

Asakusaソースコードリレーディング

#4 - Asakusa Test Driver

2011/08/26

川口 章 (@apirakun)

本日の内容

- A Brief History of Asaksa Test Driver
 - TestDriverが誕生するまで (～ver 0.1)
 - TestDriverのリアーキテクト (～ver 0.2)
 - TestDriverのこれから
- Asakusa CI on AWS
- Source Code Reading
 - TestDriver ver 0.2 API

自己紹介

● 川口 章 (@apirakun)

● Asakusa Frameworkの活動

- Asakusa Test Driverの設計/実装
- CI環境のメンテナンス (Jenkins)
- ソースリポジトリ管理 (GitHub)
- ビルド・リリース管理 (Maven)
- ドキュメントの公開 (Sphinx)
- 開発環境・デプロイ周り (misc.)

● 普段の仕事

- SI:Web系の自動テストツール開発
- 流通ミドルウェアの技術サポート



川口 章
Akira Kawaguchi

リソース

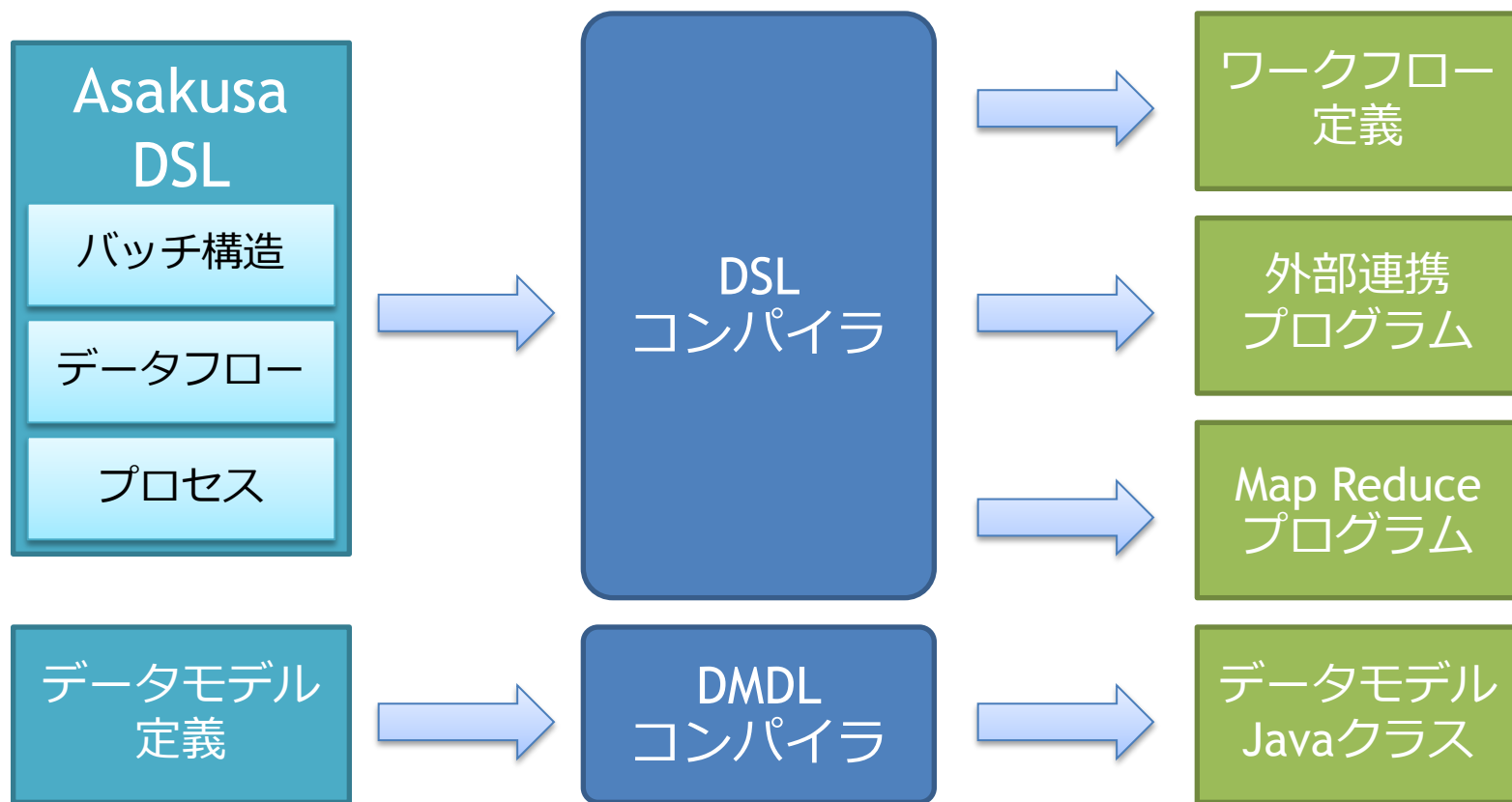
- Source Code (branch:0.2-develop)
 - <https://github.com/asakusafw/asakusafw/tree/0.2-develop>
 - asakusa-test-driver
 - asakusa-test-modelator
 - asakusa-test-data-provider
 - asakusa-thundergate-test-modelator
- Documents (アプリケーションのテスト)
 - <http://asakusafw.s3.amazonaws.com/documents/0.2/release/ja/html/testing/index.html>

「Hadoopで基幹バッチ」に耐え得るテストフレームワークとして

TestDriverが誕生するまで (～ver 0.1)

