「Rails」を使って分かった業務システム構築ノウハウ

2009.08.01

株式会社 BSN アイネット 渡邊 毅之

目次

1. 自己紹介
2. プロジェクト紹介
3. アドバイス（技術、プロジェクト運用 など）
4. まとめ（全体を通じて感じたこと）

自己紹介

自己紹介

- **株式会社 BSN アイネット　渡邊 毅之**
  - SE 暦　12年
    - Java：7年、Ruby：3年
    - その他いろいろ（UML、デザインパターン、etc）
- Ruby プロジェクト経験
  - 社内ワークフロー開発
  - 全国障害者スポーツ大会 トキめき新潟大会 運用システム
- 好きな言葉
  - なせばなる／身の丈勝負／一歩前進
- モットー
  - 同じことは2度やらない。

Ruby & Ruby on Rails とは

- Ruby
  - **オブジェクト指向スクリプト言語**
- Ruby on Rails
  - Webアプリケーションフレームワーク
  - MVCアーキテクチャ
  - **設計思想**
    - CoC 「**設定よりも規約**」
    - DRY 「**同じことを繰り返さない。重複は避ける。**」

CoC：Convention Over Configuration
DRY：Don't Repeat Yourself

Ruby との出会い

- **きっかけ**
  - **技術部長が旗ふった。**
    「Ruby on Rails **がすごいらしい！**」
- **お試し**
  - **入門本「はじめよう** Ruby on Rails」**を試したら、目からうろこが落ちた。**

画像 Amazon.co.jp **より**
株式会社 ASCII
高橋征義 監修、
かずひこ、喜多川 豪 著
ISBN-13: 978-4756147738
2006.7.18 **初版**

「目からうろこ」が落ちた理由

- Java と比較
  - フレームワーク (Struts)
    - ビュー部品 (JSTL)
    - 入力値検証 (Commons-Validator)
    - データマッピング (Commons-BeanUtils)
  - O/R マッピング (Hibernate)
  - 定型作業の自動化 (Ant, Maven)
  - テスト自動化 (JUnit, DBUnit, Cactus)

- 高い学習効率
  - はじめて Java を学習するなら、最低でも2～3ヶ月。
  - Ruby 自体、初めてなのに2週間くらいで使えるレベル。

本気で感動！

面白い！

プロジェクト紹介

プロジェクト概要

「全国障害者スポーツ大会 トキめき新潟大会」運営システム構築

- **業務機能**
  1. **参加申込**
  2. **競技プログラムの編集**
  3. **競技結果の登録**
  4. **一般公開**
  5. **基本情報 (マスタメンテ)**
  6. **競技運用帳票**

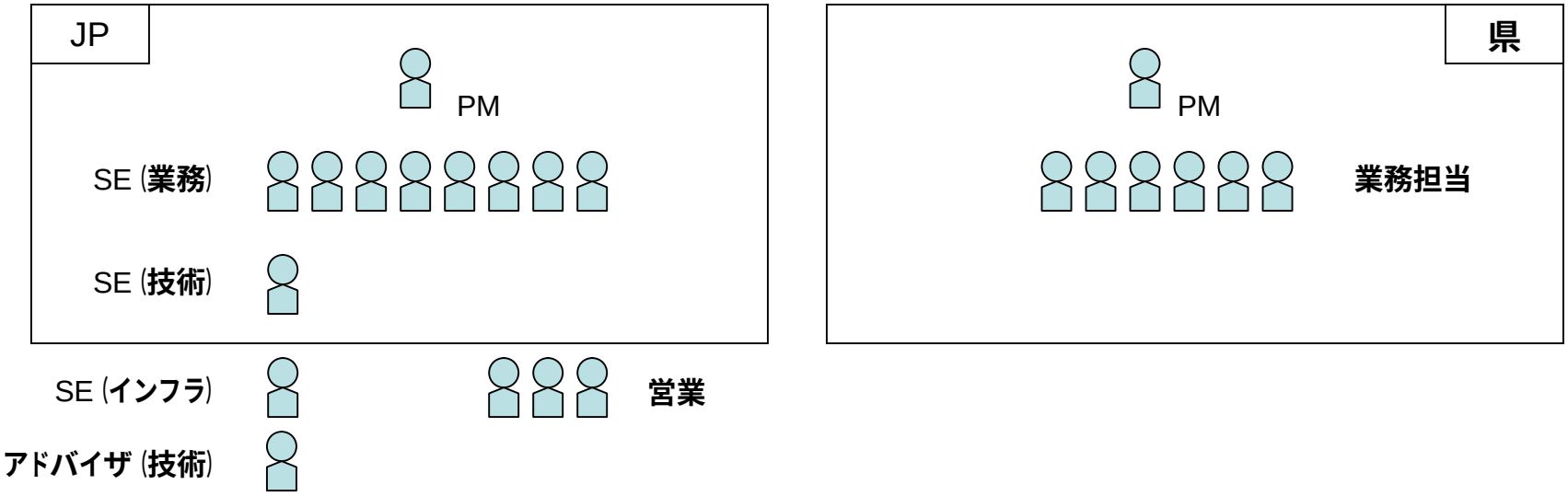
- **規模** (*1)
  - **画面** (Web)　：147
  - **帳票** (PDF **他**)　：64

