Play2実践Tips集

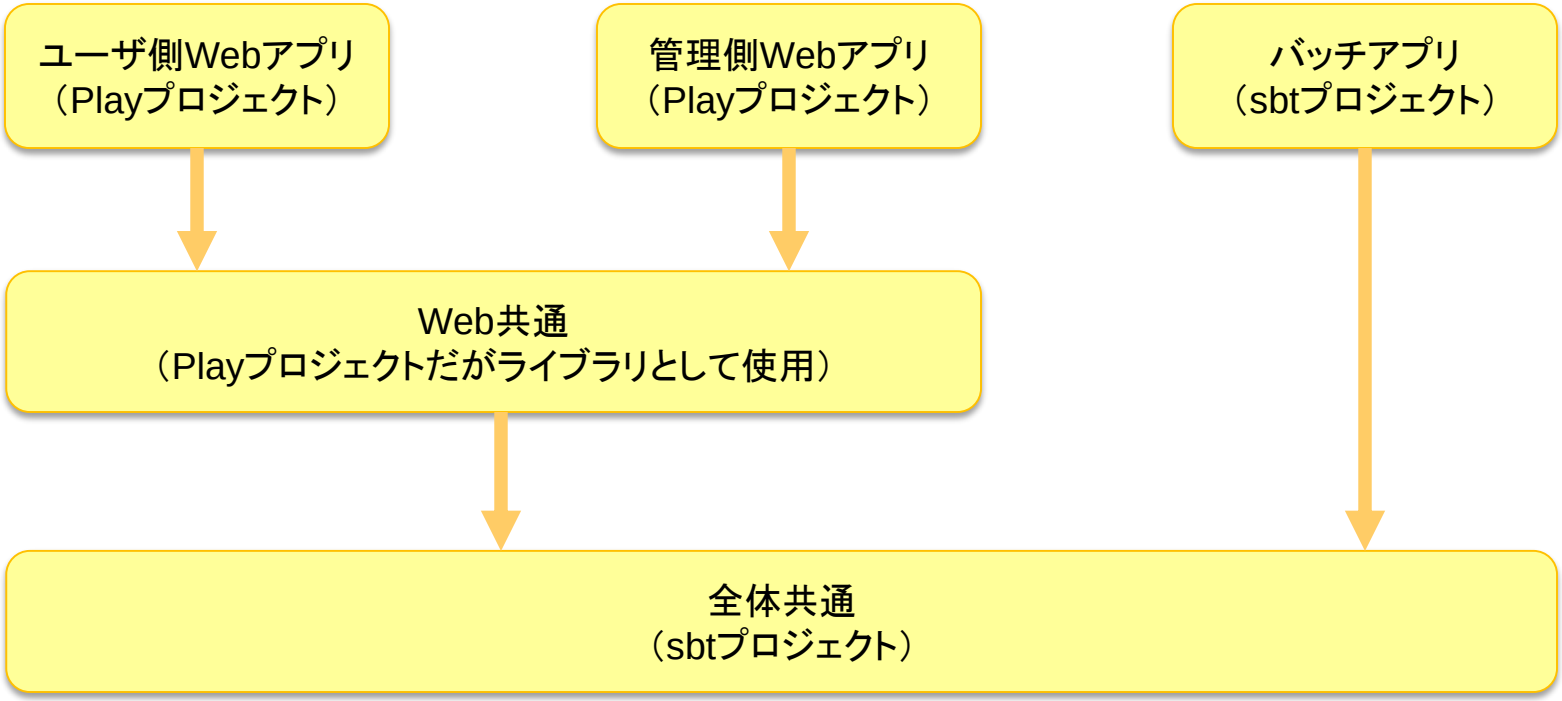
Naoki Takezoe
BizReach, Inc.