Asakusa Frameworkの構成

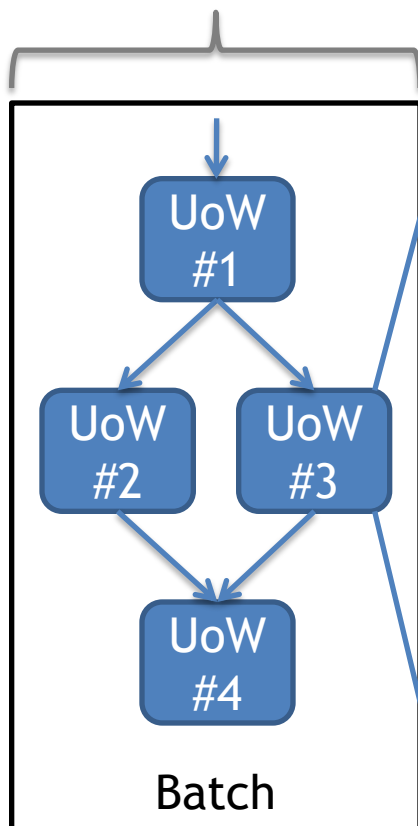
● 各種DSLを起点とした構成



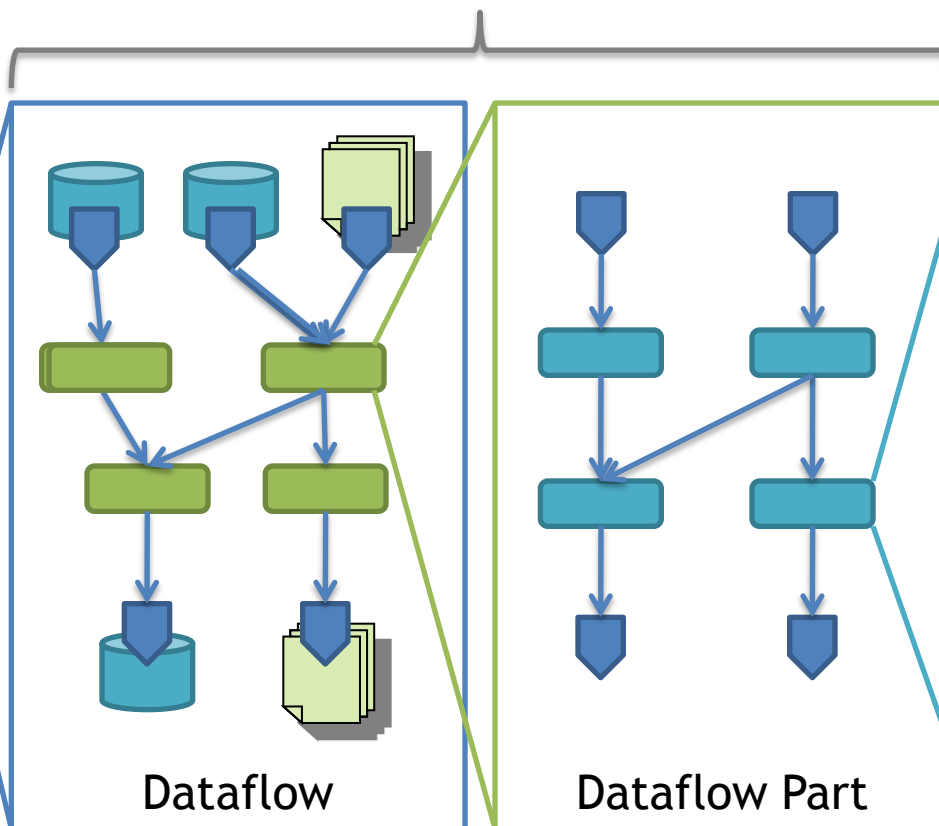
各種DSLの構造

- 層ごとに書くべきことが異なる

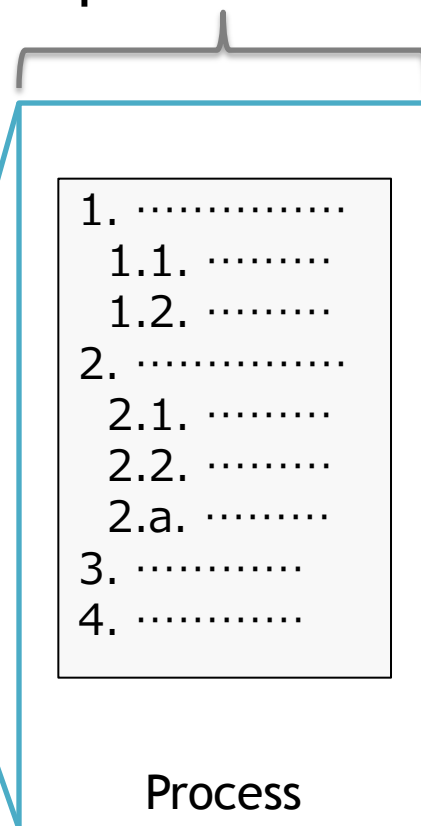
Batch DSL



Flow DSL



Operator DSL



DSLに対するテスト戦略

- 各DSL毎のテスト方法を提供
 - Operator
 - FlowPart
 - JobFlow
 - Batch
- テストデータの作成方法を提供
 - データモデル定義をベースにする
 - 複雑なデータモデルに対応する

DSL毎のテスト方針 <1/2>

- Operatorのテスト
 - 通常のJavaクラスに対するテスト
 - 少数のModelに対する入力/検証
- FlowPartのテスト
 - DSLをコンパイルして生成されるMapReduce Jobに対するテスト
 - 入出力データの実体はSequenceFile
 - 多数のModelに対する入力/検証

DSL毎のテスト方針 <2/2>

- JobFlowのテスト

- 外部インターフェース機能(ThunderGate)と連携したテスト

- Batchのテスト

- ワークフローエンジン連携機能と連携したテスト
 - 試験的ワークフロー実行機能(experimental.sh)と連携したテスト

テストデータの作成方法 <1/2>

- Model自動生成機能との連携
 - Modelと乖離しないテストデータのTemplateを作成
- 多数の複雑なModelに対する入出力の定義
 - いろいろ議論があったがExcelで
- 緻密なテスト検証方法を提供
 - テーブル毎/カラム毎に対する検査条件の指定
 - DBの定義とは別に検査用のキーカラムを指定
 - DBUnitでやろうとすると大変なことを簡単に

テストデータの作成方法 <2/2>

- DB以外のテストデータ生成・検証
 - Hadoopのシーケンスファイル
 - テスト実装者にはデータ実体の違いを隠蔽したい

テストビリティの追求 <1/5>

- テストが自動化出来る
 - CIに乗せることを前提とした方式検討
 - 必要な機能はなにか - 何を自動化するのか
 - テストデータの生成・セットアップ
 - バッチコンパイル・デプロイ
 - ThunderGate・HadoopJobの実行
 - CI Serverとの連携方式
 - 過剰な機能はなにか - 何を自動化しないのか
 - 迅速なフィードバックとテストの価値を天秤にかける

テスタビリティの追求 <2/5>

- 同じ方法でテスト出来る
 - 各種DSLの仕組みの違いを出来るだけ隠蔽
 - FlowPartとJobFlow のデータ入出力
 - SequencFile/DB上のテーブル
 - JobFlowとBatch のテスト対象の実行方法
 - `hadoopコマンド/experimental.sh`

テストビリティの追求 <3/5>

- テスト実装を迷わせない

- テスト用APIの設計

- シンプルなAPI
 - 強い制約
 - ある程度の汎用性は犠牲に

テスタビリティの追求 <4/5>

- 通常のJava開発と同様にテスト出来る
 - Eclipse上でテストも書ける
 - JUnit上でテストが実行出来る

テスタビリティの追求 <5/5>

- ローカルでテストが出来る
 - 当初はあきらめていた
 - CI環境でのみ
 - 疑似分散モード前提
- Windows上でHadoopが動作しない
 - Linux開発環境(AsakusaSDK)を提供
- 疑似分散モードはフィードバックが遅い
 - スタンドアロンモード対応

Asakusa Test Driver (0.1)

- TestDriver(0.1)が提供するもの

- testdriver.XXXTestDriver
 - FlowPart- / JobFlow- / Batch-

本資料ではパッケージ名の
com.asakusafw は省略

- 特徴

- Ashigel Compiler/HadoopJob/ThunderGateの実行を隠蔽
- ModelGeneratorから緻密なテストデータ定義が可能なテストデータのTemplate(Excel)を生成
- 各種DSLで同じようにテスト出来るシンプルなAPI
- テストが自動化出来る
- スタンドアロンモードで動作

TestDriverのさらなる成長を目指して

TestDriverのリアーキテクト

TestDriver(0.1)が出来ないこと

- 汎用性がない
 - ThunderGate決め打ち:ThunderGate以外の外部連携を使った場合のテスト手段を提供出来ない
 - Excel決め打ち:Excel以外でテストデータを定義する手段を提供していない
- 0.1の実装上に拡張を行うのは困難
 - 0.1の良いところを維持しつつ、TestDriverの内部を拡張可能にするためにリファクタリング(リアーキテクト)することに

Asakusa Test Driver (0.2)

- NewAPIの提供

- testdriver.XXXTester

- FlowPart- / JobFlow- / Batch-

- 特徴

- ThunderGate以外の外部連携に対応

- ファイルインターフェース

- Excel以外のテストデータ入出力に対応

- デフォルトはJSONとExcelを提供

- Excelフォーマットも0.1細かく改善(Calcも使える)

- SPIで拡張可能

- Ashigel Compilerに似た拡張戦略

TestDriver(0.2)のテスト実装例

● ジョブフローのテスト実装例

```
@Test
public void testExample() {
    JobFlowTester tester = new JobFlowTester(getClass());
    tester.input("shipment", Shipment.class)
        .prepare("shipment.xls#input");
    tester.input("stock", Stock.class)
        .prepare("stock.xls#input");
    tester.output("shipment", Shipment.class)
        .verify("shipment.xls#output", "shipment.xls#rule");
    tester.output("stock", Stock.class)
        .verify("stock.xls#output", "stock.xls#rule");
    tester.runTest(StockJob.class);
}
```

TestDriver(0.2)のテストデータ定義

●テストデータ定義シート例

The image displays two overlapping screenshots of the LibreOffice Calc application, showing test data definitions for a file named 'shipment.xls'.

