

Web 画面のデータ収集とデータ表示用 DSL の実現

ー Ajax フレームワークにおける目的指向の模索ー

常珂軍 (エスコ・ジャパン株式会社)

概要 SI 業界が抱えている課題の一つに、Web システムのフロント画面作成の生産性の向上がある。著者が所属するエスコ・ジャパン株式会社ではシステムのフレームワークの改善のために、データ収集とデータ表示の自動化を実現するための目的指向 DSL (domain-specific language) を試作した。本レポートではこの手法を報告するとともに実際にこれを使った運用結果を分析し、有用性について考察する。

1. はじめに

Web システム開発において、フロント画面の見栄えと操作性の実現のために相当な工数がかかっている。そこで、システムのフレームワークを改善し生産効率を向上させることは SI 会社にとって課題の 1 つになっている。

この生産性についての課題を解決するための方向性として、「画面自動作成」、「ソース自動作成」などがある。すなわち、画面項目を DB 項目を参照してレイアウトを組み立てて属性を設定する。また必要に応じてソースを作成してカスタマイズに備える。ただし、「プログラムしない」あるいは「決めたプログラムしかできない」という短所があり、現実世界に存在するさまざまな画面表現の要求を満足しがたいため、生産性のためにシステムの柔軟性まで損傷するという理由で普及まで行かないといった背景がある。

ここで、一般的な Web システムは Web サーバと DB サーバで構成され、データフローにより複数層に分けられる。図 1 は MVC (Model View Controller) 構造の関連図である。

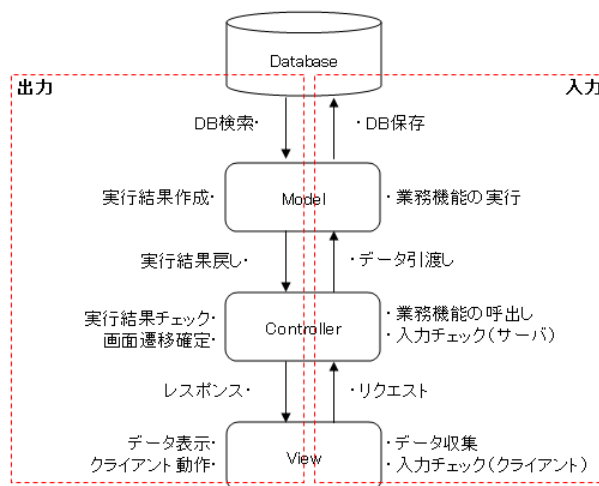


図 1 MVC 構造

MVC 構造によく現れる識別子を列挙すると表 1 のようになる。

表 1 MVC 構造に利用される識別子

場所	識別子	用途とよくある命名方法	状況の変化
クライアント	画面項目	Name 属性または Id 属性は表示される DB 項目の名称を参考する。	JQuery のセレクターを利用する場合、HTML タグのデザイン属性も識別に利用できる。
	Javascript 変数	データを取得される画面項目の名称を参考する。	
WEB サーバ	リクエストパラメータ	送信されるフォーム項目の Name 属性名になる。	フレームワークでリクエストと Bean を自動マッピングを行う仕組みが存在する。
	リクエスト属性	格納される Bean のクラス名を参考する。または用途に合わせて命名する。	
	Bean 項目	DB 項目または画面項目に関連する名称を利用する。	

表 1 より、Web システム開発には「同じデータが複数の識別子で定義される」という問題点があることが見てとれる。計算複雑性理論により、処理機に対して必要な記憶容量（識別子）が多ければアルゴリズムが複雑になる。よって、識別子を減らせばシステムの複雑度を下げることができるので、生産性向上に繋がることになる。「画面自動作成」の方法はすべての識別子を DB 項目に集約しシステムの複雑度を極力減らすため、生産性向上を実現していると理解できる。また、その方法には、画面表現の識別子が存在しないため、さまざまな画面表現の要求を実現しがたいことも容易に結論付けられる。

そして、DB 項目を除くほかの識別子を画面表現のために統合させば、柔軟性を保ったまま複雑度を減らす方法になるのではないかと考えられる。この考えに従って、さらに取扱い簡単なフレームワークを開発するため、jQuery とサーバサイド Javascript (javax.script) を用いてデータ収集とデータ表示をパターン化し、目的指向の専用言語(DSL)を提案する。

