ページング

全体の流れ

実装

1. ページングを行う際のControllerのメソッドの引数にPageableを追加する
    - リクエストパラメータ(初期表示時はデフォルト値)をもとにPageableクラスが生成されて引数に設定される
2. ControllerでAPIからの検索結果リストと総件数をもとにPageInfoクラスを生成する
    - PageInfoクラスの生成時にAPIの検索結果リスト、引数のPageable、API検索の総件数が必要
    - 生成したPageInfoクラスに検索結果リストとページングに必要な情報の両方を保持している(アプリ側では検索結果リストを画面表示に使用する)
3. Ajaxで画面へ返却するResponseにControllerで作成したPageInfoクラスを保持するためのgetter/setterを用意する
    - Responseクラス-PageInfoクラス-Pageクラス-検索結果リスト、というデータ構造になる
    - 検索結果リスト以外に画面表示使用する値(DBバージョン時間など)はResponseの配下に保持する
4. JSPの並び順の部分、ページング部分、件数表示部分にatd-paging-handlebars-partials.js(海外はati～)を使用する
    - 画面ごとのcssのclassについては各タグファイルのattributeで指定可能
5. JavaScriptに並び順とソートのイベントを実装する

データフロー

- リクエスト
    - 画面→Controller
        - リクエストパラメータ
            - ページ番号、並び順などを渡す(タグのdata属性に設定されている)
    - Controller
        - Pageable
            - RequestMappingメソッドの引数でPageableを受け取る
(引数に記載することでPageableHandlerMethodArgumentResolverがリクエストパラメータからPageableに設定する)
    - Controller→Service以降
        - 変更なし
            - Pageableからページ番号、表示件数、並び順を取得し、Service以降へ受け渡す

- レスポンス
    - Service→Controller
        - 変更なし
    - Controller
        - WebプロジェクトのResponse
            - GatewayのResponseから検索結果、総件数を取得し、PageInfoを生成
            - PageInfoをWebのResponseに設定
    - 画面
        - WebプロジェクトのResponse
            - Responseに保持しているPageInfo内のPageから検索結果リストを取得

PC

Controllerクラス

- Ajaxで画面側より呼び出されるメソッドでorg.springframework.data.domain.Pageableクラスを引数に追加する。
初回表示時にデフォルト値が設定されるようにデフォルトを設定する。

```
    /**  デフォルトの1ページ表示件数.  */
    private static final int  DEFAULT_SIZE  10;
    /**  デフォルトのソート(人気順).  */
    private static final String  DEFAULT_SORT  =  SearchConstant.SORTKEY_POPULAR;

    @GetMapping("defaultSearch")
    @ResponseBody
    public  XXXSXXXXP01PSearchResponse  defaultSearch(XXXSXXXXP01PXxxForm  form,
```

```
        @PageableDefault(size = DEFAULT_SIZE, sort = { DEFAULT_S Pageable pageable) // <1>
            ...
    }
```

<1>
初期表示時は@PageableDefaultに記載したデフォルト値が設定されたPageableが生成される。2回目以降は画面からのページ番号などが設定される。

- Service以降へ渡すページ番号、ソート条件はPageableから取得する。
  - Pageableのページ番号は 0から始まる ことに注意する。(1ページ目は0、2ページ目は1、...)

```
pageable.getPageNumber();
```

  - 表示件数はPageableからそのまま取得可能。

```
pageable.getPageSize();
```

  - ソート条件は jp.co.anas.atc.fw.web.paging.util.PagingUtils クラスを使用して文字列として取得する。

```
PagingUtils.getSortPropety(pageable);
```