Left Screenshot (Sheet 1/3: PageStyle_input):

The spreadsheet shows a table with the following data:

	A	B	C	D
1	sid	shipped_date	item_code	cost
2	1	2010-12-01 12:00:00	1	
3	2	2010-12-01 12:00:00	2	
4	3	2010-12-01 12:02:00	2	
5	4	2010-12-01 12:01:00	2	
6	5	2010-12-01 12:00:00	3	
7	6	2010-12-01 12:01:00	3	
8	7	2010-12-01 12:02:00	3	
9	8	2010-12-01 12:03:00	3	
10	9	2010-12-01 12:00:00	999	
11				
12				

The status bar at the bottom indicates 'シート 1 / 3 | PageStyle_input | 標準 | 合計=0'.

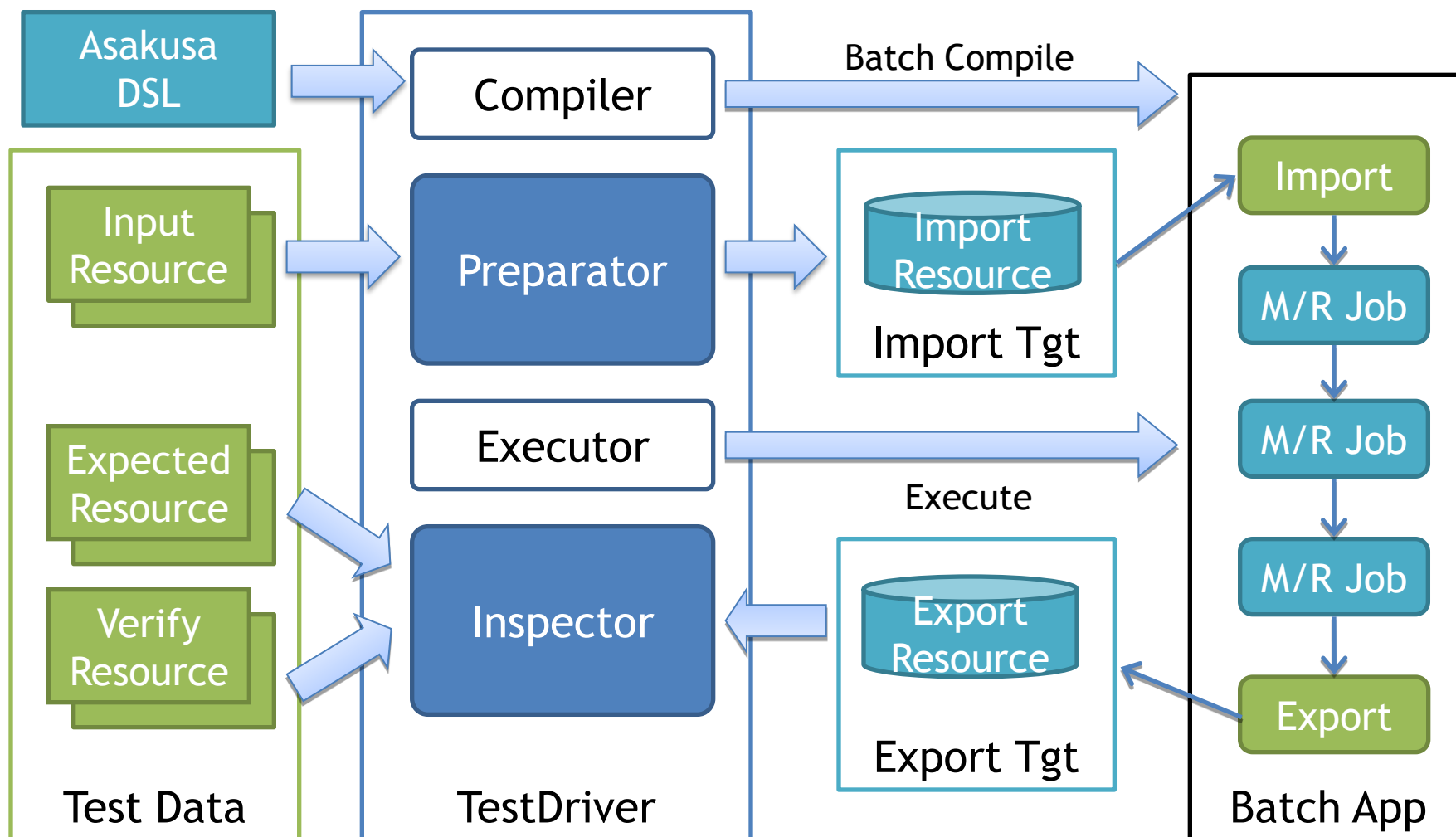
Right Screenshot (Sheet 3/3: PageStyle_rule):

The spreadsheet shows a table with the following data:

	A	B	C	D	E
1	Format	EVR-1.0.0			
2	全体の比較	全てのデータを検査 [Strict]			
3	プロパティ	値の比較	NULLの比較	コメント	
4	sid	検査キー [Key]	通常比較 [-]	SID	
5	version_no	検査対象外 [-]	通常比較 [-]	VERSION_NO	
6	rgst_datetime	検査対象外 [-]	通常比較 [-]	RGST_DATETIME	
7	updt_datetime	検査対象外 [-]	通常比較 [-]	UPDT_DATETIME	
8	shipped_date	完全一致 [=]	通常比較 [-]	SHIPPED_DATE	
9	item_code	完全一致 [=]	通常比較 [-]	ITEM_CODE	
10	cost	完全一致 [=]	通常比較 [-]	COST	
11					
12					

The status bar at the bottom indicates 'シート 3 / 3 | PageStyle_rule | 標準 | 合計=0 | 100%'.

TestDriver(0.2)のアーキテクチャ概要



さらなる進化に向けて

これからのTestDriver

テストを短時間で終了させる

- Distributed Testing

- 複数のマシンでテストを並列実行させてテストを短時間で終了させる

- Cloud Testing

- AWSと連携

検証機能の強化

- Define the relationship between input and output
 - データの移動が発生するだけの項目は「この項目は入力モデルのこれと等しい」というように定義したい
 - MOVE文のテスト
 - テストデータの定義から仕様(テストの観点)を追いやすくする