2. データ収集自動化の Ajax 仕組みと DSL

従来の WEB 開発においては、イベント処理の送信データはクライアント側とサーバ側に両方定義する必要があった（識別子の命名など）。もし定義情報を片方に集中できればフレームワークの簡略化に繋がる。そして jQuery のセレクターの適用範囲を広げてサーバからも利用できるようにするため図 2 の仕組みを設計した。※定義情報をクライアントに集中することも考えられるが、セキュリティの懸念で採用していない。

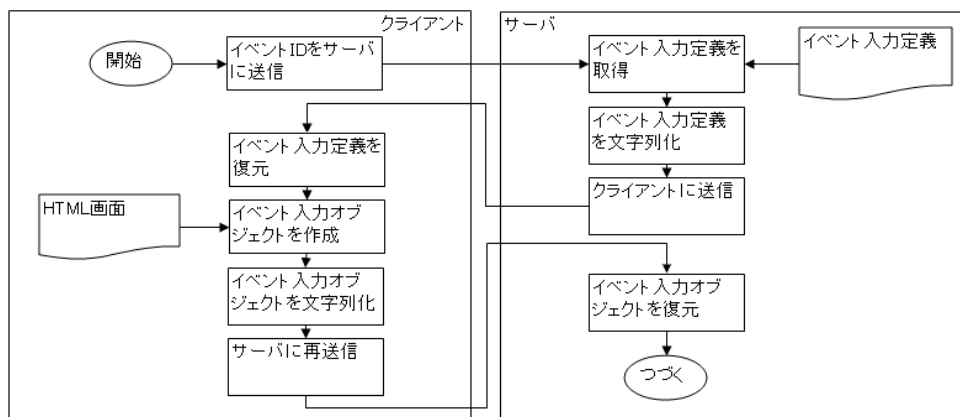


図 2 データ収集自動化

1 回のイベント処理に 2 回のサーバ送信を行うことが特徴である。1 回目の送信でクライアントはイベント実行開始の情報をサーバへ送信し、イベント入力定義を受信する。2 回目の送信でクライアントはイベント入力定義により入力オブジェクトを作成して、サーバへ再送信する。

著者らの実践では、2 回送信は JQuery の Ajax 機能を利用して 1 回目送信の success コールバックに 2 回目の送信を行う形で実現している。イベント入力定義は JSON オブジェクトで行う。JSON オブジェクトの stringify 関数で文字列化を行い、parse 関数で復元する。

JSON オブジェクトの属性は、[単純値|サブ JSON |配列<単純値>|配列<サブ JSON>] の値を持つことが可能である。イベント定義情報として利用する場合、属性の値は [null|サブ定義 |配列<サブ定義>] に制限する。イベント定義情報は、各 null の値をデータで交換すればイベント入力オブジェクトに変わる。表 2 はイベント入力定義における各種属性の説明する。

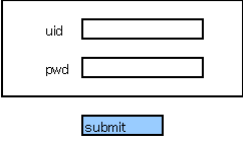
表 2 JSON のイベント入力定義の属性説明

属性種類	属性名称の用途	正常ケース	異常ケース
selector : null	単独な入力データを識別する。	属性名称をセレクタとして HTML タグを 1 つ取得する。タグの value, text などの属性は入力データと利用する	タグを複数取得する場合エラー
selector : {...}	サブ入力オブジェクトを識別する。	属性名称をセレクタとして、HTML タグを 1 つ取得する。そのタグをサブ定義処理時のコンテキストにする。	タグを複数取得する場合エラー
selector : [{...}]	サブ入力オブジェクトの配列を識別する。	属性名称をセレクタとして、HTML タグを取得する。そのタグをサブ定義配列処理時のコンテキストにする。	—

