URL構築

URL構築方法

典型的なケースに対するURL構築方法を以下に示す。
これらのURL構築方法をとることで、URLエンコードやなベースURL付与などを透過的に処理することが出来る。

内部で行われている透過的な処理については、下記「参考: URL構築の考慮事項」を参照のこと。

[JSP] リンク・フォームURLの構築

spring:url タグとspring:paramタグを使用してURLを構築する。
URLエンコードはspring:url タグが行うのでアプリ側で意識する必要は無い。

```
<spring:url var="detailUrl" value="/sample/IIAP1200/init" htmlEscape="true">
  <spring:param name="receiptNo" value="${receiptNo}" />
  <spring:param name="sampleNo" value="SAMPLE" />
</spring:url>

<a href="${detailUrl}">詳細</a>
```

- spring:param タグのvalue属性値を個別にHTMLエスケープするのはNG
  - HTMLエスケープはURLエンコードの後に実施する必要があるため
  - 代わりに、spring:urlタグのhtmlEscape属性を使用する

パラメター が無い場合は、以下の様にspring:paramタグを使わない形になる。

```
<spring:url var="downloadUrl" value="/search/DSFP0040/download" />

<form:form id="log-download-form" method="get" action="${downloadUrl}" modelAttribute="DSFS0040P01PForm">
  <%--  ...  -%>
</form:form>
```

[Controller] リダイレクトURLの構築

次のページを参照のこと。

- Controllerの実装#リダイレクト先URLの指定方法
- Controllerの実装#リダイレクト先にデータを渡す方法

[Controller/Service] 外部URLの構築 (クエリ文字列の構築)

org.springframework.web.util.UriComponentsBuilder を使用する。
URLエンコードはUriComponentsBuilder が行う(UriComponentsBuilder#toUriString() で行う)のでアプリ側で意識する必要は無い。

なお、 java.net.URLEncoder を直接使用する必要があるケースは無い認識。

```
/**  アフィリエイト用CGIURL(DP).  */
@Value("${affiliate-site-cgi-url.dp}")
private String  dpCgiUrlForAffiliateSiteUrl;

public void  sampl(/*  ...  ) '{
  //  ...

  final  UriComponentsBuilder  uriBuilder  =  UriComponentsBuilder.fromHttpUrl(dpCgiUrlForAffiliateSites);
    uriBuilder.queryPara"REC_NO",  reserveCompleteDto.getReceiptNo());
    uriBuilder.queryPara"CARRY_OUT_DD",  deptDate);
    uriBuilder.queryPara"TOTAL_AMOUNT",  reserveCompleteDto.getTotalPrice().toString());
```

```
        uriBuilder.queryPara"AMC_NO",    repAmcNo);

        model.addAttribu"cgiAffiliateSiteUrl",    uriBuilder.toUriString());

    //    ...
}
```

[Controller/Service] アプリURLの構築 (クエリ文字列の構築)

アプリURLはJSP上で作るのが基本だが、次の様なケースの場合、Controller上で作りたくなる場合がある。

- ロジカルにリクエストパラメータを切り替える必要がある (JSPで構築するには保守性が低い)
- 戻り先GET URLをパラメータ付きでセッションに保持して後で使用したい

その時は、上記「 [Controller/Service] 外部URLの構築
(クエリ文字列の構築)」と同様に、UriComponentsBuilderでURLを構築すること。
但し、完全なHTTP URLを構築するわけではないので、 UriComponentsBuilder.fromHttpUrl(...)
ではなくUriComponentsBuilder.fromPath("/search/機能ID/init") で構築する。

[Gateway] 接続先URLの構築 (クエリ文字列の構築)

「[Controller/Service] 外部URLの構築 (クエリ文字列の構築)」と同様に、UriComponentsBuilderを使用する。

[JS] AjaxリクエストURLの構築

atc.ajaxのdataオプションを使用する。
atc.ajax(内部的には$.ajax)が内部で$.paramを使用し、自動でクエリ文字列の構築・URLエンコードを行う。

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

  //    ...

       atc.ajax(contextPath'/foo/DBAP0010/search',    {
      type: 'GET',
      data:    {
        q1: '...',
        q2: '...'
            }
      }).dofunction(data)    {

