↑Top

目次

改訂情報

変更年月日	変更内容
2013-07-01	初版

概要

イベントナビゲータでは、カスタマイズによって動作の振る舞いなどを変更できる拡張ポイントが用意されています。

主な拡張ポイントは以下の通りです。

- イベントフローの作成、更新、削除をハンドリングするリスナー
- 回答項目の非表示
- ナビゲート結果のリンクの非表示
- イベントフロー、質問、回答、ナビゲート結果およびリンクのタイトル、コメントの動的な変更
- リンク情報の動的な変更
- ナビゲート結果リンクへのステータス表示

イベントフローの作成、更新、削除をハンドリングする

イベントフローの作成、更新、削除通知を受けるリスナーを定義することができます。

1. リスナークラスを作成します。

   jp.co.intra_mart.foundation.navigator.EventFlowListenerを実装したクラスを作成します。

```
package sample;

import jp.co.intra_mart.foundation.navigator.EventFlow;
import jp.co.intra_mart.foundation.navigator.EventFlowListener;
import jp.co.intra_mart.foundation.navigator.EventNavigatorException;

public class SampleEventFlowListener implements EventFlowListener {

   @Override
   public void insert(EventFlow model) throws EventNavigatorException {

     // イベントフローが作成された時に実行する処理を記述します。

   }

   @Override
   public void update(EventFlow model) throws EventNavigatorException {

     // イベントフローが更新された時に実行する処理を記述します。

   }

   @Override
   public void delete(String eventFlowId) throws EventNavigatorException {

     // イベントフローが削除された時に実行する処理を記述します。

   }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。
3. 作成したディレクトリにplugin.xmlを作成します。
4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.navigator.listener">
    <listeners
      name="Sample Listener"
      id="sample.listener"
      version="8.0"
      rank="100">
      <event-flow-listener class="sample.SampleEventFlowListener"/>
    </listeners>
  </extension>
</plugin>
```

コラム

リスナーは複数設定することが可能です。

イベントフローを非表示にする

イベントフローの表示、非表示を制御することができます。

1. バリデータクラスを作成します。

   jp.co.intra_mart.foundation.navigator.EventFlowValidatorを実装したクラスを作成します。

```
package sample;

import jp.co.intra_mart.foundation.navigator.EventFlowInfo;
import jp.co.intra_mart.foundation.navigator.EventFlowValidator;
import jp.co.intra_mart.foundation.navigator.EventNavigatorException;

public class SampleEventFlowValidator implements EventFlowValidator {

   @Override
   public boolean validate(EventFlowInfo eventFlowInfo) throws EventNavigatorException
{

      // 表示する場合は、true 表示しない場合は falseを返却します。

      return true;
   }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。
3. 作成したディレクトリにplugin.xmlを作成します。
4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.navigator.validator">
    <validators
      name="Sample EventFlow Validator"
      id="sample.event.flow.validator"
      version="8.0"
      rank="100">
      <event-flow-validator class="sample.SampleEventFlowValidator"/>
    </validators>
  </extension>
</plugin>
```

コラム

バリデータは複数設定することが可能です。
複数設定した場合、設定されたバリデータのいずれかがfalseを返した場合に非表示扱いとなります。

回答を非表示にする

回答の表示、非表示を制御することができます。

1. バリデータクラスを作成します。

   jp.co.intra_mart.foundation.navigator.SelectItemValidatorを実装したクラスを作成します。

```
package sample;

import jp.co.intra_mart.foundation.navigator.EventNavigatorException;
import jp.co.intra_mart.foundation.navigator.SelectItemInfo;
import jp.co.intra_mart.foundation.navigator.SelectItemValidator;

public class SampleSelectItemValidator implements SelectItemValidator {

   @Override
   public boolean validate(SelectItemInfo selectItemInfo) throws
EventNavigatorException {

      // 表示する場合は、true 表示しない場合は falseを返却します。

      return true;
   }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。
3. 作成したディレクトリにplugin.xmlを作成します。
4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.navigator.validator">
    <validators
      name="Sample SelectItem Validator"
      id="sample.select.item.validator"
      version="8.0"
      rank="100">
      <select-item-validator class="sample.SampleSelectItemValidator"/>
    </validators>
  </extension>
</plugin>
```