※selector の書き方は JQuery のセレクタ尾の語法に従う。

表 3 は例で説明する。

表 3 イベント入力定義の例

画面	HTML コード
	<pre> <fieldset id="fst_login"> uid <input type="text" id="txt_id"> pwd <input type="password" id="txt_pwd"> </fieldset> <input type="button" value="submit"> </pre>
id を識別子と利用する。 ※項目 txt_id と項目 txt_pwd から入力データを取得	{ "#txt_id" : null , "#txt_pwd" : null }
type 属性を識別子と利用する。 ※text タイプと password タイプの項目から入力データを取得。	{ ":text" : null , ":password" : null }
タグ名と順番を識別子と利用する。 ※1 番と 2 番の input タグの項目から入力データを取得。	{ "input:eq(0)" : null , "input:eq(1)" : null }
サブ定義を利用する。 ※項目 fst_login に含む項目 txt_id と項目 txt_pwd を取得。	{ "#fst_login" : { "#txt_id" : null , "#txt_pwd" : null } }
サブ定義の配列を利用する。 ※項目 fst_login に含む input タグの項目を配列として取得。	{ "#fst_login" : [{ "input" : null }] }

そして、クライアント側に js ライブラリを設けて 1 回目送信結果の入力定義情報の属性名称により HTML のタグ項目を特定してデータを取得し、そのデータを入力オブジェクトに変換して 2 回目送信を行うときの一連の動作は自動的に行う。

データ収集の自動化は従来方式よりデータ入力処理の中間ステップ削減を実現でき、目的指向の特徴となっている。また、上記 JQuery セレクタ+JSON の表現方法は収集したいデータを効率よく表現できるため、データ収集の専用言語 (DSL) として WEB 開発に活用できる。

3. データ表示自動化の Ajax 仕組みと DSL

従来の WEB 開発においては、画面モジュール (jsp など) に表示場所をサーバタグで標記してサーバ側のリクエストと Bean などに表示データを格納し、画面モジュールの実行により表示場所と表示データがマッピングされる。

この方法では定義される識別子が多く、システムに点在する状況となっている。もしサーバ側から表示データと表示場所をセットで画面モジュールに渡せば、フレームワークの簡略化に繋がる。2 節と同様にして、JQuery のセレクタキーの適用範囲を広げる考え方で図 3 の仕組みを設計した。

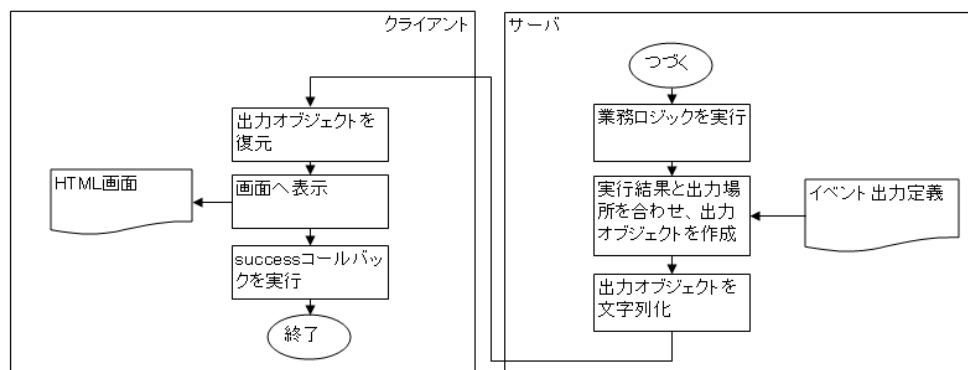


図 3 データ表示自動化

実行結果と出力場所をセットで格納することが特徴である．サーバ側は実行結果をイベント出力定義に代入し，出力オブジェクトを作成してクライアントに返信する．クライアント側は出力オブジェクトを受け取って出力場所の情報により結果データを画面に表示する．

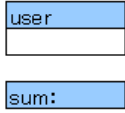
筆者らの実践では，イベント出力定義はイベント入力定義と同じ JSON オブジェクトで行なった．また，出力オブジェクトは前述の 2 回目送信の受信結果になる．イベント出力定義を整理すると表 4 のようになる．