(*1) H21.2**末 納品時点。競技結果登録／一般公開** (PC、**モバイル**) **、競技後処理帳票は含まず。**

開発チーム

- ジョイントプロジェクト
  - ㈱BSN アイネット、㈱ウイング、㈱ネトニー

- メンバー構成
  - お客様　：　7名　（PM：1名、業務担当：6名）
  - JP　　：10名　（PM：1名、SE：9名(*1)）　(*1) 繁忙期の最大人数



開発環境 & 実行環境

- **開発環境**
  - Windows XP
  - Ruby 1.8.6
  - Rails 2.1.1
  - PostgreSQL 8.3
  - Mongrel
- **実行環境**
  - CentOS 5
  - Ruby 1.8.6
  - Rails 2.1.1
  - PostgreSQL 8.3
  - mongrel_cluster
  - Apache HTTP Server
    + mod_proxy_balancer
  - monit

リリース

- **開発と運用が並走するため、段階的にリリース実施した。**

	[開発]	[運用]
– 2008.7	Rails **勉強**	
– 2008.8	**基本設計スタート**	
– 2008.12**末**	**参加申込リリース**	**参加申込開始**
– 2009.1**末**	**プログラム編成リリース**	**プログラム編成開始**
– 2009.**3末**		**プログラム編成完了**
– 2009.**5中**	**競技結果登録･一般公開**	
– 2009.5.23		**リハーサル大会実施**

技術的アドバイス

技術的アドバイス（1）

- **規約の理解 & 規約の遵守**
  - **規約** ＝ Rails
  - **規約をまもると、シンプルに実装できる。**
  - **規約から外れると、記述量が増加する。**

規約例

- **コントローラ と ビューの関係**
  - **コントローラ名　＝　ビューパッケージ**
  - **アクション名　＝　ビューファイル名**

■ パッケージ構造

```
app /
    controller /
        taikais_controller.rb
    views /
        taikais /
            - index.html.erb
            - show.html.erb
            - new.html.erb
            - edit.html.erb
```

■ ソースコード（taikais_controller.rb）

```
class TaikaisController < ApplicationController
  # 一覧表示
  def index
    @articles = Article.find(:all)
  end
  # 詳細表示
  def show
    @article = Article.find(params[:id])
  end
  …
```

技術的アドバイス（2）

- **適用のしやすさを考える**
  - **最初に技術評価し、Rails の得意／不得意な作業を見極める。**
  - **不得意な処理は Rails 以外の対応を検討する。**