コラム

バリデータは複数設定することが可能です。
複数設定した場合、設定されたバリデータのいずれかがfalseを返した場合に非表示扱いとなります。

リンクを非表示にする

ナビゲート結果のリンクの表示、非表示を制御することができます。

1. バリデータクラスを作成します。

   jp.co.intra_mart.foundation.navigator.LinkPageValidatorを実装したクラスを作成します。

```
package sample;

import jp.co.intra_mart.foundation.navigator.EventNavigatorException;
import jp.co.intra_mart.foundation.navigator.LinkPageInfo;
import jp.co.intra_mart.foundation.navigator.LinkPageValidator;

public class SampleLinkPageValidator implements LinkPageValidator {

   @Override
   public boolean validate(LinkPageInfo linkPageInfo) throws EventNavigatorException {

      // 表示する場合は、true 表示しない場合は falseを返却します。

      return true;
   }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。
3. 作成したディレクトリにplugin.xmlを作成します。
4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.navigator.validator">
    <validators
      name="Sample LinkPage Validator"
      id="sample.link.page.validator"
      version="8.0"
      rank="100">

      <link-page-validator class="sample.SampleLinkPageValidator"/>

    </validators>
  </extension>
</plugin>
```

コラム

バリデータは複数設定することが可能です。
複数設定した場合、設定されたバリデータのいずれかがfalseを返した場合に非表示扱いとなります。

タイトル、コメントを動的に変更する

タイトル、コメントを動的に変更することができます。

以下の例は、サンプルとして提供されている内容を記載しています。

- イベントフロー、質問、回答、ナビゲート結果およびリンクのタイトルおよびコメントに${userCd}という文字列が存在した場合、
  ログインしているユーザコードに変換します。

1. コンバータクラスを作成します。

   jp.co.intra_mart.foundation.navigator.TitleInfoConverterを実装したクラスを作成します。

   引数で渡された、TitleInfoの内容を変更して、返却します。

```
package jp.co.intra_mart.system.navigator;

import jp.co.intra_mart.foundation.context.Contexts;
import jp.co.intra_mart.foundation.context.model.AccountContext;
import jp.co.intra_mart.foundation.navigator.EventNavigatorException;
import jp.co.intra_mart.foundation.navigator.TitleInfo;
import jp.co.intra_mart.foundation.navigator.TitleInfoConverter;

/**
 * ユーザコードに置換するサンプルタイトルコンバータです。<br>
 * <br>
 * ${userCd}をログイン中のユーザコードに置き換えます。
 * @author INTRAMART
 * @version 8.0.0
 * @since 8.0.0
 */
public class SampleTitleInfoConverter implements TitleInfoConverter{

    private static final String TARGET = "\\$\\{userCd\\}";

    /**
     * タイトル情報を変換します。<br>
     * <br>
     * ${userCd}をログイン中のユーザコードに置き換えます。
     * @param titleInfo タイトル情報
     * @return 変換後のタイトル情報
     * @exception EventNavigatorException エラーが発生した場合にスローされます。
     */
    @Override
    public TitleInfo convert(final TitleInfo titleInfo) throws EventNavigatorException{

        final AccountContext ctx = Contexts.get(AccountContext.class);

        titleInfo.setTitle(titleInfo.getTitle().replaceAll(TARGET, ctx.getUserCd()));
        titleInfo.setComment(titleInfo.getComment().replaceAll(TARGET, ctx.getUserCd()));

        return titleInfo;
    }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。
3. 作成したディレクトリにplugin.xmlを作成します。
4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.navigator.converter">
    <converters
      name="Sample Title Converter"
      id="jp.co.intra_mart.foundation.navigator.converter.sample.title"
      version="8.0"
      rank="100">
      <event-flow-converter
class="jp.co.intra_mart.system.navigator.SampleTitleInfoConverter"/>
      <event-item-converter
class="jp.co.intra_mart.system.navigator.SampleTitleInfoConverter"/>
      <select-item-converter
class="jp.co.intra_mart.system.navigator.SampleTitleInfoConverter"/>
      <event-result-converter
class="jp.co.intra_mart.system.navigator.SampleTitleInfoConverter"/>
      <result-page-converter
class="jp.co.intra_mart.system.navigator.SampleTitleInfoConverter"/>
    </converters>
  </extension>
</plugin>
```