- Serviceを呼び出し各APIから検索結果を取得する。
  (API検索処理は各画面で実装)
- APIのResponseから検索結果の一覧の件数、総件数を取得し、 jp.co.anas.atc.fw.web.paging.PageInfo クラスを生成する。

```
// API呼び出し
final HogeResponse hogeResponse = service.getHoge(request);

// APIのResponseの結果リストと総件数をもとにPageInfoを作成しResponseファイルに設定する
final XXXSXXXXP01PSearchResponse responsenew XXXSXXXXP01PSearchResponse();
final PageInfo<HogeHoge> pageInfonew PageInfo<>( // <1>
                    hogeResponse.getResponse().getHog// <2>
                    pageable, // <3>
                    hogeResponse.getResponse().getCou// <4>
response.setPageInfo(pageInfo);
response.setDbVersionTime(pkg01Response.getDbVersionTime());
```

  - <1> PageInfoクラスに指定する総称型はList内に保持するDTOの型
  - <2> 指定した検索条件、ページ番号、表示件数で返却された検索結果のリスト
  - <3> メソッドの引数のPageable
  - <4> 指定した検索条件に合致する総件数

- Controller全体イメージ表示隠す

```
@Controller
@RequestMapping("hoge/XXXPXXXX")
@DeviceMapping(DeviceType.PC)
public class XXXPXXXXPController {

    /** デフォルトの1ページ表示件数. */
    private static final int DEFAULT_SIZE 10;
    /** デフォルトのソート(人気順). */
    private static final String DEFAULT_SORT = SearchConstant.SORTKEY_POPULAR;

    @GetMapping("search")
```

```
    @ResponseBody
    public XXXSXXXXP01PSearchResponse defaultSearch(XXXSXXXXP01PXxxForm form,
            @PageableDefault(page = DEFAULT_PAGE, size = DEFAULT_SIZE, sort = { DEFAUPageable' })
    pageable) {

        // API検索に使用するRequestを生成する(ページ番号などを設定)
        final HogeRequest request = makeHogeRequest(form, pageable);
        // API呼び出し
        final HogeResponse hogeResponse = service.getHoge(request);

        // APIのResponseの結果リストと総件数をもとにPageInfoを作成しResponseファイルに設定する
        final XXXSXXXXP01PSearchResponse responsenew XXXSXXXXP01PSearchResponse();
        final PageInfo<HogeHoge> pageInfonew PageInfo<>(
                                    hogeResponse.getResponse().getHogeList(),
                                    pageable,
                                    hogeResponse.getResponse().getCount());
                    response.setPageInfo(pageInfo);
                    response.setDbVersionTime(pkg01Response.getDbVersionTime());

        return response;
          }

    private HogeRequest makeHogeRequest(XXXSXXXXP01PXxxForm foPageable pageable) {
        final HogeRequest requestnew HogeRequest();
                    request.setXxx(form.getXxx());
                    ...
                    request.setSort(PagingUtils.getSortPropety(pageable));
                    request.setRows(pageable.getPageSize());
                    request.setPage(pageable.getPageN1);ıber() +

        return request;
          }
}
```

Response(ajax response)

- GatewayプロジェクトのResponseクラスを直接画面へ渡している場合はWebアプリプロジェクトのResponseクラス(ajax response)を作成し、
  PageInfoクラスを保持するようにgetter/setterを用意する。

- 表示隠す

```
public class XXXSXXXXP01PSearchResponse {
    private PageInfo pageInfo;

    public PageInfo getPageInfo() {
        return pageInfo;
          }

    public void setPageInfo(PageInfo pageInfo) {
        this.pageInfo = pageInfo;
          }
    // PageInfo以外に画面表示に必要な値(DBバージョン時間など)も同様にgetter/setterを用意
}
```

JSP

件数表示(表示範囲)、並び順、ページングについては atd-paging-handlebars-partials.js(海外はati-paging-handlebars-partials.js)のpartials機能を使用する。

詳細はatd-paging-handlebars-partials.js リファレンス参照

一覧への検索結果表示

- 一覧の表示については下記のようにリストを繰り返しで使用する。
  (データ構造はResponse - PageInfo - Page - List、jsテンプレートにPageInfoを渡すJavaScript参照)

```
{{#each page.content}}
    //一覧に表示する各項目を表示する
{{/each}}
```

