実案件で
Play Frameworkを
採用してみた

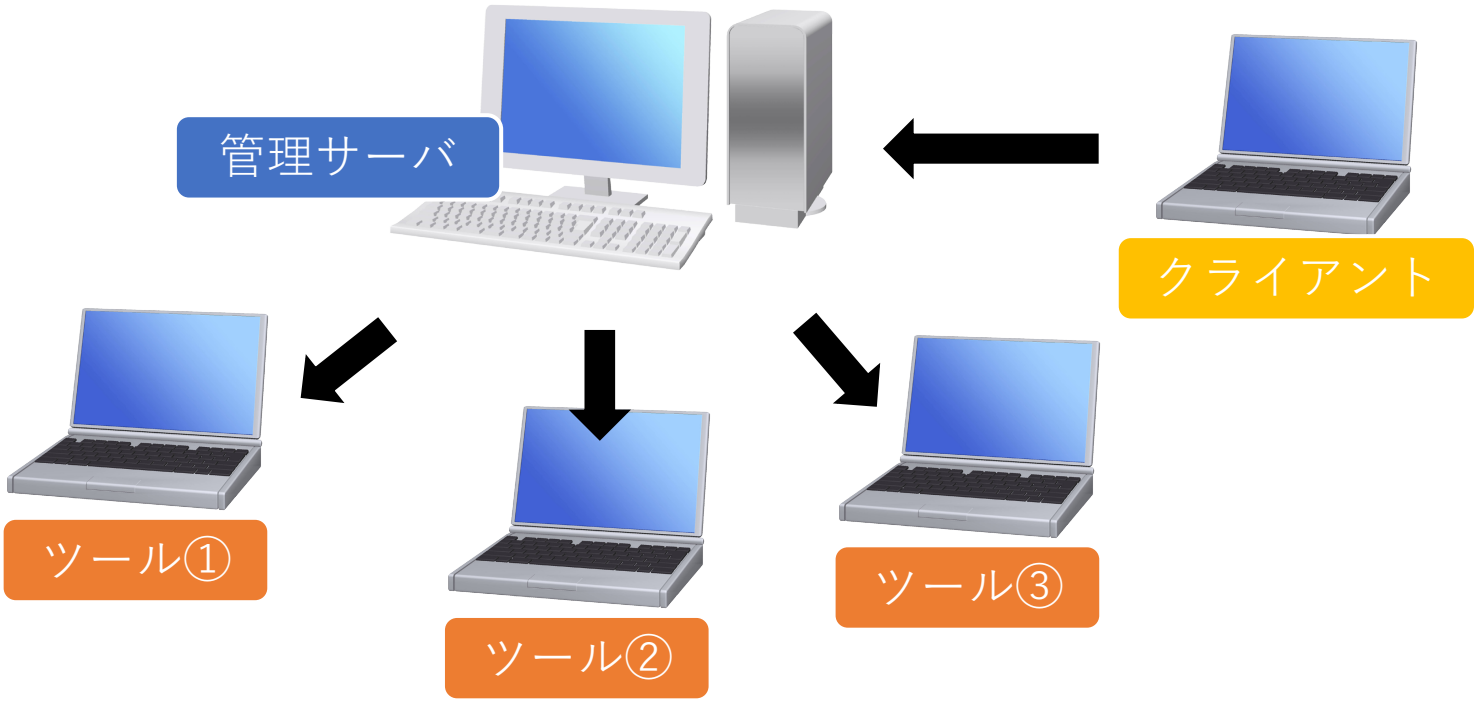
比嘉 築@ECF

軽く自己紹介を

- 比嘉　築（ひが　きずく）
  - 合同会社イー・シー・エフ代表社員
  - プログラマー/SE
  - 技術講師
- 専門
  - Android, Java, Python, …？
  - 物理学(数値計算)
  - 何でもできますよ