機能要件の例

1. Web 共通
   1. 運用担当、会場担当、各選手団は Web上で担当業務を行う。
   2. 参加申込では、競技種目別に入力できる項目を切り替える。
   3. 競技記録速報は、携帯電話でも見られること。

得意

1. ログイン認証／権限別アクセス制限（共通ロジックのフィルタ適用）
2. 入力支援（Ajax）
3. モバイル対応（プラグイン jpmobile 導入）

2. 帳票
   1. 入力データをもとにシステムから生成できること。なお、参照だけでなく、編集する場合もある。

不得意

1. 帳票出力（PDF）、データ加工（Excel）

技術的アドバイス（2）

得意（共通ロジックのフィルタ適用）

- **フィルタ**
  - **適用タイミング**　before, after, around
  - **適用範囲**　:except, :only,　（**指定無し：全アクションに適用**）

```
class  TaikaisController  <  ApplicationController

  # ユーザ認証
  before_filter  :login_required

  # アクセス制限
  before_filter  :operator_access_denied,  :except => [:index, :show, :edit, :update]
  before_filter  :player_access_denied
  before_filter  :hall_access_denied

  def   index … end
  def   show  … end
end
```

業務ロジック
（一覧表示、詳細、追加、編集など）

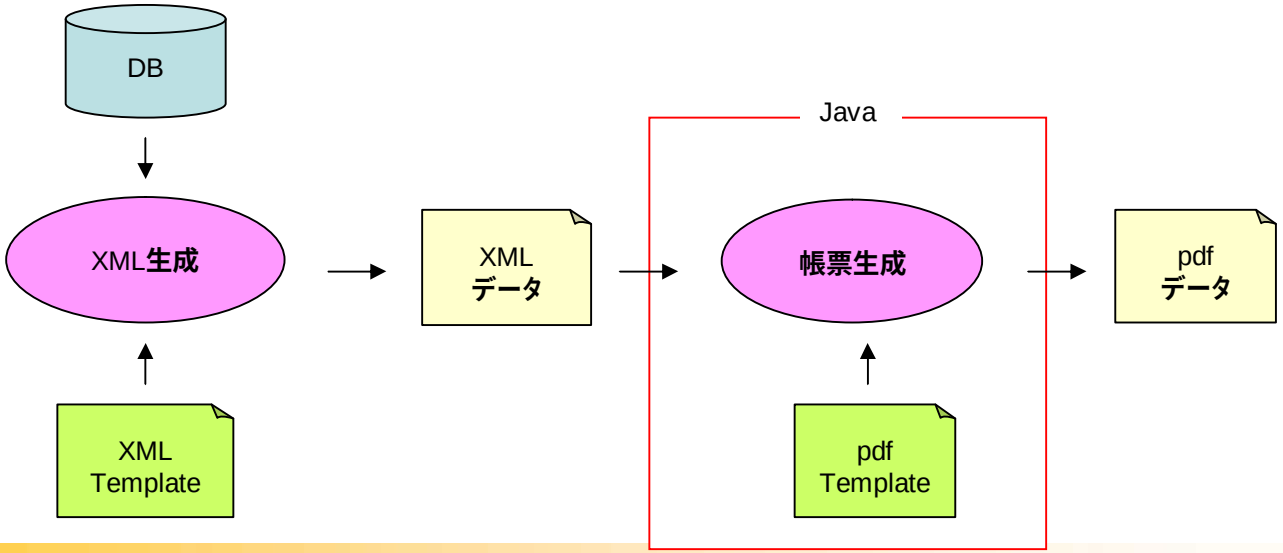
権限別に作成したアクセス不可ロジック

技術的アドバイス（2）

不得意（帳票出力（PDF））

1. Rails の作業
   1. XML 定義（xml.erb）
   2. XML データ生成

2. Java の作業
   1. iReport（GUI）で帳票テンプレート作成
   2. JasperReport で XML を帳票テンプレートにマッピング



技術的アドバイス（3）

- Java Web App サーバで実行
  - Rails サーバ の短所
    - 1プロセス ＝ 1リクエスト
    - 帳票生成など、処理時間がかかる場合、待ちリクエストがたまる。
  - 変更点
    - Warbler プラグイン導入（rails アプリを war アーカイブ化）
    - 帳票テンプレートファイル を ClassLoader 経由で取得

