スクリプトにより、雛形を生成することができる。生成されるファイルについて以下に説明する。

* vendor/plugins/[プラグイン名]/README
  プラグインの利用者向けに、プラグインのインストール方法、使い方などを記述する。
* vendor/plugins/[プラグイン名]/init.rb
  プラグインの初期化処理を記述する。プラグインのモジュールを RoR に組み込む処理などが該当する。init.rb で一般的に行われる処理を以下にあげる。
  - ActiveRecord::Base.send(:include, [モジュール名])
    Model クラスに対し、モジュール内のメソッドを追加する。
  - ActionController::Base.send(:include, [モジュール名])
    Controller クラスに対し、モジュール内のメソッドを追加する。
  - ActionView::Base.send(:include, [モジュール名])
    モジュール内のメソッドを View テンプレート内で利用可能なヘルパーとして登録する。
* vendor/plugins/[プラグイン名]/install.rb
  プラグインのインストール処理を記述する。JavaScript やスタイルシートを展開する処理などを記述する。
* vendor/plugins/[プラグイン名]/uninstall.rb
  プラグインのアンインストール処理を記述する。
* vendor/plugins/[プラグイン名]/lib/[プラグイン名].rb
  プラグインの本体となるソースコードを記述する。「vendor/plugins/[プラグイン名]/lib/」ディレクトリは、RoR の起動時にロードパスに追加される。記述の仕方としては、プラグイン独自のモジュールを定義してその中にメソッドとして処理を記述し、init.rb 内で RoR に組み込む方法が一般的である。
* vendor/plugins/[プラグイン名]/tasks/[プラグイン名]_tasks.rake
  プラグイン用の rake タスクを記述する。
* vendor/plugins/[プラグイン名]/test/[プラグイン名]_test.rb
  Test::Unit モジュールを利用してプラグイン用のテストを記述する。

3) プラグインのテスト

rake タスク「rake test:plugins」によって、全てのプラグインのテストが実行される。特定のプラグインのみテストする場合は、「rake test:plugins PLUGIN=[プラグイン名]」とする。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-9. RoR 応用アプリケーションの利用	
対応するコースウェア	第 15 回 (オープンソースシステムのカスタマイズ)	

II -15-9. RoR 応用アプリケーションの利用

典型的な Web アプリケーションであるコンテンツ管理システム (Content Management System; CMS) について説明し、CMS の Ruby による実装例として Rubricks や Radiant CMS などのアプリケーションを紹介する。Rubricks や Radiant CMS の構成と特徴について述べ、さらに各アプリケーションの導入と設定方法を解説する。

【学習の要点】

* CMS とは Web ページや画像データといったコンテンツを作成、編集、管理する機能を提供するソフトウェアである。
* RoR を用いて開発された OSS の CMS として Rubricks や Radiant CMS がある。
* Rubricks、Radiant CMS 共に拡張性が重視されており、拡張機能をプラグインとして開発、あるいは導入するための仕組みが提供されている。