テストデータ作成の自動化

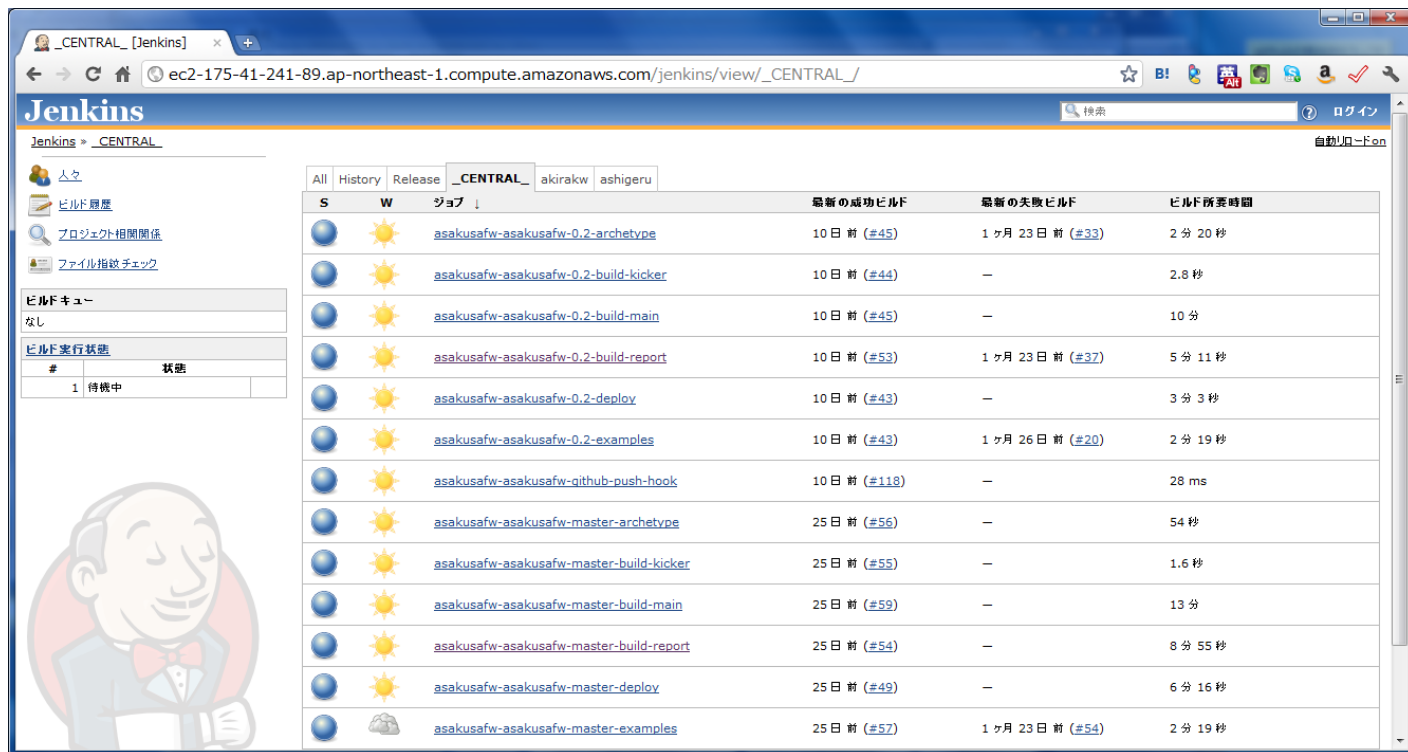
- テストデータパターンを定義する
 - テストデータ定義にテストデータパターンを記述し、パターンから生成されたテストデータに対してテストドライバを流す

Asakusa Framework の CI環境

Asakusa CI on AWS

Asakusa FrameworkのCI環境

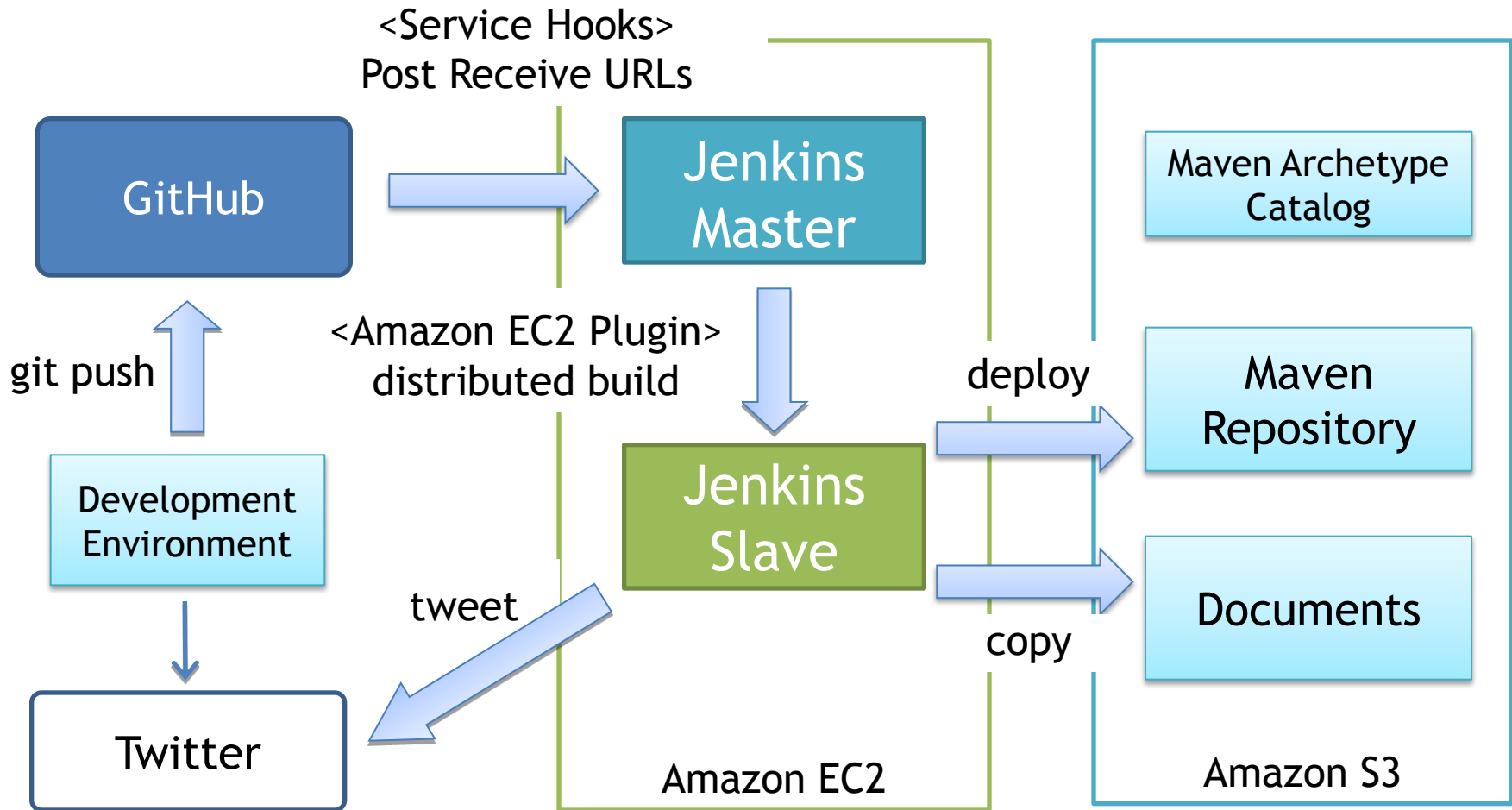
- AWS(EC2/S3)上にCI環境を構築しています
- <http://ec2-175-41-241-89.ap-northeast-1.compute.amazonaws.com/jenkins/>



The screenshot shows the Jenkins web interface in a browser. The address bar displays the URL: ec2-175-41-241-89.ap-northeast-1.compute.amazonaws.com/jenkins/view/_CENTRAL_/. The page title is "Jenkins". On the left sidebar, there are links for "ビルド履歴" (Build History), "プロジェクト相関関係" (Project Relationships), and "ファイル指紋チェック" (File Fingerprint Check). Below these, there are sections for "ビルドキュー" (Build Queue) and "ビルド実行状態" (Build Execution Status). The main content area shows a table of builds for the "_CENTRAL_" job. The table has columns for "S" (Status), "W" (Web icon), "ジョブ" (Job), "最新の成功ビルド" (Latest Successful Build), "最新の失敗ビルド" (Latest Failed Build), and "ビルド所要時間" (Build Time). The jobs listed include "asakusafw-asakusafw-0.2-archetype", "asakusafw-asakusafw-0.2-build-kicker", "asakusafw-asakusafw-0.2-build-main", "asakusafw-asakusafw-0.2-build-report", "asakusafw-asakusafw-0.2-deploy", "asakusafw-asakusafw-0.2-examples", "asakusafw-asakusafw-github-push-hook", "asakusafw-asakusafw-master-archetype", "asakusafw-asakusafw-master-build-kicker", "asakusafw-asakusafw-master-build-main", "asakusafw-asakusafw-master-build-report", "asakusafw-asakusafw-master-deploy", and "asakusafw-asakusafw-master-examples".