JavaScript

- 並び順、ページングのリンク押下時のイベント
  data属性(data-page-param)からページ番号と並び順を取得する

```
// ページング、ソートリンクのクリックハンドラ
$searchResultList.on('click', 'a[data-atd-page-param]', function(event) {
  var $pageElm =$(this);

        search({
    pageReqQueryString: $pageElm.data('atd-page-param')
        });

        event.preventDefa// ローディング表示のため、event.stopPropagation()は行わない
  }// <1>
```

  - <1> data属性(data-atd-page-param)はタグファイル内で固定で指定しているため変更しないこと
  - 海外の場合はdata属性「data-atd-page-param」の「atd」は「ati」

- 検索およびページング全体

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

  var contextPath $('meta[name="contextPath"]').attr('content');

  var $xxxForm =$('#xxx-form');
  var $searchResultList =$('#search-result-list');
  var searchResultTemplate = Handlebars.com$('#search-result-template').html());

  /** 検索処理 */
  var search function() {
    var cachedSearchCondition// 最新の検索条件

    return function(param) {
      var searchCondition = param.searchCondition || cachedSearchCondition;
      var qsPart = param.pageReqQueryStringnull ?'?' + param.pageReqQueryString'' :

      $.ajax(contextPath +'/xxx/XXXPXXXX/search' + qsPart, {
        type: 'GET',
        data: searchCondition
              function(data) {
                  cachedSearchCondition = sear// 最新の検索条件を保持しておく

        if (data.pageInfo.page.content.length 0) {
          $searchResultList.html(searchResultTemplate(data.pageInfo));
          else { }
          // 検索結果0件の場合
          $searchResultList.html(searchResultErrorTemplate());
                }
```

```
        });

    };
})();

/**　検索(デフォルト検索、再検索)の実行　*/
function searchWithInputtedCondition()　{
        search({
    searchCondition: $xxxForm.serializeArray()
        });
}

//　検索フォームのサブミットハンドラ
$xxxForm.on('submit', function()　{
        searchWithInputtedCondition();
  return false;
});

//　ページング、ソートリンクのクリックハンドラ
$searchResultList.on('click', 'a[data-atd-page-param]', function(event)　{
  var $pageElm　=$(this);

        search({
    pageReqQueryString: $pageElm.data('atd-page-param')
        });

        event.preventDefa//　ローディング表示のため、event.stopPropagation()は行わない
});

//　デフォルト検索実行
    searchWithInputtedCondition();

});
```

- 初期表示時および検索ボタン押下時のみFormから値を取得し、保持する。
  ページ番号、並び替え押下時はFormから値を取得しない。(保持しているFormを検索条件として使用する)
- 海外の場合はdata属性「data-atd-page-param」の「atd」は「ati」

atd-paging-handlebars-partials.js リファレンス

件数表示(表示範囲)

- 件数表示のタグを生成するために pagingNumber を使用してタグを生成する。
  - 国内
    - イメージ 300
    - 使用方法

      ```
      {{>　pagingNumber　numberDivCla'now-pages　pl1'　numberPClass'mt6'}}
      ```

      タグに設定するパラメータを指定する。(デザインHTML参照)
      - numberDivClass...件数表示部分(xx～xx／xx件中)のdivタグに設定するclass属性
      - numberPClass...件数表示部分(xx～xx／xx件中)のpタグに設定するclass属性
    - レンダリング結果 表示隠す

      ```
      <div class="now-pages　pl1"><p class="mt6">1～10／45件中</p></div>
      ```

  - 海外
    - イメージ 300
    - 使用方法