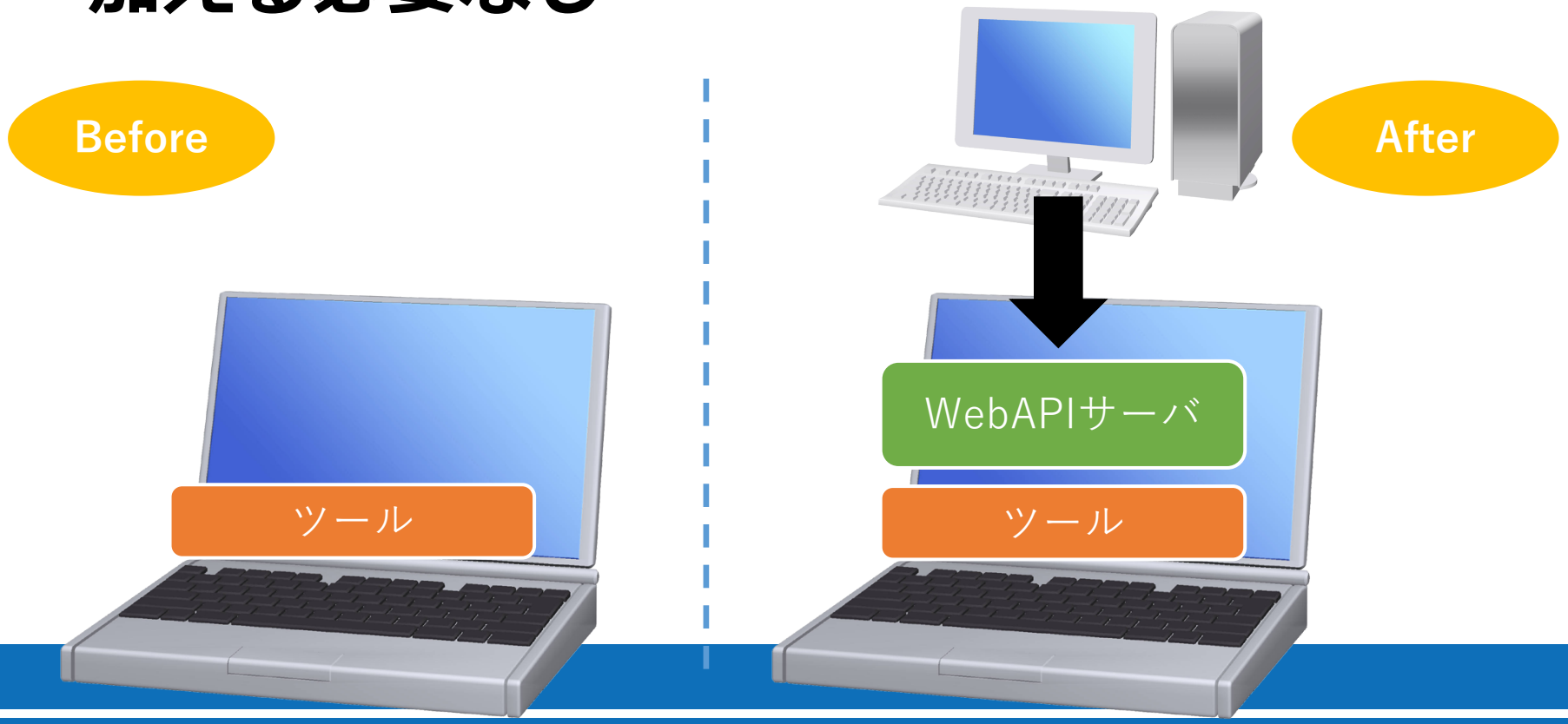
こんな要件が出てきました

- **すでにPC上で動いているツールがある**
- **サーバで複数ツールの一元管理がしたい**



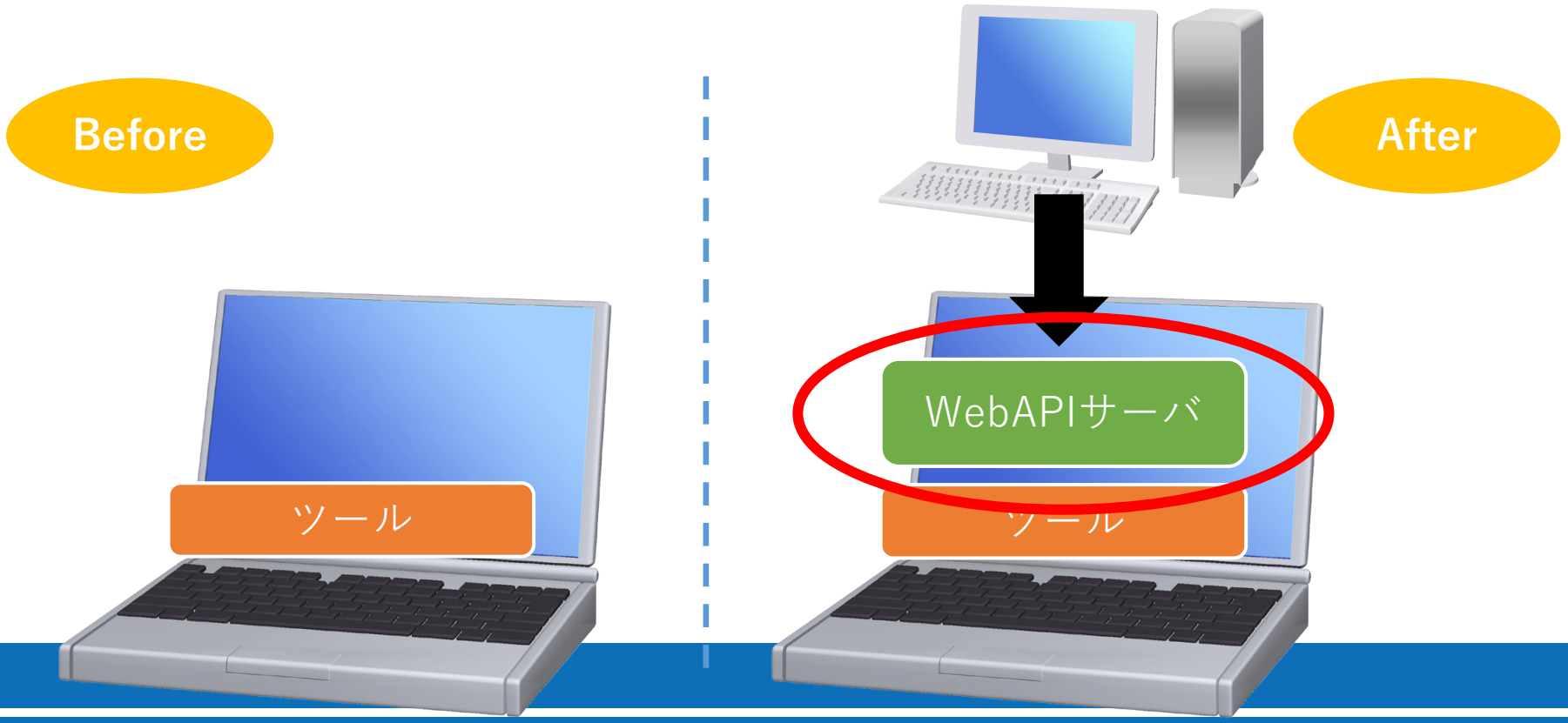
ツールにWebAPIを仕込もう

- **PCにWebAPIサーバ乗っけるだけ**
- **基本的に今動いているツール自体に手を加える必要なし**



ということで

- **このWebAPIサーバ作ってくれない？**
- **プロトタイプ的なものでいいから**



プロトタイプとな？（裏要件）

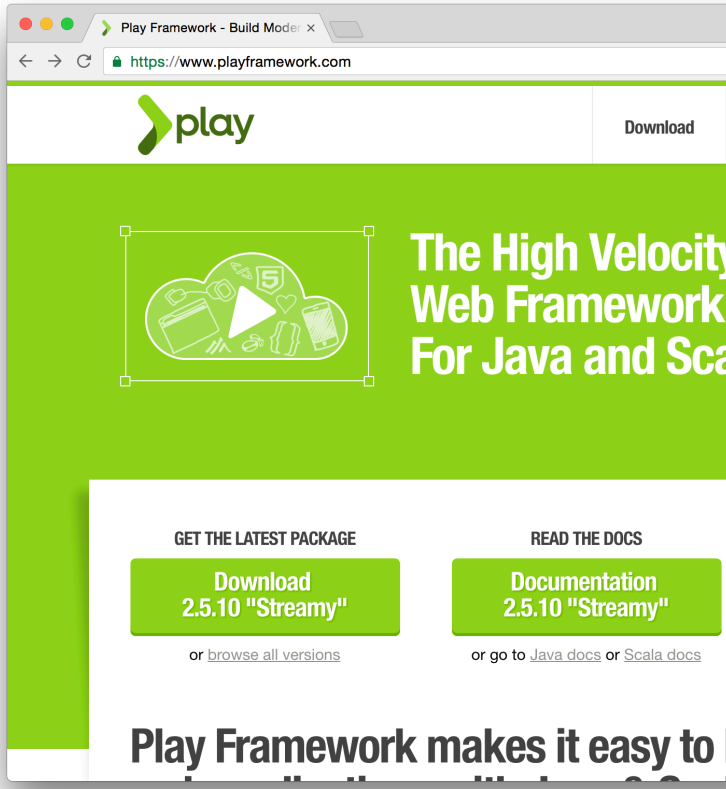
- **現プロジェクトメンバーは全員がWebの経験無し**
  - **CとかC#な人たちの集まり**
  - **少なくとも設計/構築経験は無い**
- **とりあえず動くもの作って、**
- **「引き継げるようにしてくださいね☆」**