S	W	ジョブ	最新の成功ビルド	最新の失敗ビルド	ビルド所要時間
●	☀	asakusafw-asakusafw-0.2-archetype	10日前 (#45)	1ヶ月23日前 (#33)	2分20秒
●	☀	asakusafw-asakusafw-0.2-build-kicker	10日前 (#44)	—	2.8秒
●	☀	asakusafw-asakusafw-0.2-build-main	10日前 (#45)	—	10分
●	☀	asakusafw-asakusafw-0.2-build-report	10日前 (#53)	1ヶ月23日前 (#37)	5分11秒
●	☀	asakusafw-asakusafw-0.2-deploy	10日前 (#43)	—	3分3秒
●	☀	asakusafw-asakusafw-0.2-examples	10日前 (#43)	1ヶ月26日前 (#20)	2分19秒
●	☀	asakusafw-asakusafw-github-push-hook	10日前 (#118)	—	28 ms
●	☀	asakusafw-asakusafw-master-archetype	25日前 (#56)	—	54秒
●	☀	asakusafw-asakusafw-master-build-kicker	25日前 (#55)	—	1.6秒
●	☀	asakusafw-asakusafw-master-build-main	25日前 (#59)	—	13分
●	☀	asakusafw-asakusafw-master-build-report	25日前 (#54)	—	8分55秒
●	☀	asakusafw-asakusafw-master-deploy	25日前 (#49)	—	6分16秒
●	☁	asakusafw-asakusafw-master-examples	25日前 (#57)	1ヶ月23日前 (#54)	2分19秒

Asakusa CI環境の構成



MasterとSlave

- Jenkins Master

- マイクロインスタンスを常に稼働させておく
- JenkinsのWebフロントエンドとしてのみ使用

- Jenkins Slave

- 主要なビルド処理はすべてここで実行
- High-CPUミディアムインスタンスを実行時に起動
 - スモールインスタンスのほぼ半分の時間でビルドが完了

- Amazon EC2 Plugin

- JenkinsスレーブをEC2のAMIとして起動する
- Idleが30分続くと自動的に停止
- CIサーバのランニングコスト:月々1,000円～2,000円

Jenkins Amazon EC2 Plugin

● Jenkins Slaveとして起動するAMIを指定

クラウド

Amazon EC2

Region: Asia Pacific (Singapore) ?

Access Key ID: US East (Virginia) ?

Secret Access Key: US West (N. California) ?

EC2 RSA Private Key: EU West (Ireland) ?

Asia Pacific (Singapore) ?

-----BEGIN RSA PRIVATE KEY-----

高度な設定...

Generate Key

Test Connection

AMIs

AMI ID: ami-4c1ab04d

Check AMI

Instance Type: MEDIUM_HCPU ?

Description: Asakusa-Jenkins-Slave ?

Remote FS root: /var/jenkins ?

Remote user: root ?

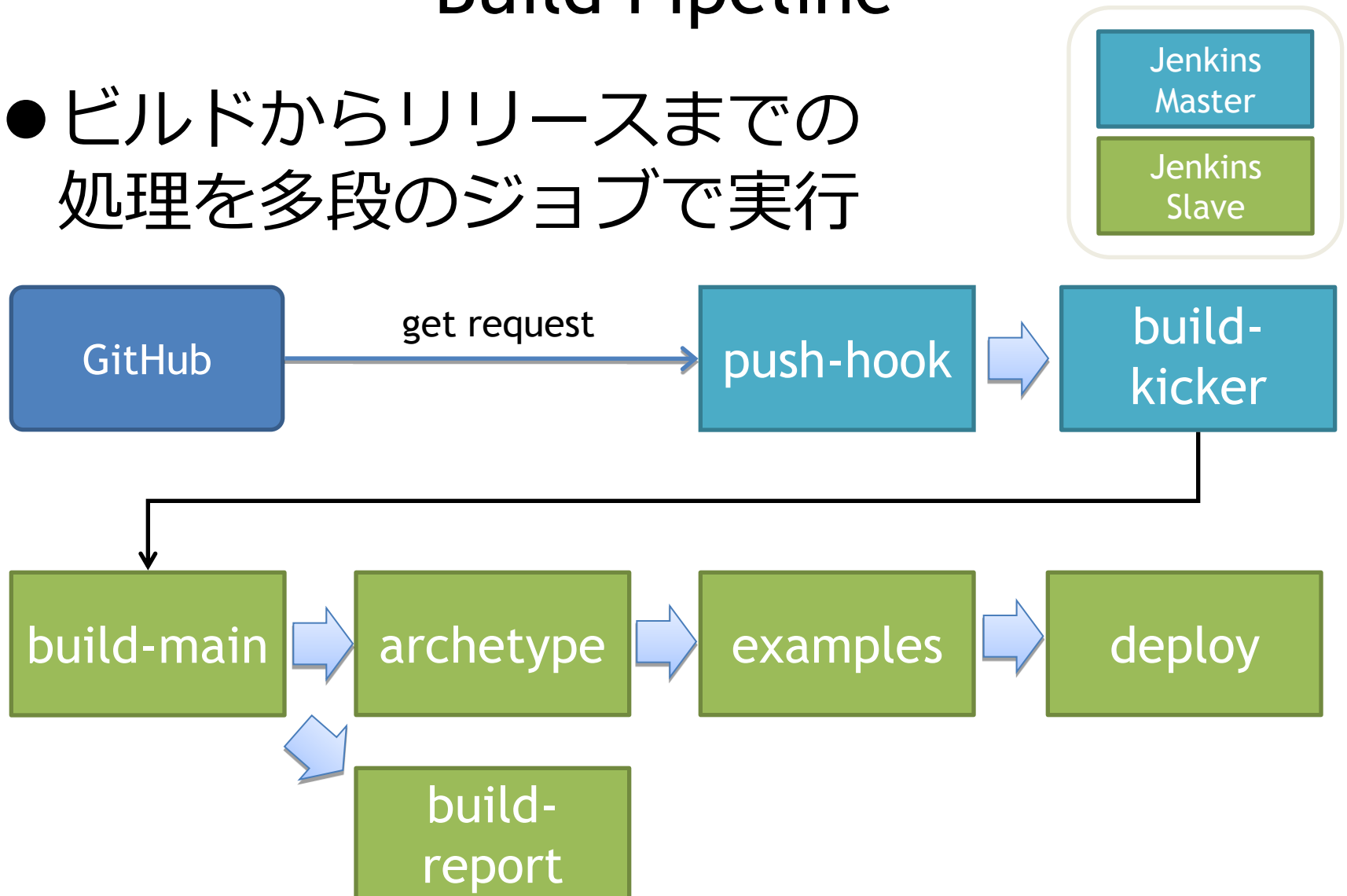
Labels: EC2-Default ?

GitHubとJenkinsの連携

- GitHubからJenkinsへpush通知
 - GitHubの「Post Receive URLs」を利用
 - git pushを受けると即時にJenkinsに対してget
 - git pushしたブランチだけをビルドする
 - Post Receive URLs はRequest URLが固定
 - 特定のJenkins Jobを起動することしか出来ない
 - 窓口用のJobで受けて、下流のJobにビルドする可能性があるブランチを定義したJobをすべて紐付け
 - これで一応出来るが、ムダなビルドが多数発生
 - Jenkinsの「Downstream-Ext Plugin」を利用
 - リポジトリに変更がない場合ビルドを中止出来る

Build Pipeline

- ビルドからリリースまでの処理を多段のジョブで実行



ビルド結果のフィードバック

● Twitterプラグイン

● @asakusa_ci

	asakusa_ci AsakusaFramework CI SUCCESS:asakusafw-asakusafw-0.2-build-main \$46 - tinyurl.com/3lub6sw 8月23日
	asakusa_ci AsakusaFramework CI SUCCESS:akirakw-asakusafw-0.2-build-main \$6 - tinyurl.com/445c6em 8月23日
	asakusa_ci AsakusaFramework CI SUCCESS:akirakw-asakusafw-0.2-examples \$6 - tinyurl.com/3hpw4pj 8月23日
	asakusa_ci AsakusaFramework CI UNSTABLE:akirakw-asakusafw-0.2-examples \$5 - tinyurl.com/3u636ol 8月23日



