第12回

Grails/Groovyで ソフトウェア開発の変化に対応する

NTTソフトウェア株式会社
技術開発センター　Grails推進室
上原 潤二

1 はじめに

連載をとおして、サービス・運用・システム構築の現場における、TCO削減やSI競争力の強化といった課題やビジネスモデルとしての期待等が、OSSの普及や広がりの加速を後押ししていることが語られてきた。

今回は、ソフトウェア開発におけるOSSの活用例として、Grails/Groovyの取り組みについて紹介する。

2 背景

ソフトウェア開発においても、従来より以下のような市場の要請に直面している。

◆開発体制の小規模化、開発期間の短縮
◆ビジネスの変化が速いことによる、頻繁な仕様変更（もしくは確定遅延）への対応

加えて、クラウドやスマートフォンの急速な普及も背景として、従来のウォーターフォール中心・開発言語としてJavaのみを採用する開発モデルのみでは市場の要請に迅速に応えることができなくなりつつある。このような問題に対する解決策の一つとして、NTTソフトウェアではGroovy（グルーヴィ）とGrails（グレイルズ）に注目している。

本記事では、Apache 2ライセンスに基づくオープンソースソフトウェアである、Javaベースのスクリプト言語Groovyと、Groovyの機能を最大限活用するフレームワークであるGrailsのメリットと普及推進の状況を紹介する。

3 Groovyの特長－簡潔記述

Groovyは、Javaをベースに機能拡張を行った言語である。Rubyの影響も強く受けており、記述能力はRubyに相当する。特長は、簡潔な記述が可能（Javaに比較して4分の1程度）であるということと、Javaとの親和性が高いことである。以下にコード例を示す。

●Javaコード

```
import java.util.HashMap;

public class MapTest {
  public static void main(String[] args) {
    Map<String,Object> map = new HashMap<>();
    map.put("age", 35);
    map.put("name", "山田太郎");
    for (Map.Entry<String,Object> entry: map.entrySet()) {
      System.out.println(entry.getKey()+":"+entry.getValue());
    }
  }
}
```

●同等のGroovyコード

```
map = [age:35, name:"山田太郎"]
map.each{println it.key+":"+it.value}
```

いうまでもなく、Javaは優れて安定した言語であ

り、産業の基盤を担っている存在であるが、Rubyなどのスクリプト言語と比べると記述の抽象レベルが低く冗長であり、頻繁に変更を行うような開発には向いていない。Groovyは記述が簡潔なので、機能拡張と改良、また自動テストに裏打ちされたリファクタリングを不断に続けていくことが比較的容易であり、開発速度を高めることができる。

4 Groovyの特長－Javaとの親和性の高さ

当社での開発において、Javaは高い割合で採用されているが、GroovyはJavaとの親和性が非常に高く、当社のような企業にとっては既存Java資産（Javaコードやライブラリ、APフレームワーク）、また無形の資産としての開発・運用ノウハウがそのまま利用可能であることが大きな利点となる。たとえばGroovyでは、JavaVMのセキュリティモデルやクラスパスといった概念を引き継いでおり、Mavenの膨大なJarリポジトリをそのまま利用したりすることもできる。また、GroovyはJavaのほぼ上位互換の言語であり、Java経験者にとっては学習コストが僅少で済むことが他のJava VM上の言語と比しての利点になる。

5 Grails

Grailsは、Java EE（企業向けJavaプラットフォーム）に基づいたWebアプリケーションフレームワークであり、Groovy言語で開発を行なう。Webアプリケーションの開発に必要なものがワンセットで提供され、簡単かつ分かりやすい記述で、画面表示やDB操作等を実現できる。

Grailsを採用にすることによって得られる利点は、Groovyとの相乗効果もあり、従来の一般的なJavaフレームワークを採用することとは別次元のものと言える。以降、Grailsの利点について説明する。

6 Grailsのメリット－フルスタック

Grailsには、DIコンテナであるSpring Frameworkを中心に複数のフレームワーク（Hibernate/Spring MVC/Embedded Tomcat/H2 Database/SiteMesh…）が統合されており、個々のフレームワークを選定したり、連携設定をしたり、バージョンの違いによる相性の検証などをする必要はない。また、個々のフレームワークは後述のDSLやCoC（※1）によって、直接使うよりもはるかに直感的に使用することができる。