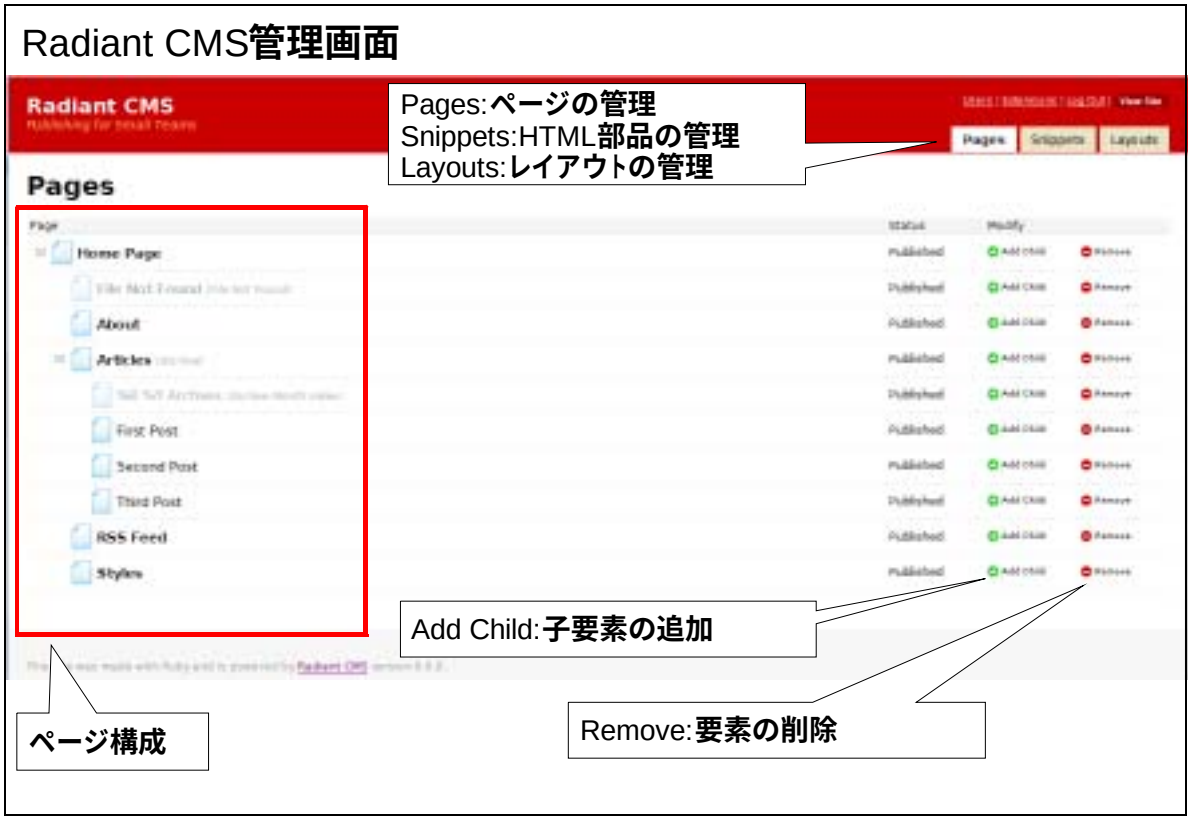

図 II-15-9.　Radiant CMS の管理画面

【解説】

1) CMS(Content Management System)

CMS とは、コンテンツの作成、編集、管理を容易にすると共に、一貫した形式で公開することを可能にするソフトウェアである。Web サイトの構築に広く利用されており、ブラウザ経由で Web ページの追加や編集を行う機能や、ユーザの権限を管理する機能などが提供される。Blog や Wiki も CMS の一種であると考えることができる。

2) Rubricks

Rubricks は日本人の開発グループにより開発されている CMS である。

* RoR 上に構築されており、MIT ライセンスで公開されている。
* BBS やニュース配信の機能が用意されており、コミュニティポータルの作成を容易にする。
* Rubricks はコンポーネントアーキテクチャと呼ばれる機構を採用しており、必要な機能はコンポーネントの単位で管理され、自由に組み合わせることが可能になっている。

3) Rubricks のインストール

* Rubricks の推奨環境は開発 Web ページ (http://dev.rubricks.org/wiki/RubricksSystemRequirementJa)を参照のこと。注意点としては 2008 年 8 月の段階で、RoR のバージョン 1.x 系のみの対応となっている。
* Rubricks のインストール手順については Windows 向け、および CentOS 向けの手順が公開されている。詳細は開発 Web ページ(http://dev.rubricks.org/wiki/RubricksInstallationGuideJa)を参照のこと。

4) Radiant CMS

Radiant CMS はシンプルでわかりやすいインタフェースを特徴とした CMS である。

* Rubricks 同様、RoR 上に構築されており、MIT ライセンスで公開されている。
* Ruby 言語の公式ページ(http://www.ruby-lang.org)で採用されるなど、Web サイト構築用の CMS として比較的多くの実績をもつ。
* Web サイトのツリー構造を柔軟に構築できる機能や、個々のページを統一的なレイアウトで表示する機能など、一般的な Web サイトを構築するための基本的な機能が提供されている。
* Rubricks と同様に拡張機能を管理する機能を持つ。拡張機能は Extension という単位で管理され、様々な機能がサードパーティから提供されている。

5) Radiant CMS のインストール

Radiant CMS は RubyGems パッケージとして提供されており、gem コマンドを利用してインストールすることができる。詳細な手順は Web サイト(http://wiki.radiantcms.org/Installation)を参照のこと。インストールの流れを以下に示す。

* Ruby、および RubyGems のインストール
* gem コマンドによる radiant パッケージのインストール
* Radiant プロジェクトの作成
* データベースの接続設定と初期化
* サーバの起動

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識	応用
習得ポイント	II -15-10. Radiant CMS のカスタマイズ	
対応する コースウェア	第 15 回 (オープンソースシステムのカスタマイズ)	

II -15-10.Radiant CMS のカスタマイズ

Radiant CMS は Extension を導入することで機能を拡張することができる。Radiant CMS の開発プロジェクトやサードパーティから提供されている各種 Extension の種類と導入方法を示し、Rediant をカスタマイズする手順について説明する。

【学習の要点】

* Radiant CMS は導入時、デフォルトで “Archive”、“Markdown”、“Textile”などが Extension として含まれるほか、様々な Extension がサードパーティから提供されており、自由に組み合わせて使用することができる。
* Extension は、Radiant プロジェクトの vendor/extensions/ディレクトリに配置する。