コラム

それぞれのタグの意味は以下の通りです。

タグ	説明
event-flow-converter	イベントフローのタイトルとコメント。
event-item-converter	質問のタイトルとコメント。
select-item-converter	回答のタイトルとコメント。
event-result-converter	ナビゲート結果のタイトルとコメント。
result-page-converter	ナビゲート結果のリンクのタイトルとコメント。

コラム

コンバータは複数設定することが可能です。
PluginManagerによる設定順序ルールに基づき、すべてのコンバータを処理します。

リンク情報を動的に変更する

リンクの情報（URLやパラメータ）を動的に変更することができます。

1. バリデータクラスを作成します。

   jp.co.intra_mart.foundation.navigator.LinkPageConverterを実装したクラスを作成します。

   引数で渡された、LinkPageInfoの内容を変更して、返却します。
   ただし、変更できるのは、URLとパラメータのみです。
   タイトルとコメントは変更しても反映されません。

```
package sample;

import jp.co.intra_mart.foundation.context.Contexts;
import jp.co.intra_mart.foundation.context.model.AccountContext;
import jp.co.intra_mart.foundation.navigator.EventNavigatorException;
import jp.co.intra_mart.foundation.navigator.LinkPageConverter;
import jp.co.intra_mart.foundation.navigator.LinkPageInfo;

public class SampleLinkPageConverter implements LinkPageConverter {

    private static final String TARGET = "\\$\\{userCd\\}";

    @Override
    public LinkPageInfo convert(LinkPageInfo model) throws EventNavigatorException {

        // パラメータの値に ${userCd}が存在した場合に、ログインしているユーザのユーザコードに変更します。

        final AccountContext ctx = Contexts.get(AccountContext.class);

        String[] keys = model.getKeys();

        for(String key : keys) {
            String value = model.getParamValue(key);
            if(value != null) {
                model.setParamValue(key, value.replaceAll(TARGET, ctx.getUserCd()));
            }
        }
        return model;
    }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。

3. 作成したディレクトリにplugin.xmlを作成します。

4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<extension point="jp.co.intra_mart.foundation.navigator.converter">
  <converters
    name="Sample LinkPage Converter"
    id="test_002"
    version="8.0"
    rank="200">

    <link-page-converter class="sample.SampleLinkPageConverter"/>

  </converters>
</extension>
```