表 4 イベント出力定義の属性説明

属性種類	属性値の用途	必須
runat : selector	属性値をセレクタとして，HTML タグを取得する．そのタグは出力定義の他の属性処理時のコンテキストとする．	必須
withdata : { ... }	runat の範囲にいくつの個別タグとマッピングするセレクタと値のセットを格納する．タグの value あるいは text を値で置換する．	—
remove : selector	runat の範囲に属性値をセレクタとして，複数のタグを取得する．取得するタグを削除する．	—
append : html マスク	runat の範囲に配列の withdata 件数毎に html マスクの置換項目を置換して画面パーツとして，runat の後ろに追加する．	—
withdata : [{ ... }]	append と連携する．html マスクの置換項目と値のセットの配列を格納する．	—

表 5 は例で説明する．

表 5 イベント出力定義の例

画面	HTML コード
	<pre><table id="tb1"> <tr><th>user</th></tr> <tr class="data"><td></td></tr> </table> <table id="tb2"> <tr><th>sum:</th></tr> </table></pre>
一覧に 3 行を表示する．内容は one two three．合計部分に 3 の数字を表示する．	
<pre>[{ runat : "#tb1", remove : "tr.data", append : "<tr class='data'><td>{nm}</td></tr>" withdata : [{ "nm" : "one" }, { "nm" : "two" }, { "nm" : "three" }], }, { runat : "#tb2", withdata : { "span" : 3 } }]</pre>	

4. データ収集とデータ表示 DSL の運用効果

上記仕組みを利用する場合、データ収集とデータ表示のクライアントプログラムが大幅に削減される。また、サーバ側とクライアント側の開発言語が Javascript に統一するから、言語間のデータ変換プログラムは必要なくなり、一般開発者に対応する知識要求も(Javascript, JQuery, Java, SQL)から(Javascript, JQuery, SQL)まで縮小する。サーバサイド Javascript (javadoc) が Java と連携しやすいから、Javascript で処理しがたい部分 (例えばファイルアップロード、ファイル操作など) は Java に任せることはでき、システムの拡張性に損がない。

運用効果を数字で表すため上記 DSL を利用する案件とほかのいくつかの案件と比較を行った。表 6 は対象案件の情報である。類似業務の案件だったらベストだが、上記仕組みがまだ若いので、選べる事例が少ない。

表 6 比較案件の特徴

案件	フレームワーク特徴	SQL 使い方
	案件説明	
案件 1	jsp Bean	インライン
	損害保険の BPM(ビジネスプロセス・マネジメント)システム。フローチャートの作成、検索閲覧、EXCEL 出力、バージョン管理などの機能。	
案件 2	jsp Bean	インライン
	郵政関連の苦情管理システム。苦情登録、検索閲覧、統計、画面出力などの機能。画面出力機能にスクラッチでグルーピング改ページなどを実現している。	
案件 3	Ajax Spring	XML ファイルで外出し
	損害保険の試験システム。業務ロジックを BRMS (Business Rule Management System) で分流。加入設計、保険料算出などの機能。	
案件 4	Ajax Spring	XML ファイルで外出し
	案件 3 の再構築。異動管理機能を追加。異動管理の実現は複雑な java プログラムになっている。	
案件 5	Spring	プロシージャ
	物流企業の倉庫管理システム。入出庫、在庫管理、PDF 出力などの機能。	
案件 6	Ajax サーバサイト javascript 上記 DSL	XML ファイルで外出し
	建設機械メーカーの予算稟議システム。ワークフロー、組織ユーザの基幹連携、PDF 出力、監査用ダウンロードなどの機能。	

各案件のボリュームを表す統計情報を表 7 に示す。SQL 数は、select insert update delete のキーワードの数である。プログラムファイル数と行数を統計する際は自動作成に近い Bean のプログラムを除いて、外出しの SQL ファイルを含むものとする。

表 7 比較案件のボリューム情報

案件	ファイル数	プログラム行数	SQL 数	IF FOR 数
案件 1	145	27012	234	2292
案件 2	152	33615	114	2018
案件 3	142	18952	130	1179

案件 4	481	79038	314	5573
案件 5	780	209680	1711	15188
案件 6	198	15020	240	671

各案件のプログラム複雑さを表す統計情報を表 8 に表す．平均複雑度はツール understand で取得している．

表 8 比較案件のプログラム複雑さ情報

案件	平均複雑度	プログラム行数/ ファイル	プログラム行数/ SQL	プログラム行数/ IF FOR のパーセント
案件 1	3.20	186.29	115.44	8.49%
案件 2	5.16	221.15	294.87	6.00%
案件 3	2.30	133.46	145.78	6.22%
案件 4	2.76	164.32	251.71	7.05%
案件 5	3.26	268.82	122.55	7.24%
案件 6	1.94	75.86	62.58	4.47%

