JSテンプレート

実装方針

- 可読性、保守性のため、DOM構造を作るような処理はJSテンプレートで行う
- JSテンプレートライブラリは Handlebars.js を使用する
- 基本的に precompile は使用せず、テンプレートを画面JSPファイルに直接埋め込む形をとる

実装方法

テンプレートは次の様に画面JSPファイルに直接埋め込む。

```
<div>
  <!--  ...  -->
  <ul id="search-result-list">
  </ul>
  <!--  ...  -->
</div>
<script id="search-result-template" type="text/x-handlebars-template"><!--1-->
    {{#each  resultLi<!--2-->
    <li>{{foo}}</li><!--3-->
    {{/each}}
</script>
```

- <1> scriptタグにテンプレート文字列を記述する。type属性にはtext/x-handlebars-templateと指定する
- <2> #each でイテレーション可能
  - 他にも if, unlessなどのBuilt-In Helper がある
- <3> {{name}} という記法でテンプレートパラメータを埋め込む
  - {{name}}という記法はHTMLエスケープをした上で値を出力する (see http://handlebarsjs.com/#html-escaping)
  - {{foo.bar}}というように、ネストしたデータを参照することも可能 (see http://handlebarsjs.com/#paths)

なお、この例は次の様なテンプレートパラメータが渡される想定で記載している。

```
{
  "resultList":  [
        {
      "foo": "bar"
        },
        {
      "foo": "baz"
        }
    ]
}
```

そして、次の様に対象画面用のJSファイルにて、テンプレート文字列を取得、コンパイルし、レンダリングする。

```
$(function()  {
  'use  strict';

  var  contextPath  $('meta[name="contextPath"]').attr('content');
  var  searchResultTemplate  =  Handlebars.com$('#search-result-template').html()); //  <1>

  var $searchQueryInput  =$('#search-query-input');
  var $searchBtn  =$('#search-btn');
  var $searchResultList  =$('#search-result-list');

  $searchBtn.on('click', function()  {
```

```
    $.ajax(contextPath + '/DBAP0010/search', {
      type: 'GET',
      contentType: 'application/x-www-form-urlencoded; charset=UTF-',
      data: {
        query: $searchQueryInput.val()
              }
          }).function(data) {
      $searchResultList.html(searchResultTemplate(data)); //  <2>
          });

    return false;
     });

});
```

- <1> テンプレート文字列を取得し、Handlebars.compileメソッドでコンパイルしておく (compileメソッドの戻り値は関数)
- <2> テンプレート関数にテンプレートパラメータを渡してレンダリングする(HTML文字列を生成する)

Handlebars利用ガイド

Handlebars利用ガイド を参照のこと。

内際共通の Custom Helper 一覧

Expression Helper

Expression Helper	仕様
eq	a === b
eqw	a == b
neq	a !== b
neqw	a != b
lt	a < b
lte	a <= b
gt	a > b
gte	a >= b
and	a && b
or	a || b
not	!a
add	a + b
subtract	a - b
divide	a / b
multiply	a * b
mod	a % b
length	a.length, 配列のみサポート
range	a .. b, 例: {{range 0 3}} で [0, 1, 2] を生成する
br	改行を に変換する。引数がnullの場合は空文字列を返す

利用例

```
{{#if (eq foo 'hoge')}}
fooが文字列「hoge」なら表示される
{{/if}}
```

```
{{#if (and (eq foo 'hoge') (eq bar 777))}}
fooが文字列「hoge」かつbarが777なら表示される
{{/if}}

{{#each bazList}}
    {{add @index <!-- 1, 2, 3, ... と表示される -->
{{/each}}

{{#if (gt (length bazList) 1)}}
bazListの要素数が1より大きい場合に表示される
{{/if}}

{{#each (range 0 3)}}
    {{this<!-- 0, 1, 2 と表示される -->
{{/each}}

{{br footext}}<!-- "あ\nい\nう" が "あ<br>い<br>う" に変換される -->
```

Number Helper

- addCommas(num)
  - 数値を通貨形式に整形する
  - API仕様
  - 利用例：

```
<dd class="price"><span>{{addCommas basePriceFrom</span>円 / </dd>
<!-- basePriceFrom=10000 の場合、'10,000円 / 人' と表示される -->
```

アプリ固有の Custom Helper 追加方法

画面固有のHelperは次の様にテンプレート関数の引数を通して登録すること。

```
$(function() {

  var searchResultTemplate = Handlebars.com$('#search-result-template').html());

  var localHelpers = {
    foo: function(value) {
      return 'Foo: ' + value;
        },
    bar: function(value1, value2) {
      return 'Bar: ' + value1 + value2;
        }
    };

  var html = searchResultTemplate(data, {
    helpers: localHelpers// <1>
    });

});
```

- <1> テンプレート関数の第2引数にhelpersを渡す
  - 参考: execution options - Handlebars.js

Handlebars.registerHelper(...) を使うとグローバル領域にHelperを追加することになるため、望ましくない。
アプリで横断的に使用するHelperを登録したい場合は、適宜共通のJSファイルを作成し、ベースのJSP(或いはベースのtiles.xml)で読み込むようにする。
その際はHandlebars.registerHelper(...)を使って良い。

テンプレートの再利用方法

partials機能を使うことでテンプレートを再利用できる。

partials機能にはnormal partialsとinline partialsの2種類があり、
基本的に次に示すinline partialsの利用を推奨する。
なお、normal partialsを利用する場合は、custom helperの追加方法と同様、
可能な限りHandlebars.registerPartial(...)を使わず、テンプレート関数の引数を通して登録すること。

inline partialsを使ったテンプレートの例

```
<div>
  <div class="main-foo">{{#each    foobarOneList}} >    foobarTmpl}}    {{/eac</div> <!--1-->
  <div class="sub-foo">{{#each    foobarTwoList}} >    foobarTmpl}}    {{/eac</div>
</div>

{{#*    inline    'foobarTmp<!--2-->
<p class="foo-bar">{{foo}}    -    {{ba</p>
{{/inline}}
```

- <1> {{> partialsテンプレート名}} という記法をpartialsテンプレートを取り込む (これはnormal partialsの場合も同じ)
- <2> inline partialsは {{#* inline 'partialsテンプレート名'}} ..テンプレート内容.. {{/inline}} という記法で定義する

例えば、上記のテンプレートに対して、次のテンプレートパラメータを渡すと、

```
{
  foobarOneList:    [{
    foo: 'ふー1',
    bar: 'ばー1'
     }],
  foobarTwoList:    [{
    foo: 'ふー2',
    bar: 'ばー2'
     }]
}
```

レンダリング結果は次の様になる。

```
<div>
  <div class="main-foo"> <p class="foo-bar">ふー1    -    ば-</p>
 </div>
  <div class="sub-foo"> <p class="foo-bar">ふー2    -    ば-</p>
 </div>
</div>
```

参考

- Using Inline Partials and Decorators with Handlebars 4.0