Future Office
Coordinate Service

次世代Webアプリ方式

脱MVC、軽量実装でスピード開発

背景

■レガシー世代の疑問

- Ajail、確かに早い　…でも、ドキュメントは？引継ぎは大丈夫？
- Webアプリ、MVCでの開発遅れ　…画面はもっと早くできないの？

■技術面の不安

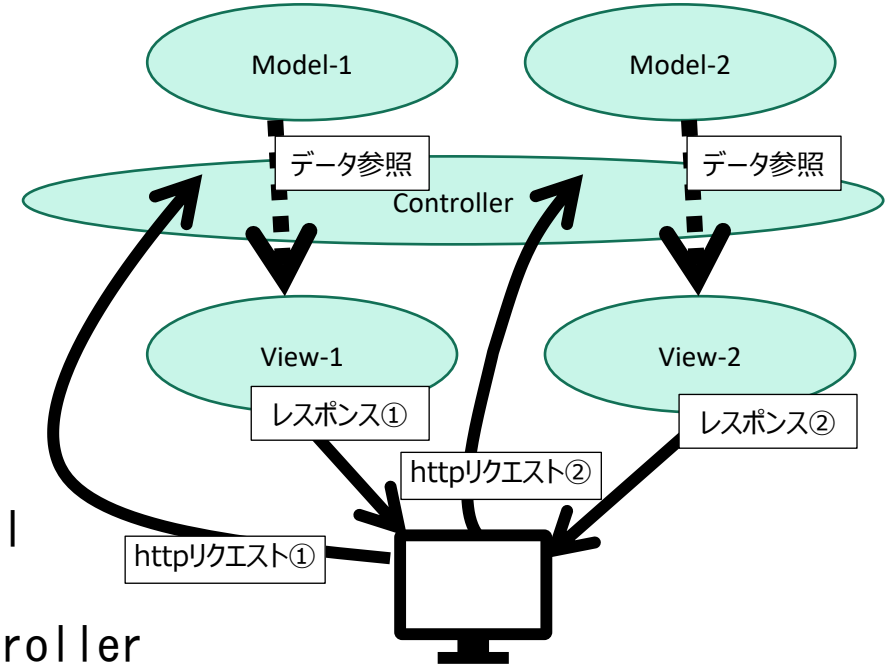
- JavaはJDK1.8以降どうなる？ ブームのPythonにする？

■悪いことばかりでもない

- JavaScriptはESMAScript2016以降、ブラウザ差異が気にならない

機能（ReactとJSONを使ったアプリ方式）

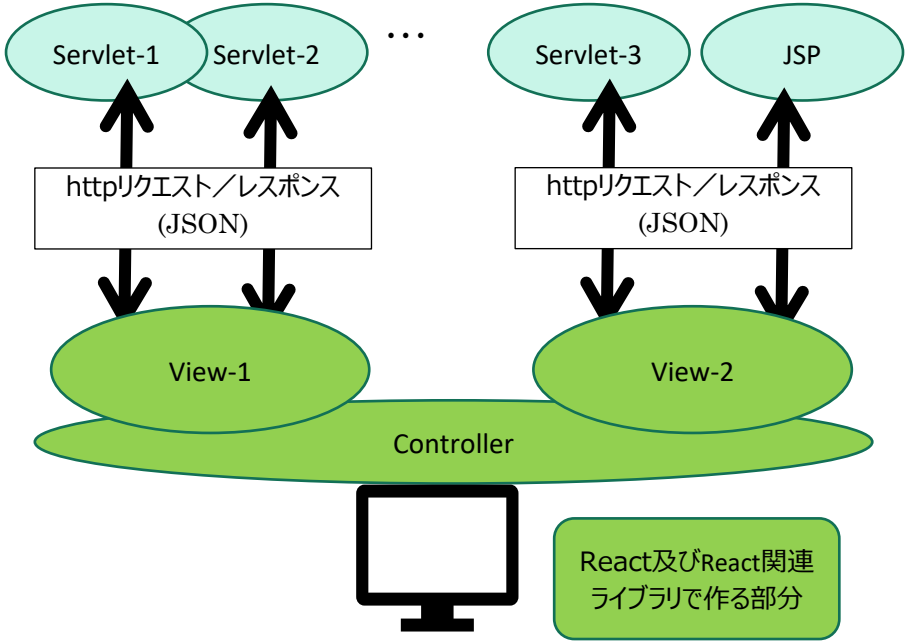
一般的なMVC*1によるWebアプリ



*1
M:Model
V:View
C:Controller

・M･V･Cの全てがサーバ内で密結合している
・ModelとViewは１：１で入力チェック／更新／出力の全てを行う

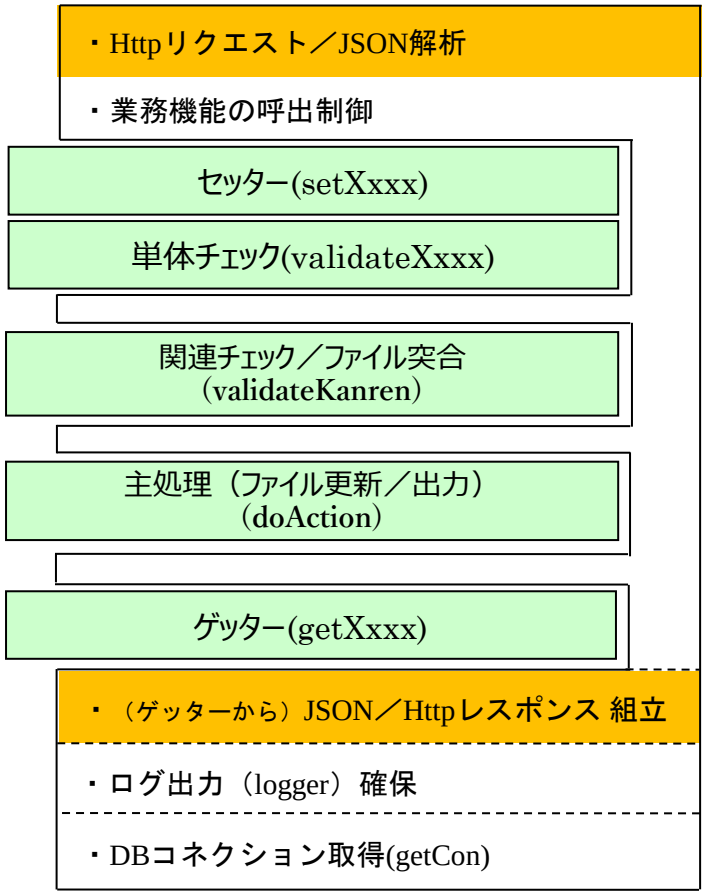
VCにReactを使ったWebアプリ



・ViewはReactのコンポーネントを使って作る
・View及びControllerは、クライアントとして動作する
・Servlet/JSPはMVCのModelを使いやすい粒度に分割し、ajaxで呼び出す
・Http/JSONをインタフェースに使える相手（Excel等）全てクライアントになり得る

機能（加えて、再利用可能な実装）

Servletの基底クラス＜実装例＞



凡例　各Servletで実装が必要な部分
（処理が不要なら宣言を省略してもよい）

この実装例は、業務固有の処理だけを個別のメソッドとして定義し、Javaのリフレクションを使って基底クラスから呼び出す。

これにより業務固有部分だけを定義したクラスが作られ再利用がし易くなる。

効果

№	観点	利点
1	ドキュメント（保守性）	・サーバ側の実装（Servlet、JSP）が入力、出力、機能単位になり分かり易い
2	画面	・（特に複雑な画面は）デザイナーがHTMLを完成させるのを待ってAP開発者がタグを埋め込んでJSPやxhtmlにしていたが、インタフェースのJSON決定以降は並行して作業ができる
3	試験	・実際の画面がすぐ表示できる ・サーバ側はcurlコマンド等を使って簡易に試験ができる
4	拡張性	・画面に出力するデータを別のツールで取得して再利用する等の拡張が容易