表 8 の情報を図 4 のチャートで表現する．

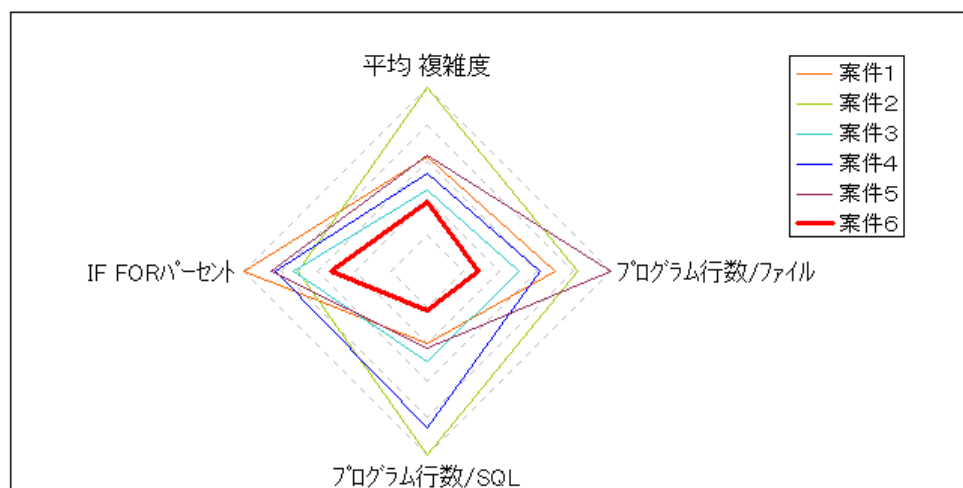


図 4

図 4 から見てとれるように，案件 6 は「仕組みが複雑ではなく，SQL の取扱いが便利，モジュールが小さい」といった特徴を持つといえる．

SQL 数を比較する場合，案件 6 は全案件 1～6 の中で中間レベルであり，著しく簡単なシステムではないことがわかる．表 6 に記載する実装上の事情を考慮し案件 2，4 を除いて，残り案件 1，3，5 の「プログラム行数／SQL」の指数は案件 6 の 2 倍前後とわかる．単純換算で，案件 6 の仕組みは案件 1，3，5 よりプログラム行数の 50% 削減を実現できたと言える．

プログラム行数の削減は，開発工数の削減に繋がる．また，プログラム行数が少なければ，不具合調査が便利にでき，運用保守のコストダウンに繋がる．そして，上記仕組みの運用は，システム生命周期にわたりそれなりの経済効果が期待できる．

5. おわりに

Ajax のクライアント動作はデータ表示のみではなく、入力チェック、入力支援、デザイン変更、イベント増減などの要素もある。現在それらの対応として、個別画面に個別プログラムを組んでいる。もしそれらの要素もパターン化し DSL に取込めるとさらに効果がでる。

また、該当 DSL の複数開発言語への展開、既存開発プラットフォームへの移植、より表現力の高い語法的设计など、未開拓の内容はまだ多く存在している。

謝辞 本レポートの作成にご協力いただいた皆様に深謝いたします。

参考文献

- 1) jQuery 日本語リファレンス：<http://semooh.jp/jquery/>
- 2) JSON の日本語オンライン説明：<http://www.json.org/json-ja.html>
- 3) 循環的複雑度のオンライン説明：<https://ja.wikipedia.org/wiki/循環的複雑度>
- 4) understand ツールのオンライン説明：<https://www.techmatrix.co.jp/quality/understand/>

常 珂軍（正会員）kejun.chang@esc.co.jp

大連理工大学 工学修士

エスコ・ジャパン株式会社 SI 事業部開発部長

2001 年より、第一線のエンジニアとしてソフトウェア開発に携わっている。

投稿受付：2016 年 1 月 6 日

採録決定：2016 年 1 月 27 日

編集担当：斎藤正史（金沢工業大学）