



ExtJS と Rails を連携させた業務依頼システム構築

メタデータ	言語: jpn 出版者: 公開日: 2019-04-05 キーワード (Ja): キーワード (En): 作成者: 森, 圭史朗 メールアドレス: 所属:
URL	http://hdl.handle.net/10458/6622

ExtJS と Rails を連携させた業務依頼システム構築

宮崎大学 工学部 教育研究支援技術センター

○森圭史朗, 甲斐崇浩, 相川勝, 西岡祐介

mori@teng.miyazaki-u.ac.jp

1. はじめに

技術センターでは、進歩し続ける情報処理技術に対応するため、情報処理系技術職員による新技術修得に取り組んでいる。その成果物として、技術センター業務依頼システムを構築し、2015 年 8 月に運用を開始した。運用開始後、利用者から複数の不具合や操作性向上の要望があったため、SPA（シングルページアプリケーション）を実現する ExtJS アプリケーションを用いた新たなシステム開発に取り組み、更なる技術修得を目指すことにした。運用開始時の業務依頼システムは、Rails アプリケーションの MVC アーキテクチャで構成されていた。今回のシステム開発では、図 1 の Rails アプリケーションにある View を ExtJS アプリケーションに置き換え、検索機能の強化や DB（データベース）のデータ出力機能を追加することにした。本稿では、開発を担当した箇所を中心に、ExtJS と Rails を ExtDirect により連携させた業務依頼システムについて報告する。

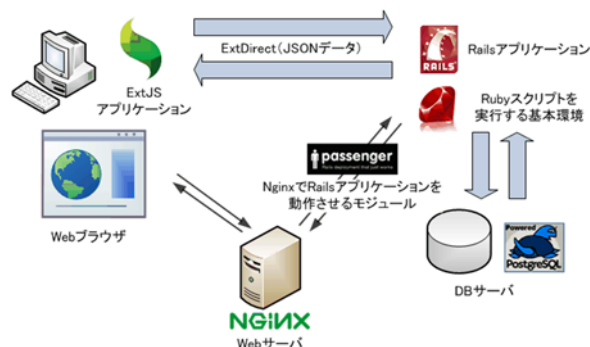


図 1. システム概要

2. ExtJS について

ExtJS は、JavaScript による RIA 構築のためのプラットフォームである。多彩な UI コンポーネ

ントを標準で備えており、アプリケーション構築に必要なクラスシステムが提供されている。ExtJS には、商用版と GPLv3 ライセンス版があるが、GPLv3 版でも必要な機能が十分に含まれていたため、システム開発に GPLv3 版を使用した。また、システム開発では、ExtJS5 から導入された MVVM 構造の Web アプリケーションを採用し、ExtJS 側 Model と Rails 間のデータ通信には、複数のリクエストを一括処理してサーバ負荷を軽減する ExtDirect の RemotingProvider を利用した。

3. システム環境

構築したシステム環境については、表 1 に示す。

表 1. システム環境（2017/10/1 現在）

分類	名称
OS	CentOS6（サーバ用 OS）
サーバ アプリケー ション	Ruby-2.4.2（Ruby 本体）
	Nginx-1.12.2（Web サーバ）
	OpenSSL-1.0.2l（通信の暗号化）
	PostgreSQL-9.6.5（DB サーバ）
主な Ruby パッケージ （Gems）	Rails-4.2.10 （Rails アプリケーション）
	Passenger-5.1.11 （Web サーバ追加モジュール）
	Capistrano-2.15.9（デプロイ）
	Seed-fu-2.3.6（DB の追加・編集）
	Activeldap-5.1.0（LDAP 認証）
	Extr-1.0.6（ExtDirect）
Sencha フレ ームワーク	ExtJS6（ExtJS アプリケーション）
	SenchaCmd-6.2.0.103（ExtJS アプリケーション作成ツール）

4. システム開発

システム開発は、週1回のミーティングを通じて互いの技術情報を共有しながら、ExtJS アプリケーション（依頼者・技術職員・管理者）と Rails アプリケーションの開発を分担して行った。ExtJS と Rails のアプリケーション構成については、図2に示す。ExtJS 開発は、Rails の public にて行い、動作確認は、Firefox のコンソールログを JavaScript デバッガとして利用した。ExtJS と Rails の Model は、アソシエーションを含めた同様の構成とし、ExtJS の Proxy によって、DB のデータと同期させた。Rails の View は、ExtJS アプリケーションルートの HTML とメール通知機能を残し、ExtJS に対応した JSON データとして Controller 内で実装した。Controller のコーディングは、ExtJS の Proxy と同じく、Read、Create、Update、Delete の Action メソッドを作成し、ファイルのダウンロードとアップロードには DomHelper、Direct Submit を利用した。また、Rails と ExtJS 間のデータ通信は、GEM パッケージ Extr の DirectRouter と ExtJS の Remoting_API を作成し、ExtDirect によるデータ通信を実現した。システムの CSRF 対策は、ExtJS アプリケーションルートへ接続する際、ExtJS の DefaultHeaders に Rails の CSRF トークンを追加することで対応した。DB データ出力機能は、管理者の業務依頼検索に追加し、CSV 形式によるダウンロードを可能にした。

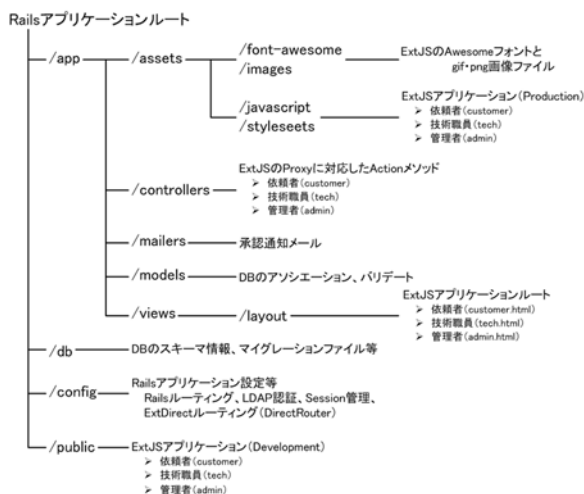


図2. ExtJS と Rails のアプリケーション構成

5. 運用に向けて

システム開発終了後、各 ExtJS アプリケーション（依頼者・技術職員・管理者）を統合した本番用業務依頼システムを構築した。まず、ExtJS は、SenchaCmd により本番用 Web アプリケーションを構築し、本番用 ExtJS にある FontAwesome、画像、JavaScript、CSS を Rails のアセットパイプラインに設置した。その後、Capistrano を用いたデプロイにより本番用 Rails アプリケーションを構築し、システムを完成させた。これにより、ExtJS の JavaScript は、不要なクラスの削除や最適化が行われ、システムレスポンスを大幅に向上させた。最後に、システムチェックと不具合修正を経て、2017年10月から ExtJS と Rails を連携させた業務依頼システムの運用を開始した（図3）。

図3. 業務依頼リスト（技術職員）

6. おわりに

ExtJS の開発は、参考となる文献が少なかったため、想定よりも時間を要してしましたが、要望を満たした検索機能と CSV 出力機能を追加した業務依頼システムを完成させることができた。また、システムを完成させたことは、メンバーの自信や意欲向上に繋がり、今後のシステム開発にも今回の経験や技術が役立つものと期待される。

参考文献

Loiane Groner. Mastering Ext JS Second Edition, Shroff Publishers & Distributors Pvt Ltd, 2014
山田祥寛. Ruby On Rails4 アプリケーションプログラミング, 技術評論社, 2014