Mavenに相当する依存性解決機構や、オートリロード可能なビルド機構、自動テストのサポート機能やwar生成の機構なども備えている。さらに、カバレッジ収集や静的解析、メトリクス収集など開発に必要な種々のツール類が、後述のプラグインの形で準備されており、簡単に利用できる。

また、Grailsを使うための包括的なドキュメントが用意されており、フレームワーク別個に用意されているドキュメント群を探し当てて、個別に読んで行く必要はない。

7 Grailsのメリット－内容に最も適した記法で書けること

Groovyは、文法や意味をカスタマイズしてDSL（※2）を実現する機能を持つ。そしてGrailsは、Groovyをカスタマイズして実現された「Webアプリケーションを書くためのDSLの集合体」と見なすことができる。

Javaでの開発においては、Java言語の仕様をカスタマイズすることはできない。「何かをわかりやすく記述したい」ときにできることはクラス名やメソッド名の付け方、あるいはその組み合せ方を工夫することだけである（もしくは、可読性の悪いXMLを組み合せる、もしくはアノテーションの山を築く）。実装言語の都合や制約から逃れることは基本的にできない。

（※1）Convention over Configration。Ruby On Railsで有名になった、設定より規約という考え方。

（※2）記述対象の領域に特化した言語。例えば「楽譜」は楽器の演奏操作を記述するためのDSLと見なせる。

しかしGrailsの場合、例えばHTTPリクエストのURLを、どのアクションに結び付けるかは、以下のような「URLマッピングDSL」を用いて記述できる。

```
class UrlMappings {
  static mappings = {
    "/$controller/$action?/$id?"{
    }
    "/"(view:"/index")
    "500"(view:'/error')
  }
}
```

詳しい説明は省くが、上記では「リクエストのURLパターンを解析し、その結果をしかるべきアクションに渡す」という処理が表現されている。このように、Javaと比べると、Groovyではコードの判り易さを向上するために努力できる範囲の次元が異なると言える。Grailsでは「どのような記述がより意味を的確に表現するか」を追求するために、徹底的にGroovyをカスタマイズしており、その結果としてコードの表現力が高くなる。他にもGrailsアプリケーションの構成要素、例えばデータベーステーブル定義（GORM）、コントローラ、サービス、各種設定ファイル等をそれぞれ工夫されたDSLで記述する。

もちろんその利点を享受するには、GrailsとGroovyを習得していることが前提になる。

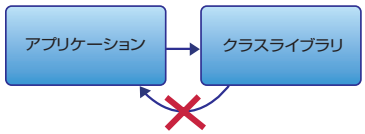
8 Grailsのメリット－プラグインとエコシステム

Grailsの3番目のメリットは、「Grailsプラグイン」と呼ばれる洗練された拡張機構を備えていることである。Grailsの公式サイトでは、多くの機能分野にまたがる854件のプラグインがオープンソースで公開されている（2012年10月現在）。また、その個数は毎週ごとに増えている。

プラグインは「コマンド一発」でインストール可能な機能拡張であり、いわゆる再利用部品と考えることができるが、一般のクラスライブラリよりもはるかに高い再利用性を実現する。その理由の一つは、プラグインがアプリケーションの構造を「知っている」ことにある。

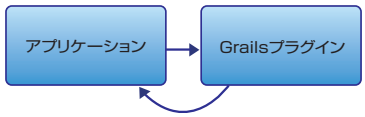
次図のように、Grailsプラグインは、アプリケーションの構造を知り、介入して動作を変更したりなど、

●クラスライブラリ（Jar,pom）の再利用



クラスライブラリは、自分を呼び出すアプリケーションについて情報を持たず、前提を置くことができない。

●Grails プラグインの再利用



アプリケーションの構造はGrailsの想定に従うので、Grailsプラグイン側からアプリケーションの実行に介在し、操作することができる。

能動的に働きかける存在である。Groovyが動的言語であることも関連し、極めて利用しやすい形でアプリケーションの機能拡張が実現できる。

9 適用事例と評価

2011年度の弊社の開発プロジェクトにおけるGrails適用事例をいくつか示す。

項番	プロジェクト名	顧客	工数削減効果(※3)	コード量比較(対Java)(※3)	規模
1	Aシステム	自社	詳細設計～結合テスト工程が、Javaの約1/2～1/5の期間で開発	Javaの約1/10	50KL
2	Bシステム	自社	詳細設計～プログラム作成工程が、Javaの約半分の工数で開発(開発全体で工数2割減)	未測定	未測定
3	Cシステム	受託	詳細設計～結合テスト工程まで、Javaの約1/7～1/10の期間で開発	Javaの約1/10	20KL