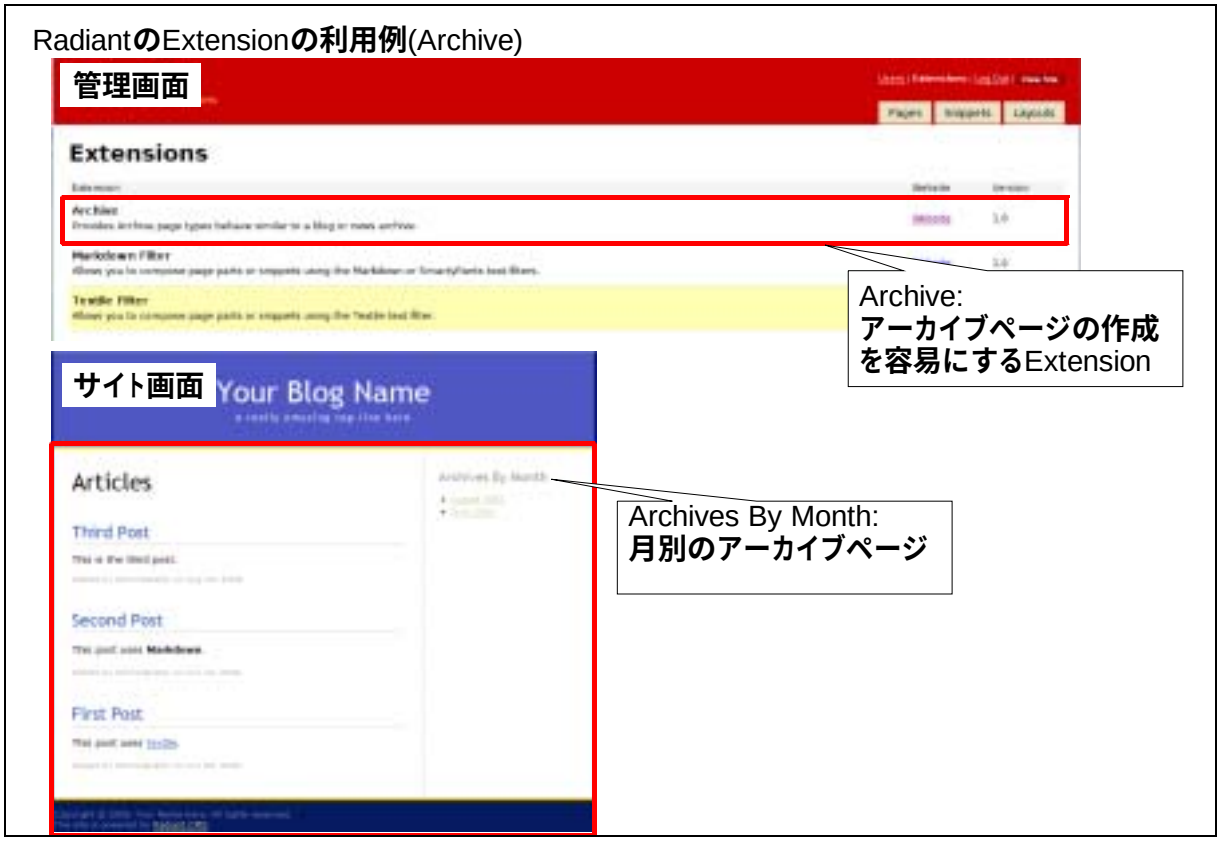

図 II-15-10.　Radiant の Archive Extension

【解説】

1) Extension の種類

Radiant CMS の導入時にデフォルトでインストールされている Extension として、“Archive”、“Markdown”、“Textile”がある。また、サードパーティから公開されている Extension の一覧が Web ページ(http://wiki.radiantcms.org/Thirdparty_Extensions)に掲載されている。利用可能な Extension の一部を以下にあげる。

* Archive
  ニュース記事を扱うサイトや、blog サイトで見られるような、年、月、日毎のアーカイブページを作成することが可能になる。
* Markdown
  HTML の代わりに Markdown 形式によってページを記述することが可能になる。
* Textile
  HTML の代わりに Textile 形式でページを記述することが可能になる。
* Search
  サイト内のページに対するサーチ機能の追加が可能になる。
* FCKeditor
  Javascript による WYSIWYG エディタ「FCKeditor」をページの編集に用いることが可能になる。

2) Extension のインストール

インストール手順の詳細は、Web ページ(http://wiki.radiantcms.org/Extensions)、および Extension ディレクトリ内の README ファイルを参照のこと。一般的な手順を以下に記述する。

* インストールしたい Extension を、リポジトリからのチェックアウトや開発 Web サイトからのダウンロードにより取得する。
* チェックアウト、もしくはダウンロード・解凍してできた Extension ディレクトリを、Radiant プロジェクトの vendor/extensions/ディレクトリの下に配置する。この際、ディレクトリ名を Extension 名にする必要がある。Extension ディレクトリの下には、ファイル名の末尾が「_extension.rb」となっているファイルが含まれており、「_extension.rb」の前の部分が Extension 名である。
* Extension がデータベースを利用する場合、マイグレーションを実行する。
* インストールされている Extension は、Radiant CMS 管理ページの「Extensions」メニューから確認することができる。

3) Extension のアンインストール

インストールの場合と同様、手順の詳細は Radiant CMS の公式 Web ページ、および Extension パッケージ内の README ファイルを参照のこと。一般的な手順を以下に記述する。

* Extension がデータベースを利用する場合、VERSION=0 を指定してマイグレーションを行い、データベースをインストール前の状態に戻す。
* Radiant プロジェクトの vendor/extensions/ディレクトリ以下に配置された Extension ディレクトリの内、アンインストールしたいディレクトリを消去する。