```
{{>  pagingNumber  numberPref'ご希望条件に合うツアーは'  numberPClass'asw-search-counter'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
- numberPrefix...総件数の前に表示する文言
- numberPClass...件数表示部分(xx～xx件)のpタグに設定するclass属性

■ レンダリング結果 表示隠す

```
<p class="asw-search-counter">ご希望条件に合うツアーは<strong>80件</strong>です　1～10件を表示</p>
```

- 海外の件数表示(下部)のタグを生成するために pagingNumberBottom を使用してタグを生成する。
  - イメージ _ 300
  - 使用方法

```
{{>  pagingNumberBottom  numberPCla'asw-search-counter'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
- numberPClass...件数表示部分(xx～xx件)のpタグに設定するclass属性

  - レンダリング結果 表示隠す

```
<p class="asw-search-counter"><strong>80件中</strong>　1～10件を表示</p>
```

並び順

- 並び順のタグを生成するために sortItem を使用してタグを生成する。
  - イメージ _ 300
  - 使用方法(ソートキーごとに作成)

```
{{>  sortItem  sortKe'price_low'  linkClass'asw-show-loading-indicator'  linkAttr'data-targetlayer="tour-table"'
  linkLabel'▼安い順'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
- sortKey　...APIへ渡すソート順項目の文字列 [必須]
- linkLabel　...Aタグを設定するソート順の画面表示文字列 [必須]
- liClass　...liタグのclass属性
- linkClass　...ソート順の文字列のAタグに設定するclass属性
- linkAttr　...ソート順の文字列のAタグに設定するclass属性以外の属性(data属性など)

  - レンダリング結果 表示隠す

```
<li><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table" data-atd-page-param="
page=0&sort=price_low">▼安い順</a></li>
```

ページング

- ページングのタグを生成するために pagingArea を使用してタグを生成する。
  - イメージ _ 300
  - 使用方法

```
{{>  pagingArea  pageDivCla'asw-pnr-list-pager  pt'  prevLiClass'asw-prev'  nextLiClass'asw-next'  linkClass'=
asw-show-loading-indicator'  linkAttr'data-targetlayer="tour-table"'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
- pageDivClass...ページング部分のdivタグに設定するclass属性
- prevLiClass...前へボタン(＜)のliタグに設定するclass属性
- nextLiClass...次へボタン(＞)のliタグに設定するclass属性
- linkClass...前へ、次へ、ページ番号のaタグに設定するclass属性
- linkAttr...前へ、次へ、ページ番号のaタグに設定するclass属性以外の属性(data属性など)

◦ レンダリング結果 表示隠す

```
<div class="asw-pnr-list-pager  pt">
  <ul>
    <li class="asw-prev"><a href="#" onclick="return false;"></a></li>
    <li><span>1</span></li>
    <li><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table" data-atd-page-param="
page=1&sort=popular">2</a></li>
    <li><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table" data-atd-page-param="
page=2&sort=popular">3</a></li>
    <li><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table" data-atd-page-param="
page=3&sort=popular">4</a></li>
    <li><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table" data-atd-page-param="
page=4&sort=popular">5</a></li>
    <li class="asw-next"><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table"
data-atd-page-param="page=1&sort=popular"></a></li>
  </ul>
</div>
```

SP

PCと差異がある箇所としてJSP、JSテンプレートの構成、JavaScript部分のみを記載している。

JSP

件数表示(SP)、並び順(SP)、ページング(SP)については
atd-paging-handlebars-partials-sp.js(海外はati-paging-handlebars-partials-sp.js) のpartials機能を使用する。
詳細はatd-paging-handlebars-partials-sp.js リファレンス参照

例