検討

- フレームワークは必須だよね
  - 開発速度重視
    - 数週間でデモ稼働させなきゃいけない
      - プロトタイプって言ってたのに・・・
  - 現メンバーの学習コストがかからないよう
- 言語は？
  - やはり学習コストが気になる
    - まあ、並のエンジニアなら何でもいけるはず？
  - でもトラブル（想定外）があると困る
    - スケジュール的に
  - それなら Java でいきましょう

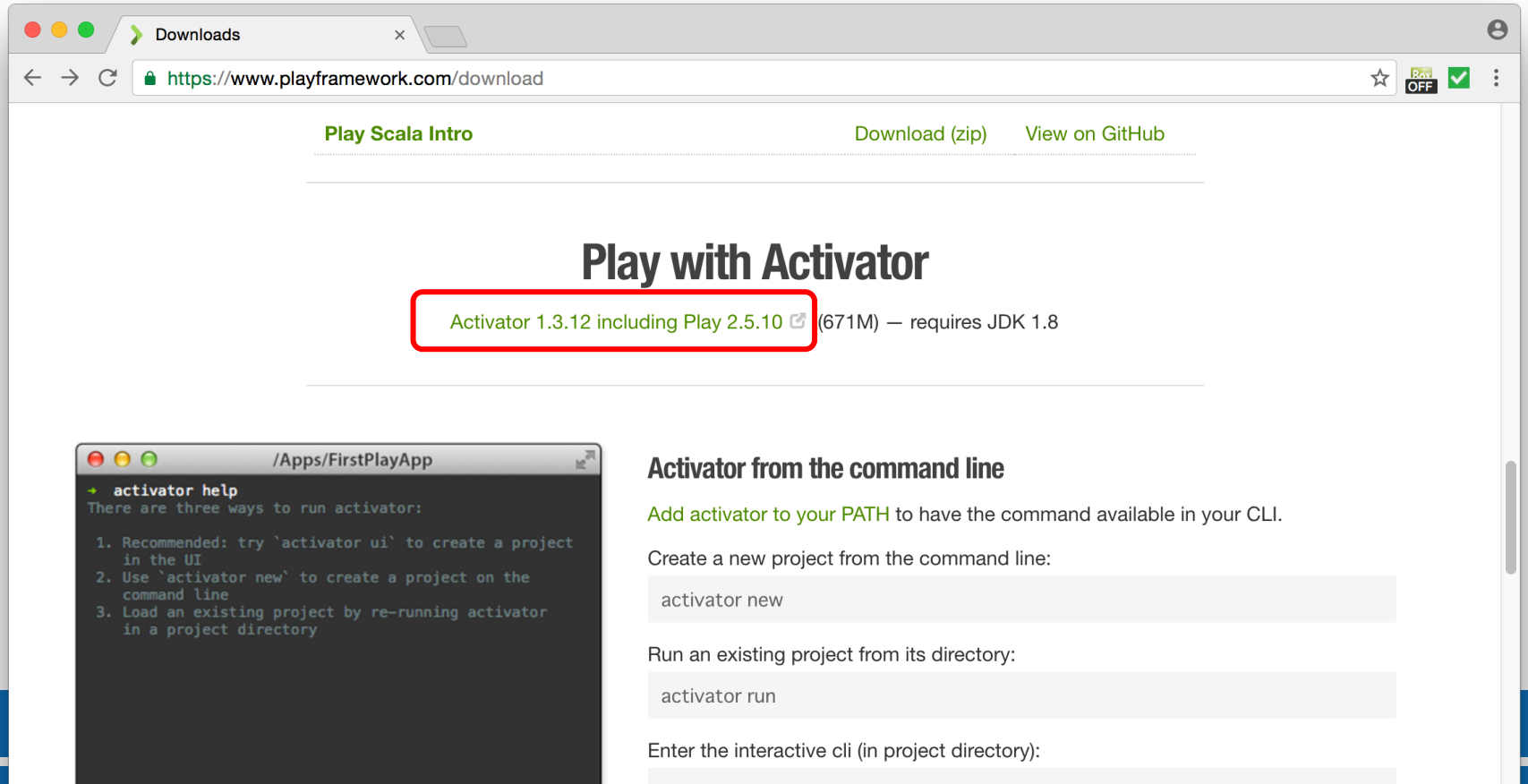
Let's "Play Framework"

- Webアプリケーションフレームワーク
- Java (もしくは Scala)
- よくあるMVCモデル
- Java EE を使っていない
  - 軽量
  - ある意味、前提知識が不要
- Netty(Webサーバ)内蔵
  - この辺もすっとばせる