      });
```

詳細は次のページを参照のこと。

- Ajax

[JS] 画面表示URLの構築

クエリ文字列を$.paramで構築する。
URLエンコードは$.paramが行うのでアプリ側で意識する必要は無い。

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

  //    ...

       location.href    =    atc.url(contextPa'/foo/DBAP0010/init?'    +$.param({
      sample: 'SAMPLE',
      foo: 777
       }));
```

[JSテンプレート] リンクURLの構築

JSテンプレートで書き出すURLにJSテンプレートパラメータを含める必要がある場合は、
以下の様に、URL構築用のHandlebars Helperを作成し、その中でクエリ文字列を$.paramで構築する。
URLエンコードは$.paramが行うのでアプリ側で意識する必要は無い。

URL構築にmodelデータが必要な場合

modelデータ(JSPで書き出すデータ)はJSテンプレートの外で書き出しておく。
ここで示している例はパラメータ数が多いケースに対する実装例であり、
パラメータが少ない場合は $(..).val() などで一つ一つJS側で取得しても良い。(可読性の高い方を選択する)

jsp

```
<div id="hotel-detail-url-params" style="display:none">
  <input type="hidden" name="screenCls"       value="DETAIL_TAB">
  <input type="hidden" name="sortCls"         value="${f:h(shutterForm.sortCls)}">
  <input type="hidden" name="checkInDt"       value="${f:h(shutterForm.checkInDt)}">
  <input type="hidden" name="checkOutDt"      value="${f:h(shutterForm.checkOutDt)}">
  <input type="hidden" name="adultCnt"        value="${f:h(shutterForm.adultCnt)}">
  <input type="hidden" name="childAHtlCnt"    value="${f:h(shutterForm.childAHtlCnt)}">
  <input type="hidden" name="childBHtlCnt"    value="${f:h(shutterForm.childBHtlCnt)}">
  <input type="hidden" name="childCHtlCnt"    value="${f:h(shutterForm.childCHtlCnt)}">
  <input type="hidden" name="childDHtlCnt"    value="${f:h(shutterForm.childDHtlCnt)}">
  <input type="hidden" name="childEHtlCnt"    value="${f:h(shutterForm.childEHtlCnt)}">
  <input type="hidden" name="childFHtlCnt"    value="${f:h(shutterForm.childFHtlCnt)}">
  <input type="hidden" name="gdCls"           value="0"> <%--  プラン詳細商品区分  0:単品 >--%
</div>
```

js template

```
{{#each  pageInfo.page.content}}
  <%--  ...  -%>

  <a href="{{hotelDetailUrl  this">{{faclNam}}</a></dt>

  <%--  ...  -%>
{{/each}}
```

js

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

//  ...

//  ホテル検索結果テンプレート用Helper
var  searchHotelResultTemplateHelpers  =  {};

//  ホテル詳細URLの構築用Helper
searchHotelResultTemplateHelpers.hotelDetailUrl  =function(){
  var  extraQueryString  $('#hotel-detail-url-params  input[type="hidden').serialize();

  return function(hotelInfo)  {
    var  queryString  $.param({
      coptCd:  hotelInfo.coptCd,
      faclCd:  hotelInfo.faclCd
```

```
        '&' + extraQueryString;

    return atc.url(contextPath '/search/DSEP0080/init?' + queryString);
    };
}());

// ...

$searcHotelResultList.html(searchHotelResultTemplate(data, {
  helpers: searchHotelResultTemplateHelpers
}));
```

URL構築にmodelデータが不要な場合

modelデータが不要な場合は、以下の様に簡素化できる。

js template

```
{{#each pageInfo.page.content}}
  <%-- ... -%>

  <a href="{{hotelDetailUrl this">{{faclNam}}</a></dt>

  <%-- ... -%>
{{/each}}
```

js

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

// ...

// ホテル検索結果テンプレート用Helper
var searchHotelResultTemplateHelpers = {
  // ホテル詳細URLの構築用Helper
  hotelDetailUrl: function(hotelInfo) {
    var queryString $.param({
      screenCls: 'DETAIL_TAB',
      coptCd: hotelInfo.coptCd,
      faclCd: hotelInfo.faclCd,
      gdCls: '0' // プラン詳細商品区分 0:単品
        });

    return atc.url(contextPath '/search/DSEP0080/init?' + queryString);
    }
};

// ...

$searcHotelResultList.html(searchHotelResultTemplate(data, {
  helpers: searchHotelResultTemplateHelpers
}));
```