プロジェクト運用アドバイス

プロジェクト運用アドバイス（1）

- **最初に学習期間を設ける**
  - **規約の理解 ＝ 開発効率ＵＰ**
- **自己学習にこだわらない。**
  外部研修も積極的に活用しよう。

（研修例）
「Ruby on Rails を用いた Web アプリケーション開発」

- **主催：財団法人にいがた産業創造機構**（NICO）
- **講師：CTCテクノロジー株式会社　松田 慎弥　様**
- **期間：３日間**

プロジェクト運用アドバイス (2)

- **テストの自動化を怠らない**
  - **自動化しないと、回帰テストやバージョンアップが困難。**
  - **特に開発期間の短いプロジェクト、長期運用するシステムの場合品質維持のために必要。**

- Rails **で使えるテスト用プラグイン／ツール**
  - Shoulda、RSpec、Cucumber、Selenium **など**

その他 運用アドバイス

(Rails と直接関係ない話ですが)

その他 運用アドバイス（1）

- アジャイル開発を実践する
  - イテレーション
    - 6機能 × 13競技 × 設計製造
  - プロトタイピング
    - 紙芝居効果（システムの振る舞いの理解）
    - 顧客のシステムの動作に対する理解度が飛躍的に高まる。
  - メリット
    - 機能要件が明確になる。
    - 顧客のフィードバックを効果的に得ることができる。
    - 開発ボリューム、実現可能性（機能や期間）が判断できる。

（参考）@IT Rubyでアジャイルプロトタイピング
http://www.atmarkit.co.jp/farc/rensai2/proto01/proto01b.html

その他 運用アドバイス（2）

- **技術担当を設置する**
  - Rails の技術調査を専門に行う。
  - 他メンバーが技術ではまったら調査を代行する。
  - 共通化作業を行う。
    （プラグイン導入検討、汎用部品＆プロトタイプ作成、技術情報共有）

その他 運用アドバイス（3）

- バグトラッキングシステムを利用する
  - 開発現場が離れていても、情報共有ができる。
  - 技術情報を掲載する。
  - QA 窓口開設する。

例：redmine

- タスク管理（機能、バグ）ができる。
- 担当作業が明確になる。
- 進捗状況を把握できる。
- レポジトリ（例 SVN）との連携ができ、変更点を把握しやすい。

まとめ

(全体を通して感じたこと)

開発効率で幻想は持たない

- 学習効率は確かに高い。
  - 慣れてくると開発も早くなる。
  - 規約に従えば、コード量も少ない。

- ただし 過信はしないこと。
  - 「Java と比べて 数～10倍以上の開発効率」は幻想。
  - LOC (Line of Code) の話？

バージョンアップの落とし穴

- バージョンアップが早く、新しい機能が次々に投入される。
  - 変遷が早すぎる
    - 今、使っている技術が陳腐化しやすい。
    - テスト自動化がないと、バージョンアップに追いつけない。
  - バグフィックス
    - Rails 2.1.1 + JRuby 実行時に現れた潜在バグがあった。
  - プラグインのバージョン互換性
    - Rails 2.0 では動作するが Rails 2.1 では動作しない プラグインがあった。

Rails	
2.0.2	2007.12
2.1.0	2008.3
2.1.1	2008.9
2.1.2	2008.10
…	
2.3.2	2009.3
2.3.3	2009.7（最新）

Rails ありきの選択はしない

- Rails の特性に合わせた設計をする。
  - Rails 得意／不得意な処理（前述）

- Ruby／Ruby on Railsは、選択肢のひとつ。
  - なぜ Rails で開発するのか？を明確にしよう。

まずはレールに乗ろう！

- Rails の学習効率＆開発効率が高いのは事実。
- 開発の選択肢を広げるのは大切。
- Railsの特性をとらえて上手に活用しよう。

質疑応答