ユーティリティコマンド操作

- すべては activator コマンドで
  - 製造の開始はこいつの入手から



activator new

- 新規プロジェクト作成
- プロジェクト名やJava or Scalaを選択

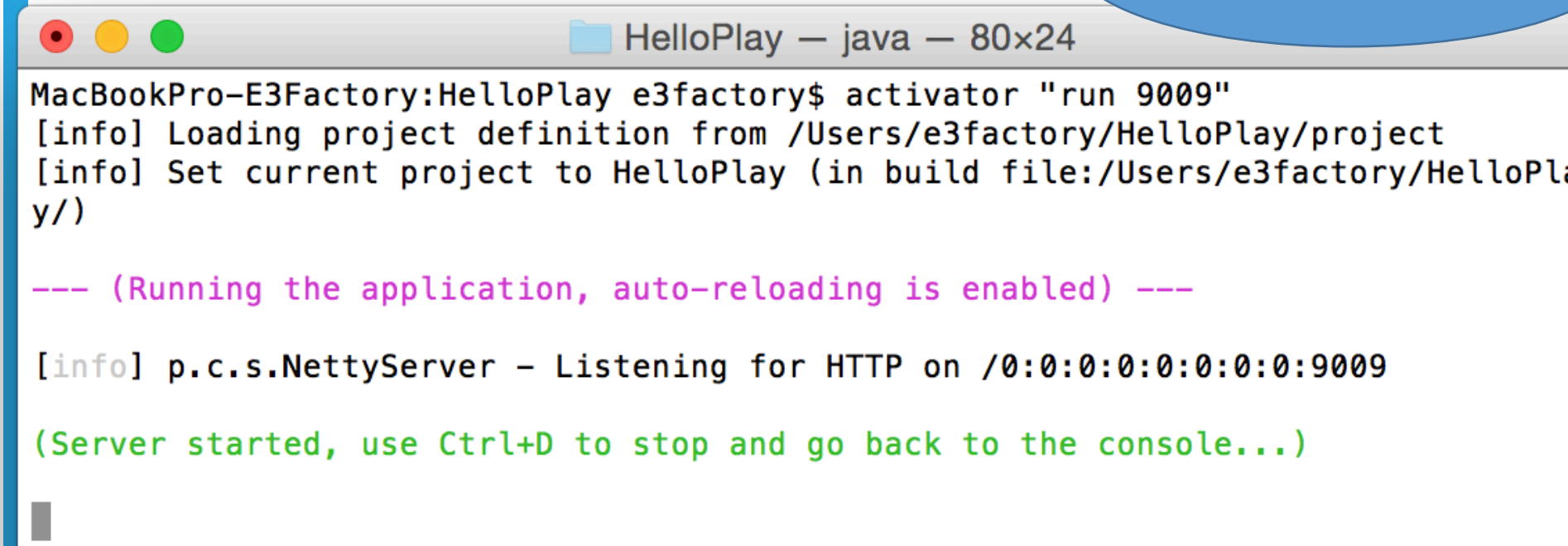
e3factory — java — 80×24

```
MacBookPro-E3Factory:~ e3factory$ activator new

Fetching the latest list of templates...

Browse the list of templates: http://lightbend.com/activator/templates
Choose from these featured templates or enter a template name:
  1) minimal-akka-java-seed
  2) minimal-akka-scala-seed
  3) minimal-java
  4) minimal-scala
  5) play-java
  6) play-scala
(hit tab to see a list of all templates)
> 5
Enter a name for your application (just press enter for 'play-java')
> HelloPlay
```