URL構築にそもそもJSテンプレートパラメータが不要な場合

URLにJSテンプレートパラメータを含める必要が無ければ、
JSPでのリンク作成と同様にspring:urlタグとspring:paramタグを使用してURLを構築すれば良い。

jsp

```
<spring:url var="sampleUrl" value="'/search/DSEP0080/init">
  <spring:param name="sample" value="SAMPLE" />
</spring:url>
```

js template

```
<a href="${sampleUrl}">サンプル</a></dt>
```

参考: URL構築の考慮事項

内部で行われている各々の透過的な処理についての説明、
及び、その透過的な処理を実施するために必要なアプリ側の実装を以下に示す。

用語定義

- 全アプリURL : Controller RequestMapping URL
- 静的リソースURL : warに含まれるcss/js/image URL

URLエンコード

```
/csm/search/DSDP0010/?paramA=ほげ&paramB[0].foo=YYY
  ↓
/csm/search/DSDP0010/?paramA=%E3%81%BB%E3%81%92&paramB%5B0%5D.foo=YYY
```

- 実装
  - java
    - spring:url x spring:param
    - RedirectAttributes
    - UriComponentBuilder
  - js
    - atc.ajaxのdataオプション (内部で結局、$.paramが使われる)
    - $.param

透過的なパラメータ付与

- トランザクショントークン
- CSRFトークンはURLに付与しないから考慮不要

```
/csm/search/DSDP0010/init
  ↓
/csm/search/DSDP0010/init?_tx=dsd-b5a3-57bebc5f
```

- 実装
  - java
    - spring:url x spring:param
    - redirect: (RedireceView)
  - js
    - atc.url
    - atc.ajax (内部でatc.urlを使用している)
- 修正が必要な実装パターン
  - <spring:url="/csm/search/DSDP0010/init" />?paramA={{paramA}}`
    - <spring:url="/csm/search/DSDP0010/init?paramA={{paramA}}" />` は修正しなくても大丈夫 (URLエンコードは別問題として)

- 素のformタグを使用している (明示的なトークン埋め込みが必要になる)

静的リソースURL変換

/csm/resources/js/app.js
↓
/csm/resources/js/app-58c07a622ea3c101af0d90c6e5f2547f.js

- 変換対象
  - 全てのJS、CSSファイルURL
  - imageファイルURLはベストエフォートで変換
- 実装
  - java
    - spring:url
  - js
    - 実装不可

参考: URL設計標準

URL設計標準.xlsx から転記↓

2.URL設計標準

(1) 内部URL
内部URLとは、WebLogicにデプロイされたアプリケーションが解釈可能なURLを指す。全体としては下記の構造となる（コンテキストパスを含む）。

サーバアドレス/ www.ana.co.jp
システム識別子/ システムを識別
固定文字列/ booking ※spookと
コンテキストルート/ アプリケー
機能ブロック名/ 機能ブロック単
機能ID(URL向け）/ 機能IDから端
※1
イベント名/ 起動するイベントを

例） www.ana.co.jp/domtour/booking/csm/search/DSBP0010/init

※1 機能ID(URL向け)

参考：機能ID
[サブシステムID]＋[機能ブロックの通番(1桁)]＋[ID種別(「P」固定)]+[連番(4桁)]＋[端末

例）
機能ID： DSBP0010P（PC向け）、DSBP0010P（S
機能ID（URL向け）： DSBP0010

(2) 公開URL
SEO対策のため、一部の画面については、内部URL異なる構造を持つ。コンテキストルート以下（パスパラメータを除く）個別の文字列として定義する。

内部URLのコンテキストルート以降と1:1の関係となる。

サーバアドレス/ www.ana.co.jp

システム識別子/ システムを識別

固定文字列/ booking ※spookと

個別URL/ 個別に定義したURL(ho

パスパラメータ/ ※1 個別パラ

例） www.ana.co.jp/domtour/booking/hotel-detail/hotel-id/9999/

※1 パスパラメータに含まれないリクエストパラメータについては、通常リクエストパラメータとして後続に出

。