```
    ...
<div class="asw-operator">
  <div class="asw-sort">
    <div class="asw-sort-table" id="sort-table">      <%-- <1>  --%>
    </div>
  </div>
  <div class="asw-searchResult" id="result-count">    <%-- <2>  --%>
  </div>
</div>
<div id="xxx-table-inner">                            <%-- <3>  --%>
</div>
<div class="asw-operator  btn-open-wra">
  <div class="asw-operator-inner">
    <p id="next-link-area"></p>                       <%-- <4>  --%>
            ...
  </div>
</div>
    ...
```

- デザイン上、classのみでidが指定されていない箇所はidを付与する
  - <1> 並び順のテンプレートで更新される場所
  - <2> 総件数のテンプレートで更新される場所

- <3> 検索結果のテンプレートで更新または追加される場所
- <4> 次のn件を表示のテンプレートで置き換える場所(pタグそのものを置き換えるのでJSP上はpタグはそのまま残す)
  付与するidはpartials機能のpagingAreaで指定するidと合わせる。

JSテンプレート

JSテンプレート化すべき箇所は下記のようになる。

1. 【並び順】
2. 【総件数】
3. 【検索結果1】
4. 【検索結果2】
5. ...
6. 【検索結果10】
7. 【次のn件を表示】

- 【次のn件を表示】押下時はPCとは異なり、検索結果のみ追加表示する必要があるため、下記の4つにテンプレートを細分化する。
  - 並び順のテンプレート
  - 総件数のテンプレート
  - 検索結果(3-6)までのテンプレート
  - 次のn件を表示(7)のテンプレート

並び順のテンプレート

- 並び順は親タグとなるdivタグを残し、ulタグ配下をテンプレートとする。 表示隠す

```
<script id="search-sort-template" type="text/x-handlebars-template">
  <ul>
      >  sortItem  sortKey='recommend'  liClass='fz11'  linkLabel='おすすめ順'}}
      >  sortItem  sortKey='price_low'  liClass='fz11'  linkLabel='安い順'}}
      >  sortItem  sortKey='price_high'  liClass='fz11'  linkLabel='高い順'}}
      >  sortItem  sortKey='rating'  liClass='fz11'  linkLabel='評価が高い順'}}
  </ul>
</script>
```

総件数のテンプレート

- 総件数は親タグとなるdivタグを残し、テンプレートにはpartials機能を使用して総件数のpタグを生成する。 表示隠す

```
<script id="search-count-template" type="text/x-handlebars-template">
    {{>  pagingNumber  }}
</script>
```

検索結果のテンプレート

- 検索結果は画面によって親タグとなるdivタグが存在する場合としない場合がある。
  どちらの場合でも1件分のdivタグは存在するので1件分の表示範囲をテンプレートとする。 表示隠す

```
<script id="search-result-template" type="text/x-handlebars-template">
   #each  page.content}}
  <div class="asw-planData">
    <div class="asw-planData-inner">
      <p class="asw-planData-title">{{hoge}}</p>
      <div class="asw-planData-main">
        <p class="asw-planData-img"></p>
        <div>
          <p class="asw-planData-name">{{hogehoge}}</p>
```

```
        </div>
        /div>  <
              <div   class="asw-planData-point">
                  ...
                  <div   class="asw-planData-dtl">
                      <div>
                          <p   class="asw-planData-state/span></p>uga}}<
                          <p   class="asw-btn-base   asw-btn-main-stream   asw-btn-arrow-next   asw-btn-width-variable   asw-btn-ver
nput   type="submit"   value="ツアー詳細"   onClick="window.open('#'/p><
          /div>          <
        /div>      <
        <div class="asw-planData-notice">
          <ul>
            <li>{{fugafuga}}</li>
          </ul>
        </div>
        /div>  <
      /div>
    /div>
    /each}}
</script>
```

次のn件を表示のテンプレート

- 次のn件を表示は親タグとなるdivタグが存在するが並列で他のタグも存在するのでJSP上はpタグはそのまま残し、jsで置き換える
  テンプレートにはpartials機能を使用して次のn件を表示のpタグを生成する。 表示隠す