activator run

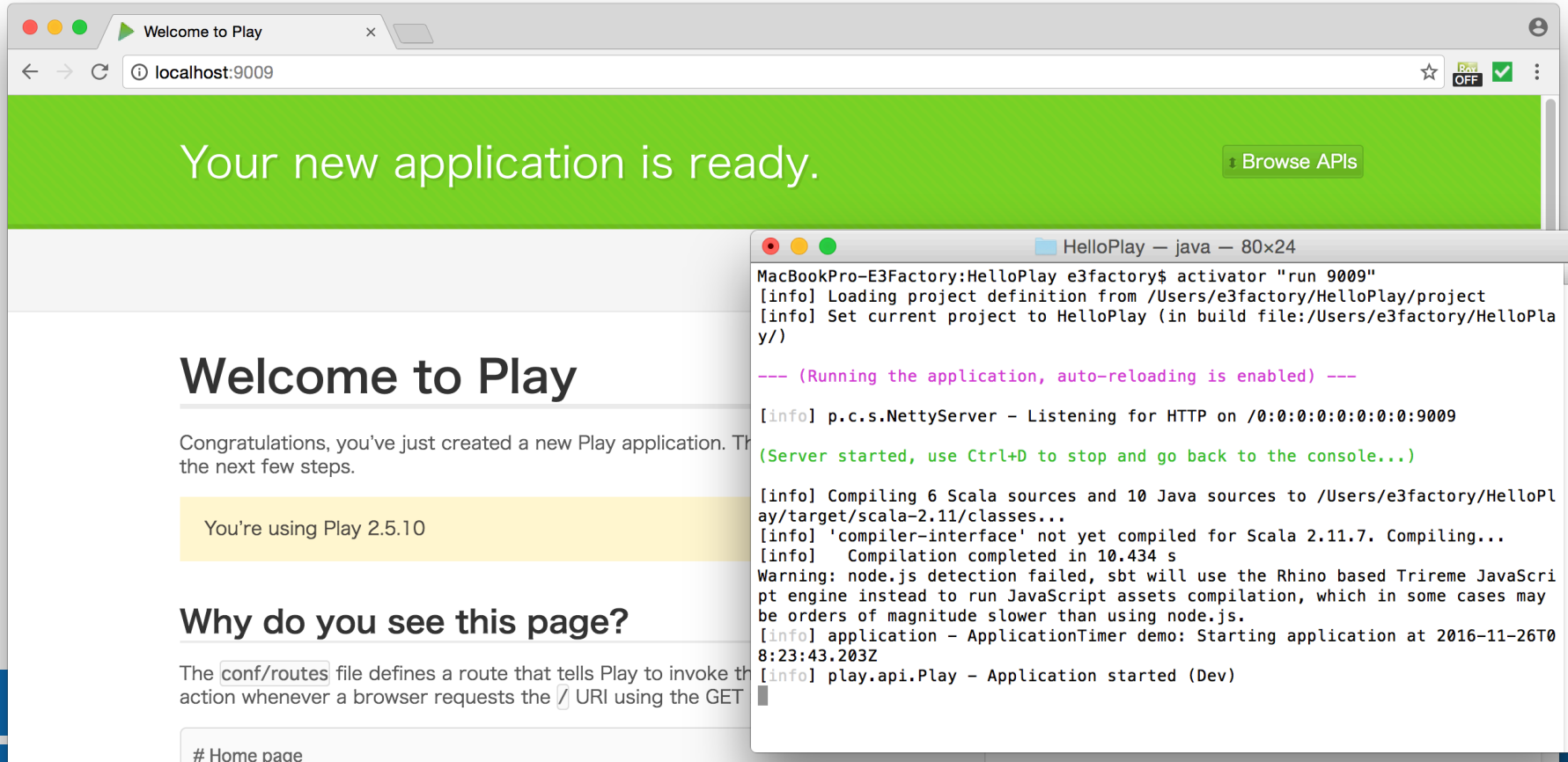
- **サーバ起動**
- **初回は時間かかるので放置**

ポートを指定できる
デフォルトは9000



アクセス確認

- ホットデプロイ
  - ビルドが入るので初回や変更後は遅い

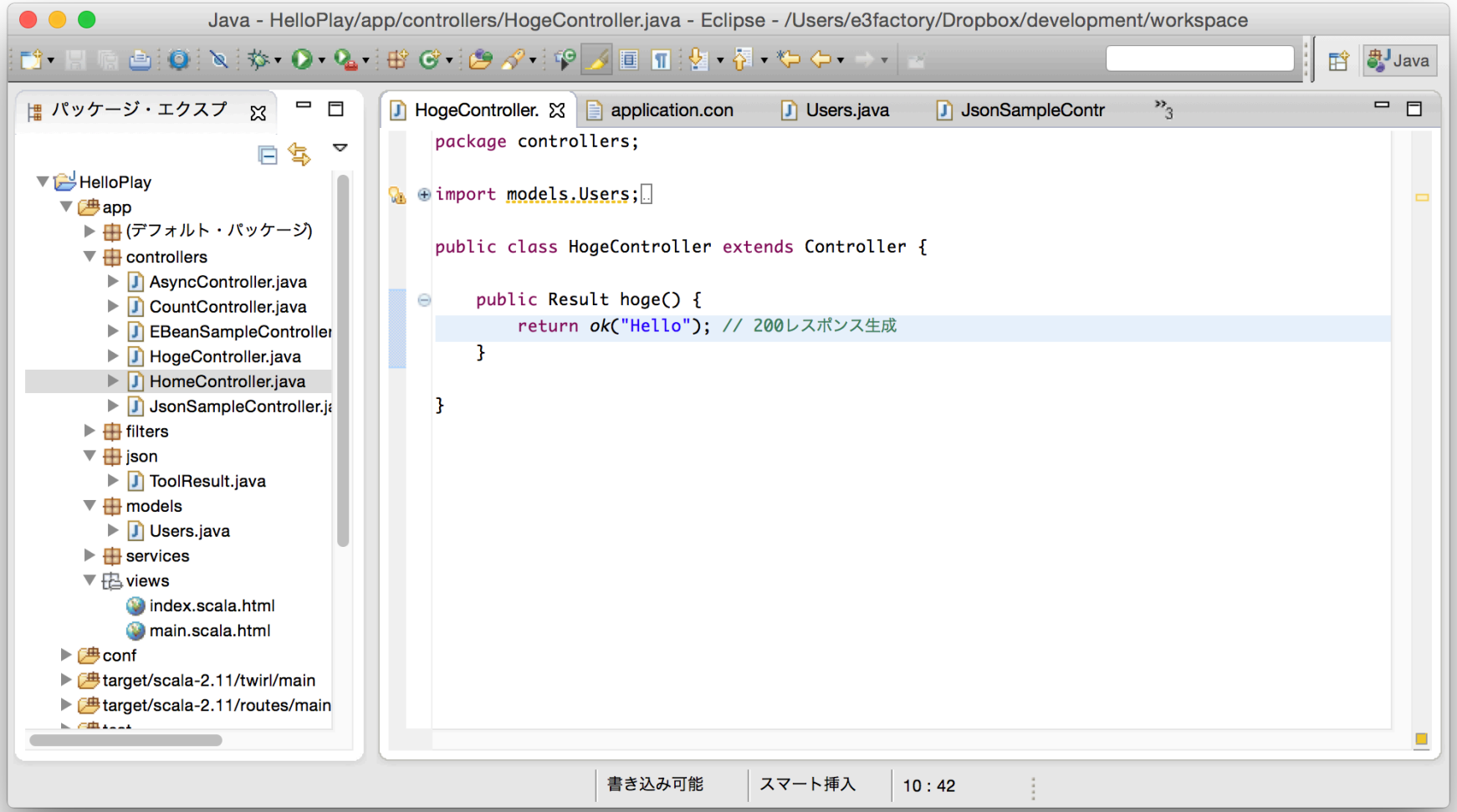


Eclipseで開発

- **各種IDEにも対応可能**

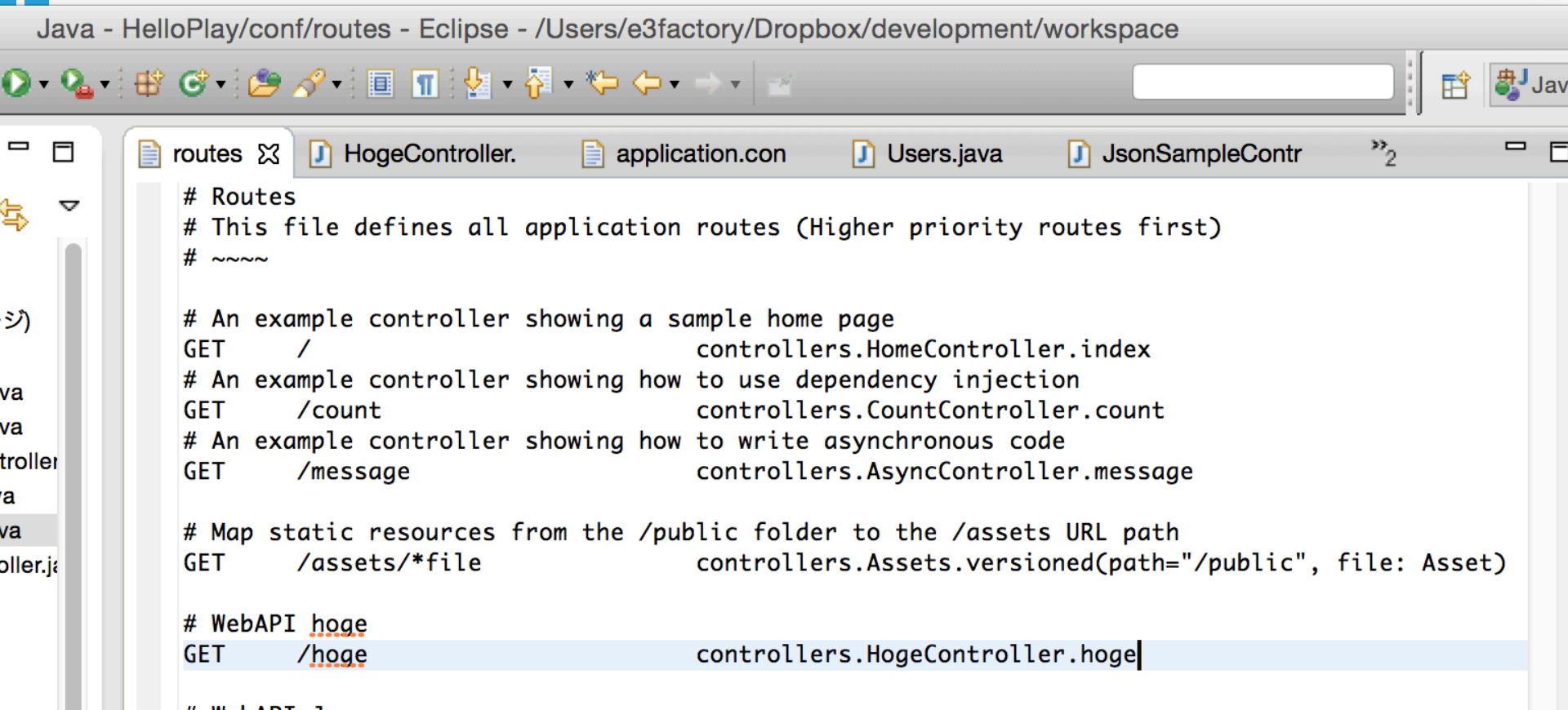
- **例：Eclipse**
  - **関連プラグインを読み込む設定**
  - **「activator eclipse」**
    - **Eclipseプロジェクト関連ファイル生成**
  - **Eclipseでインポート**

Eclipseで開発


Java - HelloPlay/app/controllers/HogeController.java - Eclipse - /Users/e3factory/Dropbox/development/workspace
Java
パッケージ・エクスプ
HogeController.
application.con
Users.java
JsonSampleContr
HelloPlay
app
(デフォルト・パッケージ)
controllers
AsyncController.java
CountController.java
EBeanSampleController
HogeController.java
HomeController.java
JsonSampleController.j
filters
json
ToolResult.java
models
Users.java
services
views
index.scala.html
main.scala.html
conf
target/scala-2.11/twirl/main
target/scala-2.11/routes/main
package controllers;
import models.Users;
public class HogeController extends Controller {
public Result hoge() {
return ok("Hello"); // 200レスポンス生成
}
}
書き込み可能
スマート挿入
10 : 42