Jenkins

Jenkins > asakusafw-asakusafw-0.2-build-main > #46

プロジェクトへ戻る

検索

変更履歴

コンソール出力 [未加工]

説明の編集

Git Build Data

テスト結果

指紋を見る

前のビルド

ビルド #46 (2011/08/23 21:04:14)

Changes

1. Enable to input expect data from database table (#64). ([commit: 18334429c48ce0360a4448810b795c7dc7078a1e](#)) ([detail](#))
2. fixed #65. Redundant assert log message with date type. ([commit: c775fdead0e6f7538eaa166e54b602e6fda1359f](#)) ([detail](#))

上流プロジェクト [asakusafw-asakusafw-0.2-build-kicker](#) の #45 が実行

Revision: c775fdead0e6f7538eaa166e54b602e6fda1359f

- origin/0.2-develop

テスト結果 (全て成功)

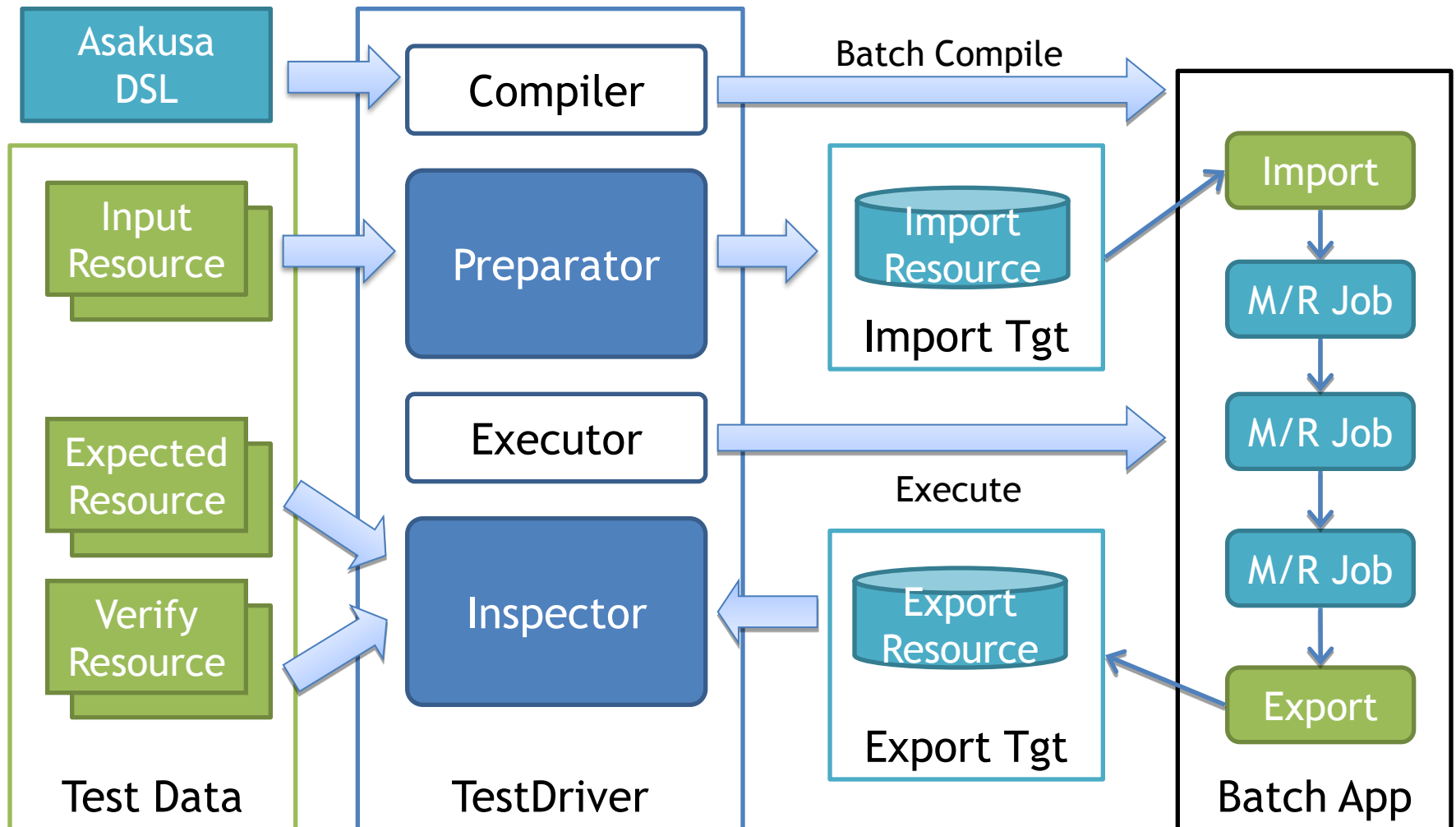
モジュールビルド

Asakusa Framework Aggregator POM	0.16 秒
Asakusa Framework Maven Archetype for BatchApp	0.94 秒
Asakusa Cleaning Tool	3.4 秒
Asakusa Framework Distribution Module	2.1 秒
DMDL Core	3.9 秒
DMDL for Java Data Models	5.2 秒
Vocabularies for Asakusa DSL	7.4 秒
Asakusa Data Model Generator	5.6 秒
Asakusa Framework Parent POM	2.9 秒
Asakusa Batch Runtime Library	15 秒

拡張ポイントを中心に

Asakusa Test Driver Source Code Reading

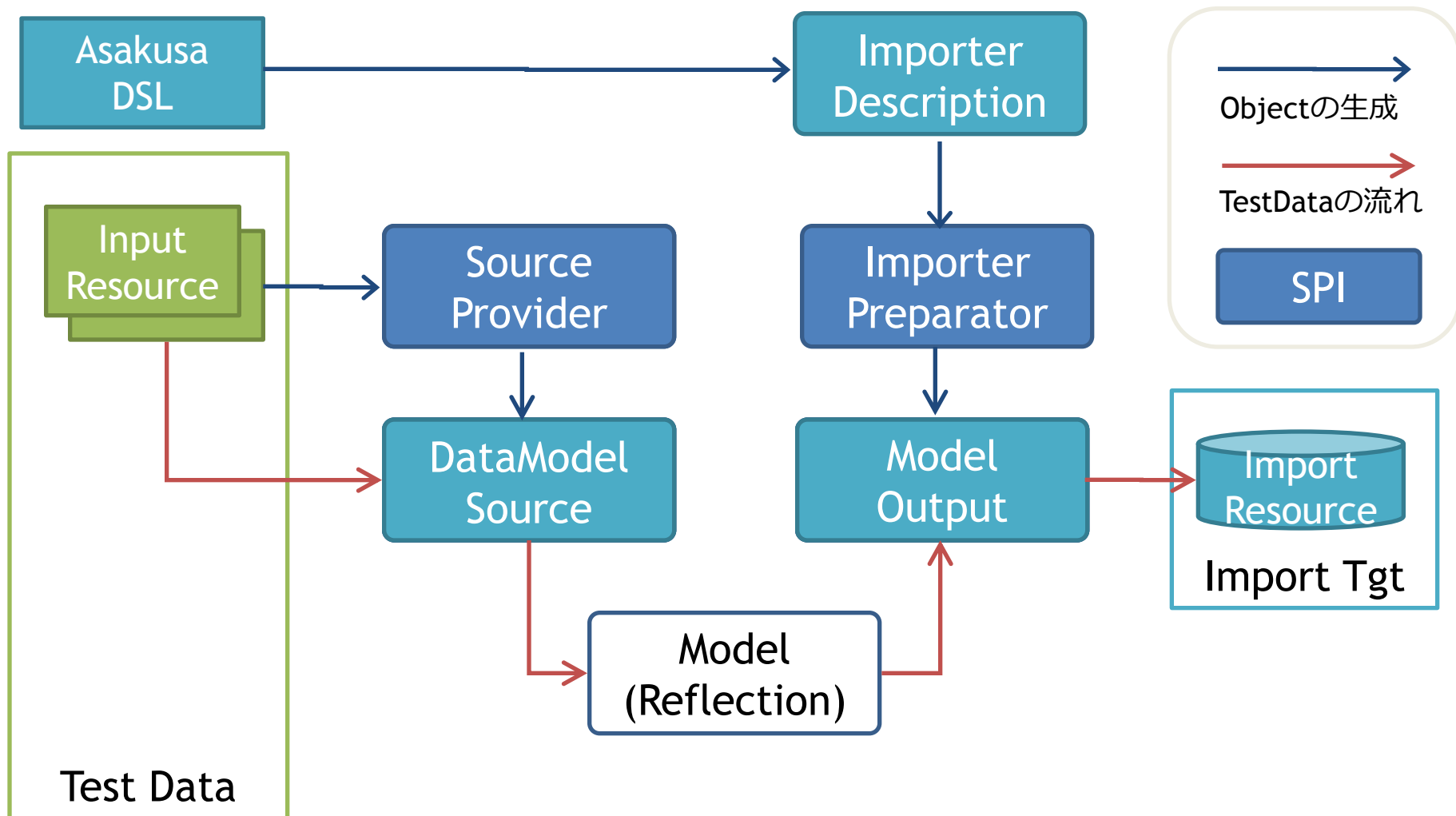
TestDriver(0.2)のアーキテクチャ概要(再掲)