```
<script id="search-next-template" type="text/x-handlebars-template">
    {{>   pagingArea  'next-link-area'    linkClass'btn-open'}}
</script>
```

JavaScript

PCとの差異は初期表示時、並び順・検索ボタン押下時は検索結果部分のテンプレートを更新し、「次のn件を表示」押下時は検索結果部分の挿入を行う点。

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

  var   contextPath   $('meta[name="contextPath"]').attr('content');

  var $xxxForm   =$('#xxx-form');
  var $searchResultList   =$('#xxx-table-inner');
  var $searchSort   =$('#sort-table');
  var $searchCount   =$('#result-count');

  var   searchResultListTemplate   =   Handlebars.com$('#search-result-list-template').html());
  var   searchResultErroTemplate   =   Handlebars.com$('#search-result-error-template').html());
  var   searchSortTemplate   =   Handlebars.com$('#search-sort-template').html());
  var   searchCountTemplate   =   Handlebars.com$('#search-count-template').html());
  var   searchNextTemplate   =   Handlebars.com$('#search-next-template').html());

  /**   検索処理   */
  var   search   function()   {
    var   cachedSearchCondition//   最新の検索条件

    return function(param)   {
      var   searchCondition   =   param.searchCondition   ||   cachedSearchCondition;
      var   qsPart   =   param.pageReqQueryStringnull   ?'?'   +   param.pageReqQueryString'';   :
```

```
    $.ajax(contextPath + '/xxx/XXXPXXXX/search' + qsPart, {
      type: 'GET',
      data: searchCondition
          function(data) {
              cachedSearchCondition = sear// 最新の検索条件を保持しておく

      if (data.pageInfo.page.content.length 0) {

        if (!param.isAppendRenderMode) {                    // <1>
          // 検索(デフォルト検索、再検索)、並び順の場合、すべてのテンプレートを更新する
          $searchResultList.html(searchResultListTemplate(data.pageInfo));        // <2>
          else {                                            // <3>
          // 次のn件を表示の場合、検索結果を追加する
          $searchResultList.append(searchResultListTemplate(data.pageInfo));      // <4>
                  }

        $searchSort.html(searchSortTemplate(data.pageInfo));
        $searchCount.html(searchCountTemplate(data.pageInfo));
        $('#next-link-area').replaceWith(searchNextTemplate(data.pageInfo));      // <5>
        else { }
        // 検索結果0件の場合
        $searchResultList.html(searchResultErroTemplate());
        // TODO 0件の場合の並び順や件数表示などは仕様を確認
              }
          });

        };
    })();

/** 検索(デフォルト検索、再検索)の実行 */
function searchWithInputtedCondition() {
        search({
    searchCondition: $xxxForm.serializeArray()
        });
  }

// 検索フォームのサブミットハンドラ
$xxxForm.on('submit', function() {
        searchWithInputtedCondition();
  return false;
  });

// 次のn件を表示、ソートリンクのクリックハンドラ
$mainContents.on('click', 'a[data-atd-page-param]', function(event) {
  var $pageElm =$(this);

        search({
    pageReqQueryString: $pageElm.data('atd-page-param'),
    isAppendRenderMode: $pageElm.is('[data-atd-page-control="nextLink"]')        // <6>
        });

        event.preventDefa// ローディング表示のため、event.stopPropagation()は行わない
  });

//デフォルト検索実行
   searchWithInputtedCondition();

});
```

- <1> 検索結果部分の更新(初期表示、検索ボタンまたは並び順押下時)の場合
- <2> 細分化したテンプレートをすべて更新する
- <3> 検索結果部分の追加(「次のn件を表示」押下時)の場合
- <4> 検索結果を表示するエリアにappendするjQuery#append
  デザインHTMLの構成上、append()が難しい(検索結果と並列で他の要素が存在する)場合は検索結果の次の要素にidを付与してそのidで取得したセレクタの前にbefore()で挿入するjQuery#before