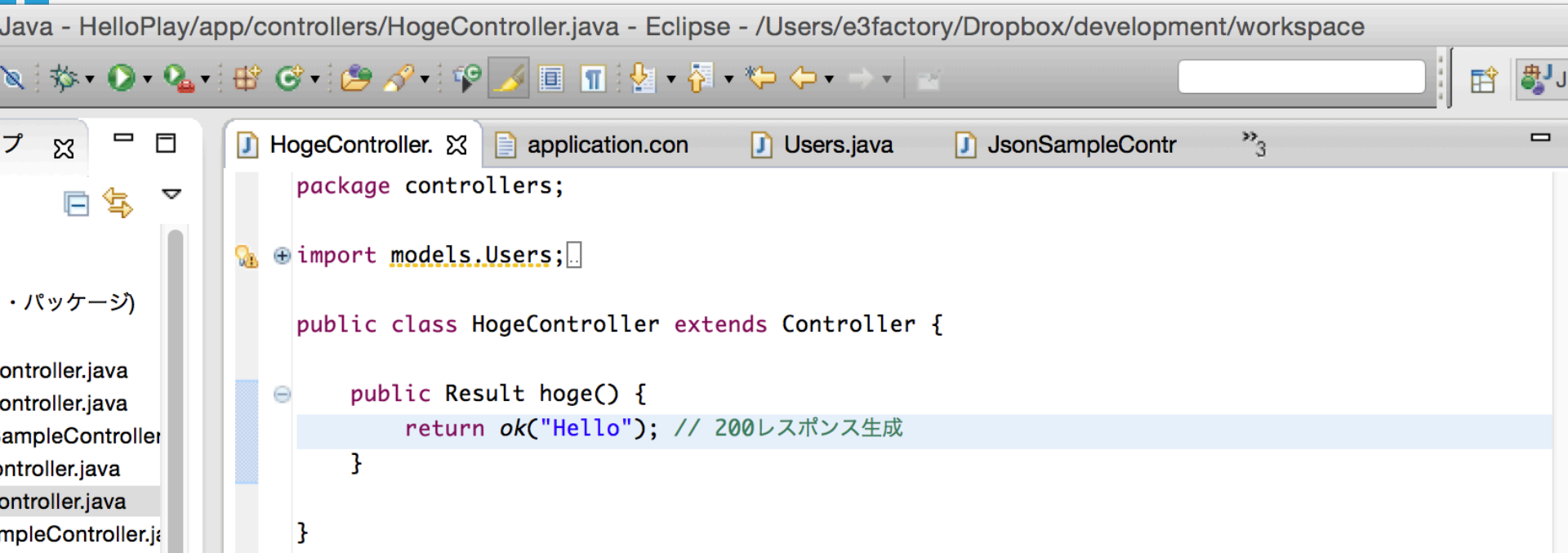
ルーティング

- routesファイル
- URLとメソッドの対応づけ→アクション



Controller

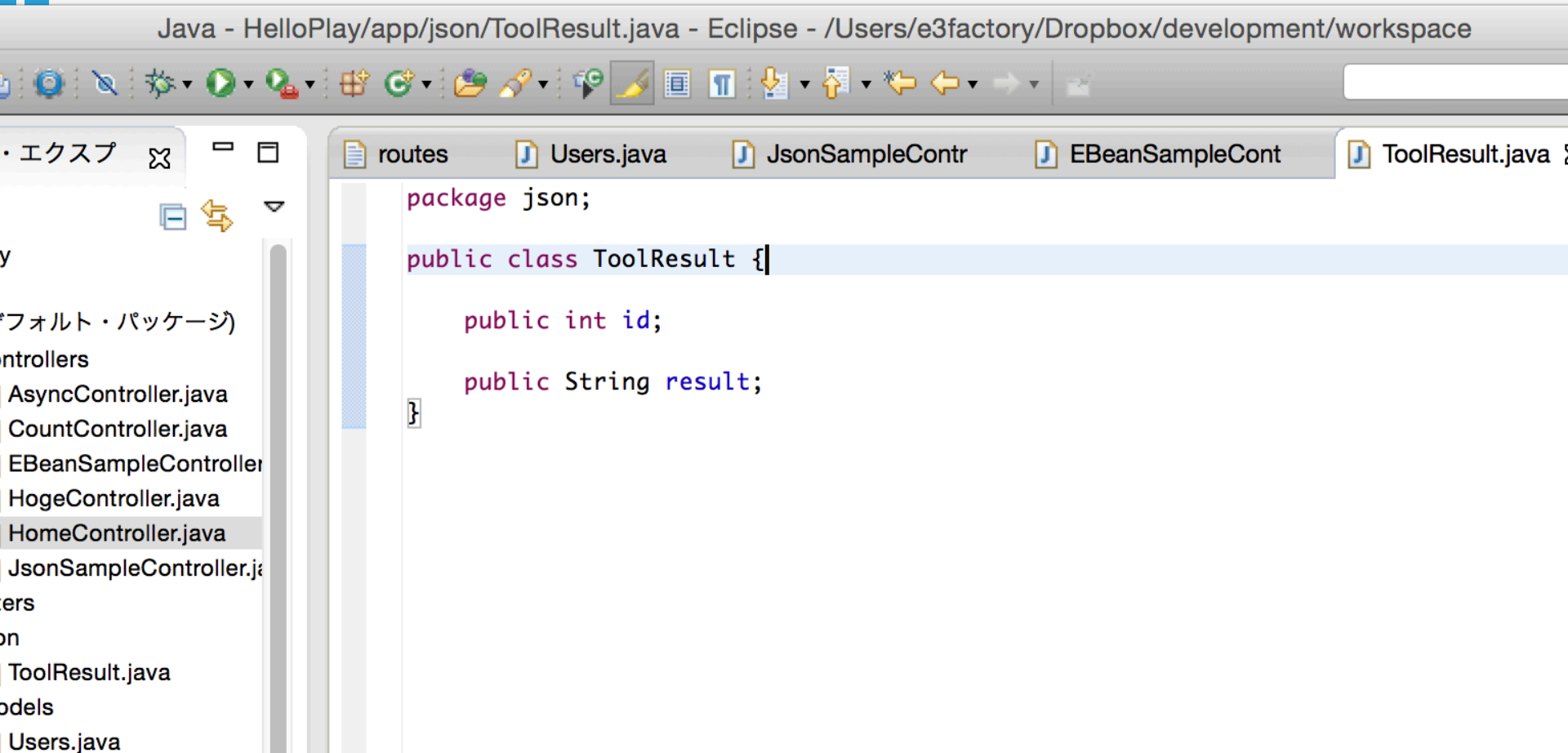
- Controllerクラスを継承したクラス
- routeで設定したアクションと対応するメソッド
  - 戻り値Resultオブジェクト



View

- Scalaベースで記述
  - 基本的な形式なので特に困りはしない
  - 今回の案件では使わなかったので割愛
    - あまりわかっていない
  - JSONがビュー代わり

JSONレスポンス



- **POJOなBeanを作成**

Java - HelloPlay/app/json/ToolResult.java - Eclipse - /Users/e3factory/Dropbox/development/workspace

routes | Users.java | JsonSampleContr | EBeanSampleCont | ToolResult.java

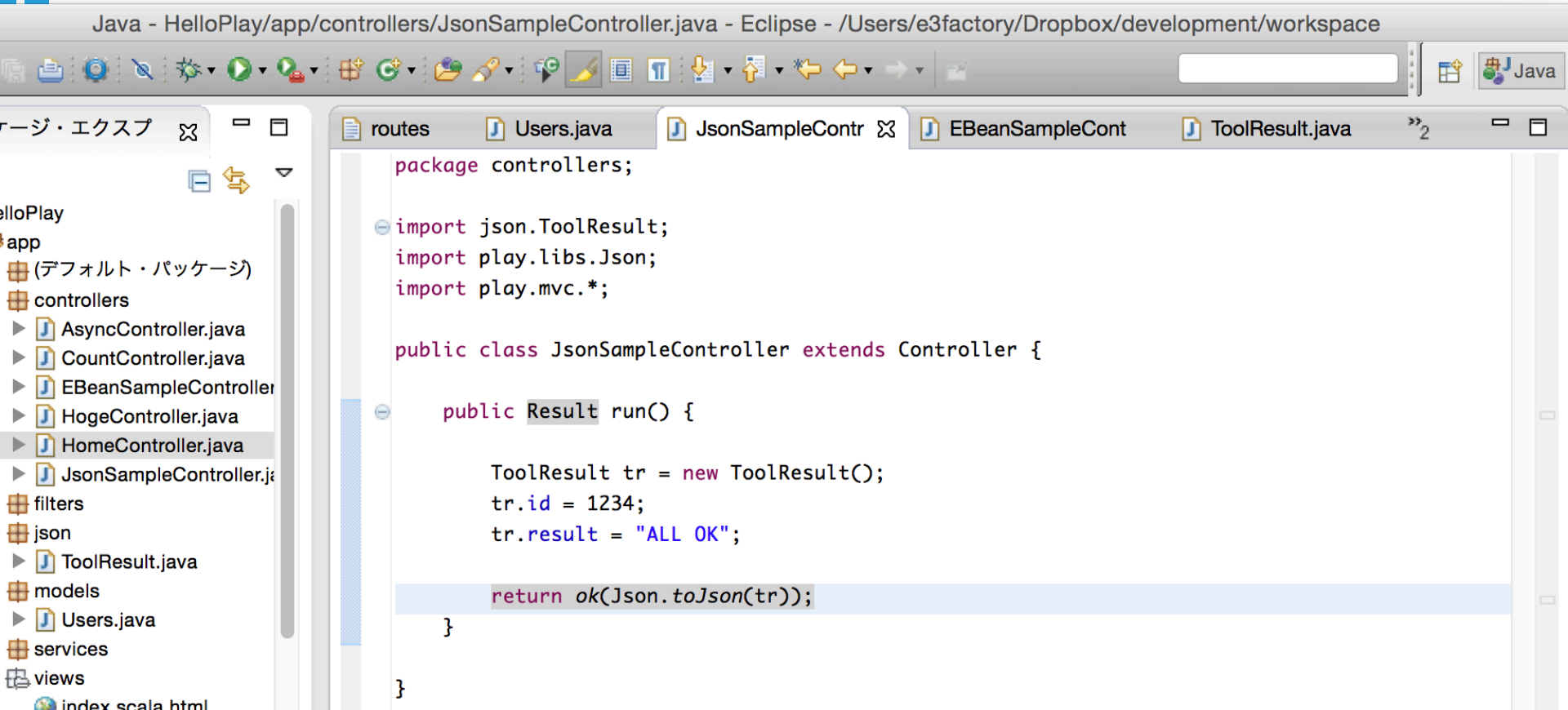
```
package json;

public class ToolResult {

    public int id;

    public String result;
}
```