コラム

コンバータは複数設定することが可能です。
PluginManagerによる設定順序ルールに基づき、すべてのコンバータを処理します。

ナビゲート結果のリンクにステータスを表示する

ナビゲート結果のリンクにステータスを表示することができます。

このプロバイダが1つ以上設定されている場合、「ナビゲート結果」画面のリンク一覧のカラムに「状態」が表示されます。

以下の例は、イベントナビゲータ IM-Workflow アドオンにサンプルとして提供されている内容を記載しています。

- IM-Workflow 連携されたリンクに対して、ログインしているユーザが申請中である場合に、申請済のステータスを返却しています。

1. プロバイダクラスを作成します。

   jp.co.intra_mart.foundation.navigator.LinkPageStatusProviderを実装したクラスを作成します。

   ステータスの判定を行わない場合は、nullを返却します。
   ステータスに設定する値は、多言語を意識してコーディングする必要があります。

```
package jp.co.intra_mart.foundation.navigator.workflow;

import jp.co.intra_mart.foundation.navigator.EventNavigatorException;
import jp.co.intra_mart.foundation.navigator.LinkPageInfo;
import jp.co.intra_mart.foundation.navigator.LinkPageStatus;
import jp.co.intra_mart.foundation.navigator.LinkPageStatusProvider;
import
jp.co.intra_mart.foundation.workflow.application.general.ProcessedActvMatterList;
import
jp.co.intra_mart.foundation.workflow.application.general.condition.ListSearchCondition;
import jp.co.intra_mart.foundation.workflow.application.general.condition.OperatorType;
import
jp.co.intra_mart.foundation.workflow.application.model.column.ActvMatterPullBackType;
import
jp.co.intra_mart.foundation.workflow.application.model.condition.ProcessedAuthCondition;

import jp.co.intra_mart.foundation.workflow.exception.WorkflowException;
import jp.co.intra_mart.system.navigator.workflow.message.NaviCaption;

/**
 * ワークフロー申請状態チェック（サンプル）
 * @author INTRAMART
 * @version 8.0.0
 * @since 8.0.0
 */
public class SampleWorkflowLinkPageStatusProvider implements LinkPageStatusProvider
{
```

```
    /**
     * ワークフロー申請が申請済みかどうか判定します。
     * @param userCd ユーザコード
     * @param linkPageInfo リンクページ情報
     * @return 申請状態
     * @throws EventNavigatorException エラーが発生した場合にスローされます。
     */
    public LinkPageStatus getStatus(final String userCd, final LinkPageInfo linkPageInfo)
throws EventNavigatorException {

        if (WorkflowLinkPageConst.WORKFLOW_URL.equals(linkPageInfo.getUrl())) {

            // 申請されているか判定
            try {

                // フローIDを取得
                final String flowId =
linkPageInfo.getParamValue(WorkflowLinkPageConst.PARAM_NAME_FLOW_ID);

                // 検索条件
                final ProcessedAuthCondition condition = new ProcessedAuthCondition();
                condition.setApplyFlg("1");

                final ListSearchCondition<ActvMatterPullBackType> condition2 = new
ListSearchCondition<ActvMatterPullBackType>();
                condition2.addCondition(ActvMatterPullBackType.FLOW_ID, flowId,
OperatorType.EQ);

                // 判定
                final ProcessedActvMatterList processedActvMatterList = new
ProcessedActvMatterList(userCd);

                // 1件以上あれば、
                if (processedActvMatterList.getProcessedListCount(condition, condition2) > 0)
{
                    return new
LinkPageStatus(NaviCaption.CAP_Z_NAVIGATOR_WORKFLOW_LINK_PAGE_STATUS_APPLIED
"green");
                } else {
                    return new
LinkPageStatus(NaviCaption.CAP_Z_NAVIGATOR_WORKFLOW_LINK_PAGE_STATUS_UNAPPLY
"red");
                }
            } catch (final WorkflowException e) {
                throw new EventNavigatorException(e);
            }
        }
        return null;
    }

}
```

2. WEB-INF/plugin フォルダに任意のディレクトリを作成します。
3. 作成したディレクトリにplugin.xmlを作成します。
4. 作成したplugin.xmlを以下のように定義します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.navigator.provider">
    <providers
      name="Sample Provider"
      id="jp.co.intra_mart.foundation.navigator.provider.sample"
      version="8.0"
      rank="100">
      <link-page-status-provider
class="jp.co.intra_mart.foundation.navigator.workflow.SampleWorkflowLinkPageStatusProv

    </providers>
  </extension>
</plugin>
```

コラム

プロバイダは複数設定することが可能です。
PluginManagerによる設定順序ルールに基づき、上位のプロバイダから処理を行い
最初にステータスが取得できたプロバイダの値を利用します。