- <5> id指定したセレクタ自体を更新するため、html()ではなくreplaceWith()で置き換えるjQuery#replaceWith
  また、セレクタは毎回自身を置き換えるので、変数として保持するのではなく毎回取得する
- <6>
  data属性(data-atd-page-control)を取得し、検索結果を追加するかどうかを検索処理に渡す(data属性の付与は共通部品で実施)

atd-paging-handlebars-partials-sp.js リファレンス

件数表示(SP)

- 件数表示のタグを生成するために pagingNumber を使用してタグを生成する。
  - 国内イメージ _ 300
  - 海外イメージ _ 300
  - 使用方法

```
{{>  pagingNumber  numberPCla'taR'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
    - numberPClass...件数表示部分のpタグに設定するclass属性
  - レンダリング結果 表示隠す

```
<p class="taR">検索結果80件</p>
```

並び順(SP)

- 並び順のタグを生成するために sortItem を使用してタグを生成する。
  - イメージ _ 300
  - 使用方法(ソートキーごとに作成)

```
{{>  sortItem  sortKe'price_low'  liClass'fz11'  linkClass'asw-show-loading-indicator'  linkAttr'=
data-targetlayer="tour-table"'  linkLabel'安い順'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
    - sortKey ...APIへ渡すソート順項目の文字列 [必須]
    - linkLabel ...Aタグを設定するソート順の画面表示文字列 [必須]
    - liClass ...liタグのclass属性
    - linkClass ...ソート順の文字列のAタグに設定するclass属性
    - linkAttr ...ソート順の文字列のAタグに設定するclass属性以外の属性(data属性など)
  - レンダリング結果 表示隠す

```
<li class="fz11"><a href="#" class="asw-show-loading-indicator" data-targetlayer="tour-table" data-atd-page-param="
page=0&sort=price_low" data-atd-page-control="sortLink">安い順</a></li>
```

ページング(SP)

- ページングのタグ(「次のn件を表示」)を生成するために pagingArea を使用してタグを生成する。
  - イメージ _ 300
  - 使用方法

```
{{>  pagingArea  p'next-link-area'  linkClass'btn-open'}}
```

タグに設定するパラメータを指定する。(デザインHTML参照)
    - pId...pタグに設定するid [必須]
    - linkClass...aタグに設定するclass属性

※ pタグのclass属性については固定であることから指定しない形式としている

- レンダリング結果 表示隠す

```
<p class="asw-btn-base asw-btn-triton asw-btn-width-variable asw-btn-vertical-main asw-btn" id="next-link-area">
<a class="btn-open" data-atd-page-param="page=1&sort=price_low" data-atd-page-control="nextLink">次の10件を表示
</a></p>
```

- 補足
  - 「次のn件を表示」の「n」はController引数のPageableの1ページの表示件数を使用している(10固定ではない)
  - 次ページに表示する件数がPageableの1ページの表示件数に満たない場合は、「次のn件を表示」の「n」を残件数とする
    (例：トータル21件で1ページの表示件数が10件の場合)
    - ボタンは「次の10件を表示」となり、ボタン押下後は20件表示された状態で「次の1件を表示」となる
  - 次ページが存在しない場合はボタンを非表示とする

ファイル

sort.png	2.22 KB	2016/11/28	二階堂 弘貴
number.png	955 Bytes	2016/11/28	二階堂 弘貴
number_int.png	2.52 KB	2016/12/02	二階堂 弘貴
number_sp_dom.png	1.5 KB	2016/12/07	二階堂 弘貴
sort_sp.png	2.64 KB	2016/12/07	二階堂 弘貴
paging_sp.png	1.95 KB	2016/12/07	二階堂 弘貴
number_sp_int.png	1.53 KB	2016/12/08	二階堂 弘貴
paging.png	879 Bytes	2016/12/09	二階堂 弘貴
number_int_bottom.png	1.5 KB	2017/01/04	二階堂 弘貴