上記適用を通じての評価結果は以下の通り。

- 全体としては2～3割の工数削減。
- 概要定義まで、および総合テスト工程以降の工数は従来と大きく変わらない。これは開発プロセスとして従来と同様のウォーターフォール開

（※3）削減効果・コード量の比較対象は，当社の見積りシステムによる予測

発を適用したため。

・結合テストまでの工程における削減効果は大きい。詳細設計工程では、アプリケーションの基本的な構造についての設計が不要になる。

以下、項番2のプロジェクトメンバによる評価を示す。

・習熟が容易。Grailsはリファレンスドキュメントがまとまっており、基本的にGrails/Groovyの知識のみで開発を行うことができる。従来はJavaEE開発で必要となる複数の要素技術（Spring、JavaEE、ORマッピング、RDMBS）すべてに習熟していないと開発を開始できなかった。

・Javaスキルが活かせた。Javaと言語仕様が共通しており、Java部分の開発と並行して作業しても違和感が少ない。

・プロトタイプ開発に向いている。画面・ロジック・DB間のI/FはGrailsで規定されており、これらに関する設計工数を削減できるとともに、開発開始直後の段階からこれらを連携動作させることができる。

全体評価として、現時点では「Grails/Groovyに適したプロジェクトで採用すれば効果は大きい」と判断している。適したプロジェクトとは以下の特徴を持つものである。

・データベースのレコード内容をCRUD（作成・表示・変更・削除）するための多数画面を含む。

・開発前にすべての要件が十分に確定できず、インクリメンタルな機能拡張や変更が多い。

今後、Grails開発の経験を積み、プラグインとしての再利用部品の整備を進め、開発効率のさらなる向上、適用領域の拡大を目指していく。

10 オープンソース貢献とコミュニティ活動

NTTソフトウェアでは、Grails/Groovyを利用するだけではなくOSSとしての開発への貢献、コミュニティ活動、書籍執筆などを通じてのGrails/Groovyの普及促進に取り組んでいる。具体的には、2009年からGrails/Groovyに対する機能拡張の開発や不具合の報告、機能拡張の提案を行ってきており、また2010年にはGroovyのスクリプト言語としての使い勝手を高めるGroovyServを開発、OSSとして公開し大きな反響を呼ぶことができた。また、JGGUG（日本Grails/Groovyユーザ会）法人会員、運営委員としても活動を行なっている。

OSSであるGrailsおよびGroovyについて、現時点では日本語ドキュメントがまだ不足しているため、コミュニティを通じての情報交換や情報発信、翻訳などを活発にしていくことが非常に重要である。

11 まとめ

以上、GrailsとGroovyの利点について説明してきた。これらの機能セットは非常に大きく、紹介できたのはそのごく一部であること、また、課題についての情報紹介についても割愛していることについてはご容赦頂きたい。課題としては、例えば、従来慣れ親しんできたフレークワークや言語、手法を変更した場合、「慣れていないこと」による当初生産性の低下が確実に存在する。

しかし、初期投資としてそれを覚悟して問題を乗り越え、手足のように自在に使える、使い込んだ道具としてのフレームワークや言語を持っていることは、現代のソフトウェア開発組織として極めて重要であると考えている。取り組みの途上ではあるが、Grails/Groovy導入事例も確実に増えてきており「投資に値するものだ」という実感を日々得ている。

今後、Grails/Groovy技術が広く使われていくようにすることによって、自社のみならず、Java開発を主に行なっている企業の生産性を高め、ソフトウェア開発産業の発展を期待している。

お問い合わせ先

NTTソフトウェア株式会社
技術開発センター　Grails推進室
TEL：045-212-7023
URL：http://www.ntts.co.jp/products/grails/index.html