Topic

- プロジェクト構成
- 開発モードの高速化
- マッピング、フォーマット
- コネクションプール
- 日本語ルーティング
- セッション管理
- 認証処理
- 非同期処理、ジョブスケジューリング
- 運用時のあれこれ

プロジェクト構成

- IntelliJで1プロジェクトで開発するためにsbtのマルチプロジェクト構成にしています

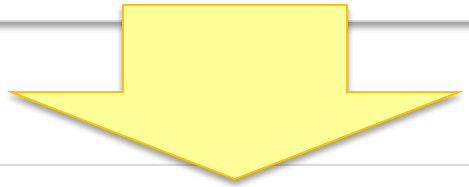


開発モードの高速化

BIZREACH

- play2-fastassetsを使用して静的リソースをブラウザにキャッシュすることでリクエストを抑制
  - https://github.com/takezoe/play2-fastassets

```
<link rel="stylesheet" media="screen"
      href="@FastAssets.at("stylesheets/main.css")">
```



```
<link rel="stylesheet" media="screen"
      href="assets/stylesheets/main.css?142014022630">
```

バージョンタグ付きのパスを生成するのでファイルを変更した場合はリクエストが飛ぶ
運用時のバージョニングにも使用（運用時のバージョニングはPlay 2.3で入る模様）
https://github.com/playframework/playframework/issues/1897

フォームのTuple22問題

BIZREACH

- 項目数の多いフォームを定義できない
  - 項目を適当な粒度でグルーピングしてネストさせる
  - Scala 2.11では解消されるはず…

```
val userForm = Form(
  mapping(
    "firstName"      -> text,
    "lastName"       -> text,
    "mailAddress"    -> email,
    "password"       -> text,
    "companyInfo"    -> mapping(
      "company"      -> text,
      "department"   -> text
    )(CompanyInfo.apply)(CompanyInfo.unapply)
  )(UserInfo.apply)(UserInfo.unapply)
)
```

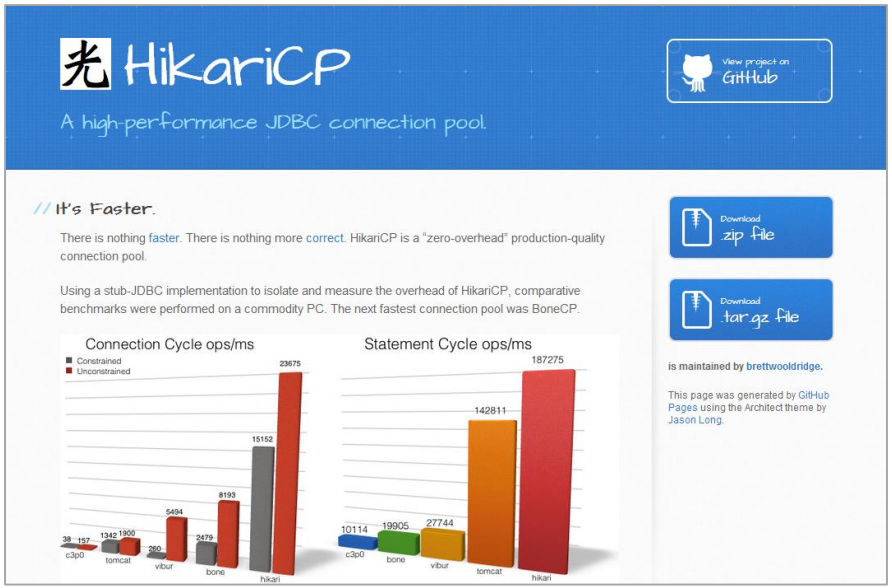
テンプレート

- Play標準のヘルパー（@inputText等）は基本的に使用していない
  - 余計なマークアップが出力されてしまう
  - エラーメッセージが微妙
  - デザイナとの連携の問題

コネクションプール

BIZREACH

- BoneCPには複数の問題があるため、Play標準のdbpluginを無効にしてHikariCP play pluginを使用
  - http://edulify.github.io/play-hikaricp.edulify.com/