Compilerを詳しく

- テスト対象のDSLをバッチコンパイル
 - Ashigel Compilerの呼出
 - DirectFlowCompiler
 - DirectBatchCompiler
 - Asakusaソースコードリーディング#1 を参照
 - TestDriverでは簡易実行計画を扱う
 - ステージの実行計画をトポロジカルソート
 - 逐次実行

Preparatorを詳しく



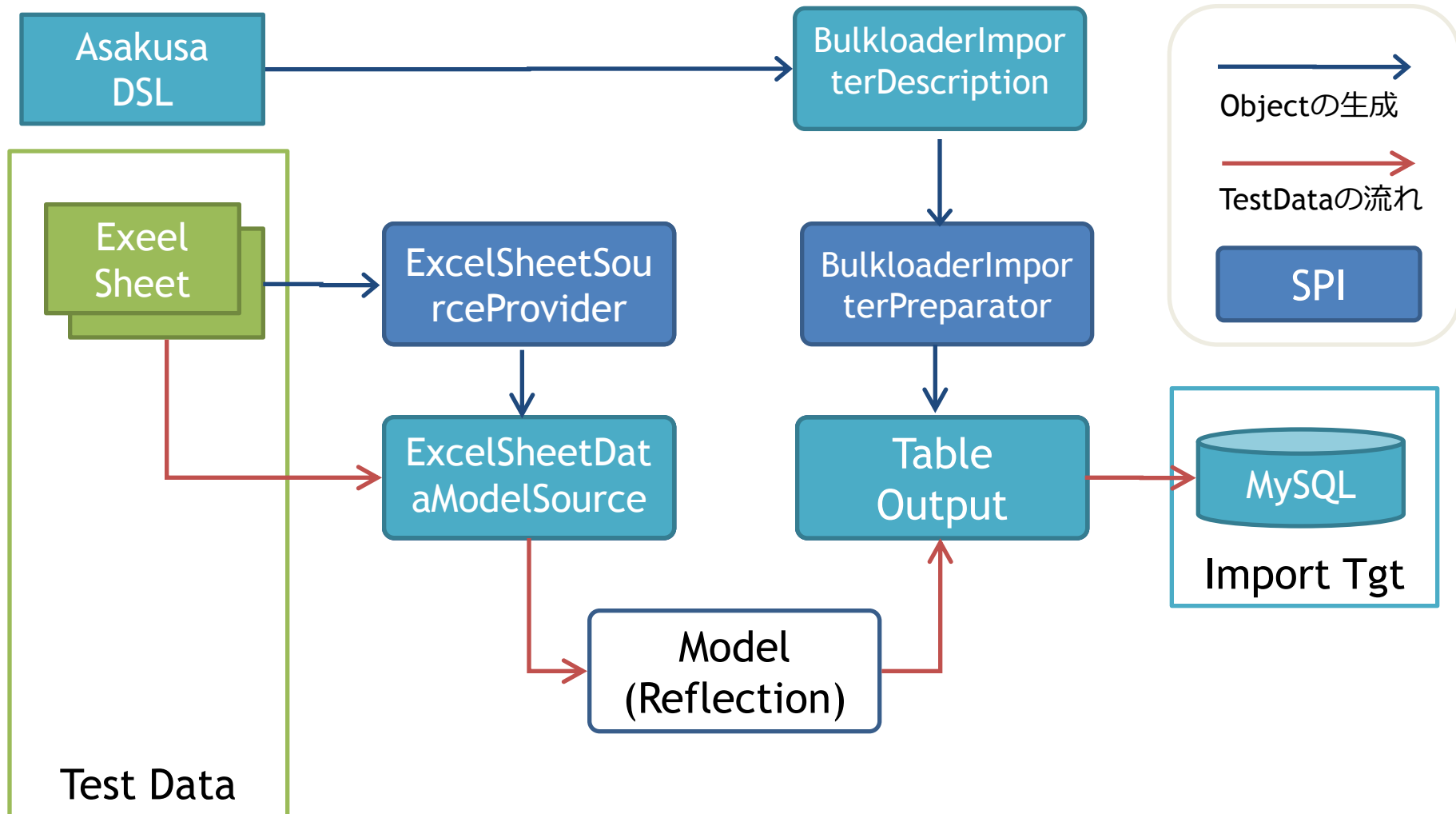
SPIで拡張可能な部品

- `testdriver.core.SourceProvider`
 - URIに対応するModelのデータソース (`DataModelSource`)を提供
 - ver0.2.1時点では以下の実装を提供
 - `testdriver.json.JsonSourceProvider`
 - `testdriver.excel.ExcelSheetSourceProvider`

SPIで拡張可能な部品

- `testdriver.core.ImporterPreparator`
 - Import対象に対応したモデル出力オブジェクト(`ModelOutput`)を提供
 - ver0.2.1時点では以下の実装を提供
 - `testdriver.file.FileImporterPreparator`
 - `testdriver.bulkloader.BulkloaderImporterPreparator`