・エクスプ

フォルト・パッケージ)
ntrollers
AsyncController.java
CountController.java
EBeanSampleControlle
HogeController.java
HomeController.java
JsonSampleController.j
ers
on
ToolResult.java
odels
Users.java

JSONレスポンス

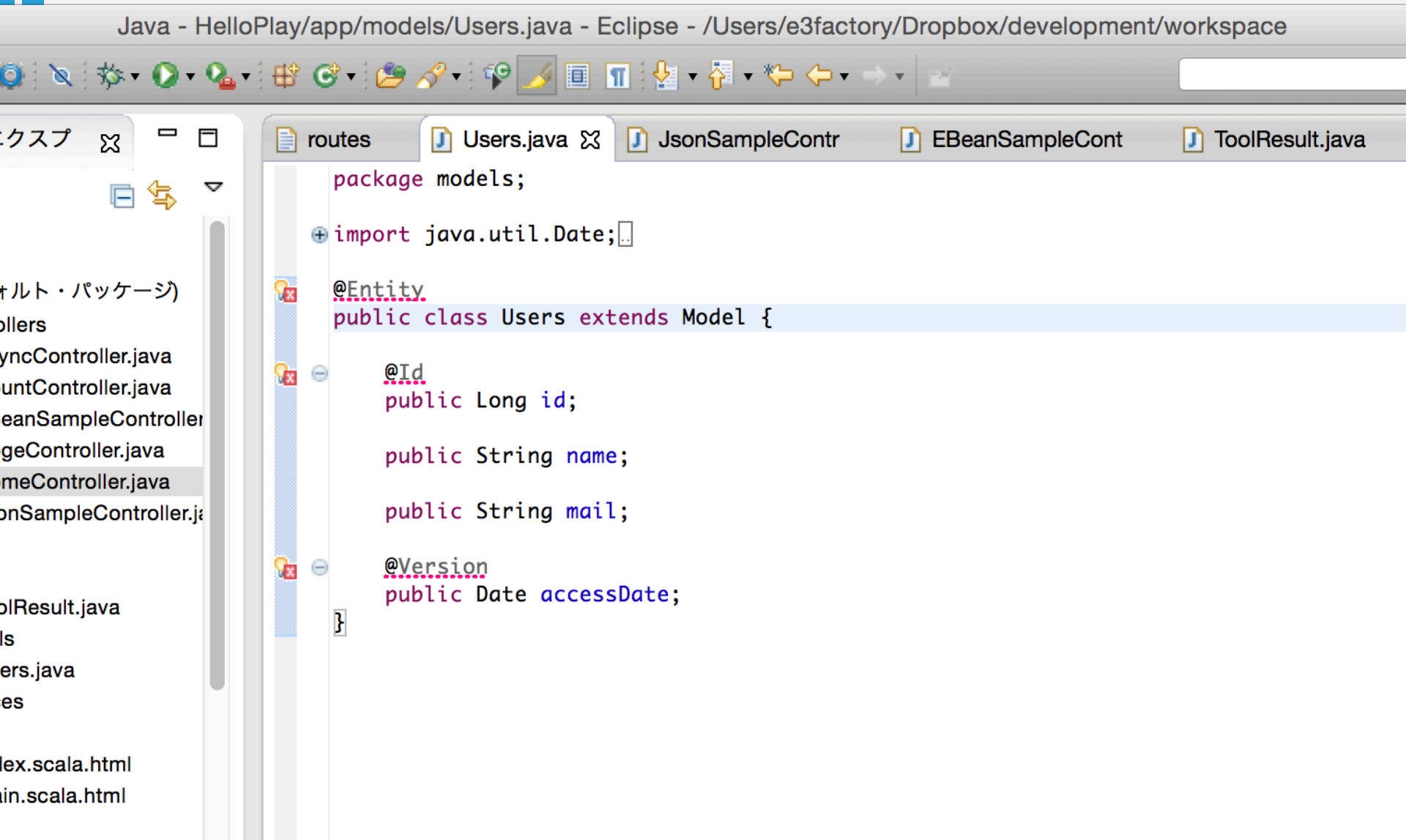
- **Jsonオブジェクトに変換するだけ**
- **Content-Typeを自動で判別して設定**



Model

- 標準ではEBeanを利用
  - JPAっぽい
  - 別のO/Rマッピングを使うことも可能
  - DB関連設定は application.conf へ記載
- Modelクラスを継承したクラスを作成
  - テーブル/カラムに対応したBean
  - アノテーションでDB関連属性を指定
- Modelに変更があると自動的にDB変更
  - Evolution機能

Model


Java - HelloPlay/app/models/Users.java - Eclipse - /Users/e3factory/Dropbox/development/workspace
routes
Users.java
JsonSampleContr
EBeanSampleCont
ToolResult.java
package models;
import java.util.Date;
@Entity
public class Users extends Model {
@Id
public Long id;
public String name;
public String mail;
@Version
public Date accessDate;
}

感想

- WebAPI程度ならすんなり
  - エラーレスポンスやCORS対応など問題なくフレームワーク内で吸収可能
- ドキュメントは思ったより豊富な印象
  - 公式ドキュメントが丁寧
  - ただし一般の情報はScalaばかり
- 「Playを使いたいからScalaを勉強」はありだと思う
  - Javaでは理由がない限りどうだろう？