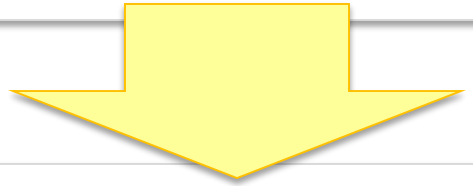
dbpluginを無効にするとevolutionが使えなくなるので注意！
Playのdbpluginに依存しないevolutionの代替としてplay-flywayがあります。
https://github.com/tototoshi/play-flyway

日本語ルーティング

BIZREACH

- ルーティングに日本語パスを定義したい

```
# 標準ではURLエンコードしたパスをroutesに記述する必要がある
GET     /%E3%83%8F%E3%83%AD%E3%83%BC    controllers.Application.hello
```



```
# パスを日本語で記述できるように拡張
GET     /ハロー     controllers.Application.hello
```

日本語ルーティング

- Globalでパスを変換

```
import play.api._
import play.api.mvc._
import java.net.URLDecoder.decode

object Global extends GlobalSettings {

  override def onRouteRequest(req: RequestHeader): Option[Handler] = {
    super.onRouteRequest(
      if(req.path.startsWith("/%")) req.copy(path = decode(req.path,
"UTF-8")) else req)
  }

}
```

Globalにもこの他にも様々なフックメソッドが用意されており、オーバーライドすることでPlayの全体的な動作のカスタマイズが可能です。

セッション管理

- PlayはRails同様のクッキーセッション機能
- サーバサイドにデータを保存する仕組みは自分で用意する必要がある
- PlayのCache APIを使用してセッションIDと紐付けて保存する機能を自作
- ローカルではEhcache、プロダクションではMemcachedを使用（設定で切り替え可能）
  - https://github.com/mumoshu/play2-memcached

認証処理

- Secure Social
  - http://securesocial.ws/
  - デフォルトで様々なソーシャルメディアに対応
  - ユーザ登録、メール通知等の機能も組み込まれており、設定は複雑だが高機能
  - そこまで高機能なものが不要であればpac4jもいいかも
- play2-auth
  - https://github.com/t2v/play2-auth
  - 権限によるアクションの認証機構を提供
  - システム独自のユーザ管理を行う業務アプリケーション等に向いている

非同期処理

- メール送信や外部APIの呼び出しなど時間のかかる処理はPlayのコントローラからアクターを起動して非同期実行

```
object SendMailController extends Controller {

  val actor = Akka.system.actorOf(Props[SendMailActor],
                                  name = "sendMailActor")

  def send = DBAction { implicit request =>
    // メール送信対象のユーザを取得
    val accounts = Accounts.filter(_.acceptMail is true.bind).list
    // 送信対象のユーザを指定してアクターを起動
    actor ! users
    // レスポンスを返す
    Ok(views.html.test("test page."))
  }

}
```

非同期処理

- 定期的に行う処理はakka-quartzを使用してアクターをスケジュール実行
  - https://github.com/theatrus/akka-quartz

```
// スケジューラを取得
val quartzActor = Akka.system.actorOf(Props[QuartzActor])

// スケジュールを登録
quartzActor ! AddCronSchedule(
  Akka.system.actorOf(Props[HelloWorldActor]), // アクター
  "0/5 * * * * ?",                             // スケジュール
  "HelloWorld"                                 // メッセージ
)
```

Typesafeで開発しているakka-quartz-schedulerというものもありますが、application.confにスケジュールを記述する必要があるので動的にスケジュールを設定したいという用途には向いていません。
https://github.com/typesafehub/akka-quartz-scheduler

デプロイ、運用にかんするあれこれ

BIZREAC月

- distで生成したzipファイルをサーバに転送、展開してシェルスクリプトを実行して起動
  - Commons Daemon（jsvc）を使ってデーモン化 http://d.hatena.ne.jp/takezoe/20140516#p1
- Playの起動時にオプションで設定ファイルを指定
  - -Dconfig.resource=クラスパス内の設定ファイル
  - -Dconfig.file=ファイルシステム上の設定ファイル
- ログにはslf4j+LogBackを使用
  - LogBackの設定は全開にしておき、Play側の設定でログレベルを調節している