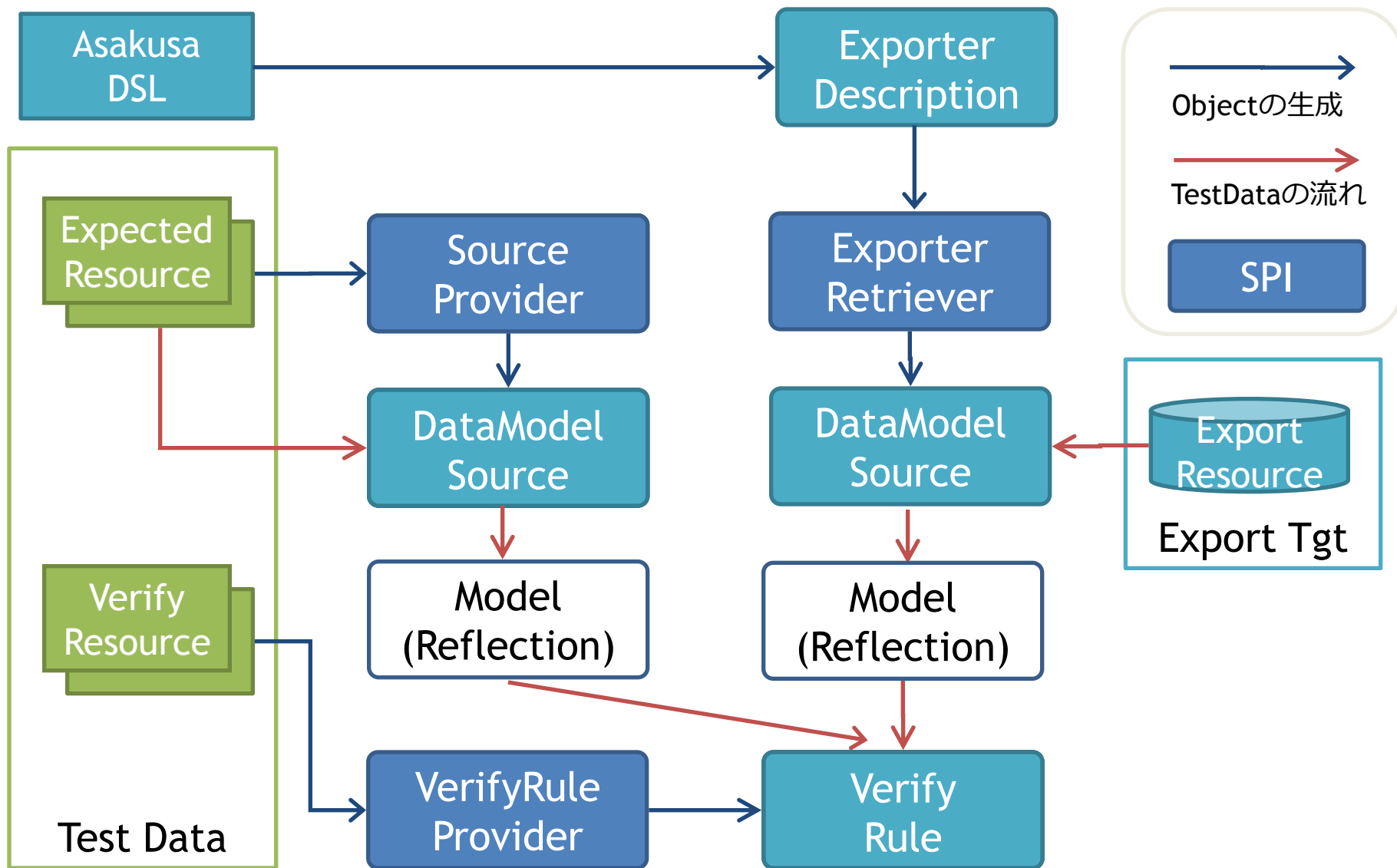
ThunderGate/Excelデータの場合



Executorを詳しく

- ステージに対応するコマンドを順次実行
 - JobflowInfo
 - ExternalCommandProvider
 - ver0.2.1時点ではOS(Linux)依存のコマンドを出力

Inspectorを詳しく



SPIで拡張可能な部品

- `testdriver.core.ExporterRetriever`
 - Export対象に対応したModelのデータソース(`DataModelSource`)を提供
 - ver0.2.1時点では以下の実装を提供
 - `testdriver.file.FileExporterRetriever`
 - `testdriver.bulkloader.BulkloaderExporterRetriever`

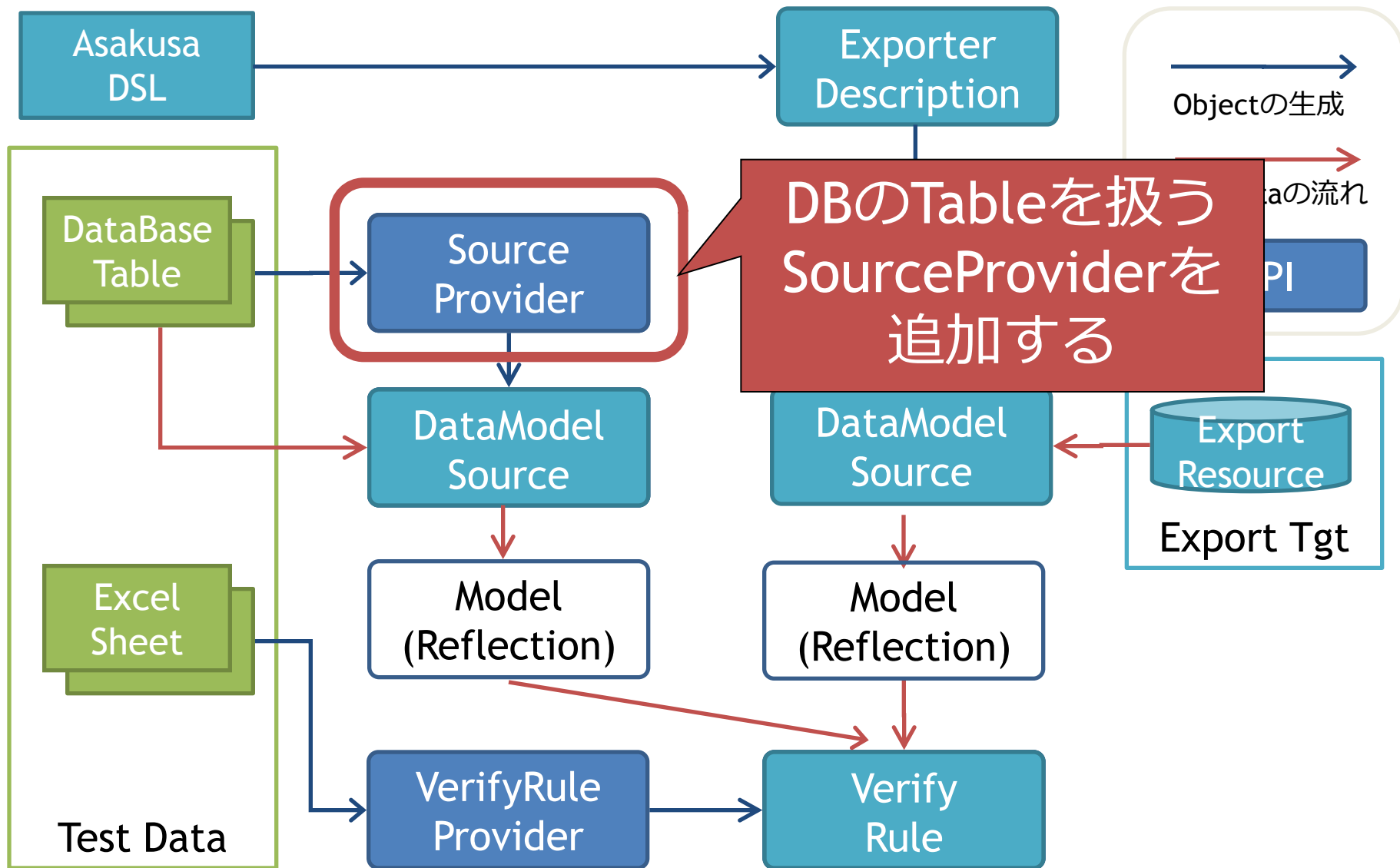
SPIで拡張可能な部品

- `testdriver.core.VerifyRuleProvider`
 - URIに対応する検証ルール解析オブジェクト (`VerifyRule`)を提供
 - ver0.2.1時点では以下の実装を提供
 - `testdriver.excel.ExcelSheetRuleProvider`
- 検証ルールをJavaで記述することも可能
 - `testdriver.core.ModelVerifier`

拡張してみる

- 例:期待値をデータベースに定義する
 - モチベーション
 - Excelのレコード数の上限を超える
 - 既存システムのテーブルデータをそのままテストデータに活用
 - ツールでテストデータを作りやすいかも

SourceProviderを追加



実はあります

- Issue#64: Enable to input expect data from database table.
 - <https://github.com/asakusafw/asakusafw/issues/64>
- 修正内容
 - TableSourceProviderを追加
 - testdriver.bulkloader.TableSourceProvider
 - SPIの定義ファイルを追加
 - src/main/resources/META-INF/services/com.asakusafw.testdriver.core.SourceProvider
- 使い方
 - URIに以下の文字列を指定
 - “bulkloader:<target_name>:<table_name>”
 - 例:”bulkloader:asakusa:ORDER”