PopupLinkSubmit

NOTE: This is outdated information that applies only to Tapestry 4.

PopupLinkSubmit

This is a solution for submitting a form and at the same time open a popup with an external link. It is a little lousy since it requires alot of copy paste from Tapestry code. I hope this can look better with future Tapestry versions.

Example

```
<a href="#" jwcid="@PopupLinkSubmit" parameters="ognl:article" listener="listener:acheterArticle" popuppage="
PopupPanier" popupparameters="ognl:article.AR_Design">Acheter</a>
```

PopupLinkSubmit.jwc

```
<!DOCTYPE component-specification PUBLIC "-//Apache Software Foundation//Tapestry Specification 4.0//EN"
"http://jakarta.apache.org/tapestry/dtd/Tapestry_4_0.dtd">
<component-specification class="actualis.web.tapestry.components.PopupLinkSubmit" allow-body="yes" allow-
informal-parameters="yes" deprecated="no">

  <description>
  Creates a hyperlink that submits its enclosing form using JavaScript.
  </description>

  <parameter name="disabled"/>
  <parameter name="selected"/>
  <parameter name="tag"/>
  <parameter name="id" property="idParameter" default-value="id"/>

  <parameter name="listener">
      <description>
          A listener that is notified if this component is triggered.
      </description>
  </parameter>
  <parameter name="action">
      <description>
          A listener that is notified if this component is triggered
          just before the form's listener, after all components
          enclosed by the Form have had a chance to update their properties.
      </description>
  </parameter>
  <parameter name="parameters">
    <description>
    An object, or list of objects, gathered when the link is triggered and
    made available as listener parameters in the request cycle, making
    the parameters available to a deferred listener.
    </description>
  </parameter>

  <parameter name="popuppage" property="popupPage" required="yes"/>

  <parameter name="popupparameters" property="popupParameters" required="no"/>

  <reserved-parameter name="name"/>
  <reserved-parameter name="href"/>

  <inject property="listenerInvoker" object="infrastructure:listenerInvoker"/>
  <inject property="script" type="script" object="PopupLinkSubmit.script"/>

</component-specification>
```

PopupLinkSubmit.script

```
<?xml version="1.0"?>
<!DOCTYPE script PUBLIC
        "-//Apache Software Foundation//Tapestry Script Specification 3.0//EN"
        "http://jakarta.apache.org/tapestry/dtd/Script_3_0.dtd">
<script>

<input-symbol key="name" class="java.lang.String" required="yes"/>
<input-symbol key="popupLink" class="java.lang.String" required="yes"/>
<input-symbol key="form" class="org.apache.tapestry.IForm" required="yes"/>

<let key="href">
  javascript:submitPopupLink('${form.name}', '${name}','${popupLink}');
</let>

</script>
```

```
package actualis.web.tapestry.components;

import java.util.HashMap;
import java.util.Map;

import org.apache.hivemind.ApplicationRuntimeException;
import org.apache.hivemind.service.BodyBuilder;
import org.apache.tapestry.IComponent;
import org.apache.tapestry.IForm;
import org.apache.tapestry.IMarkupWriter;
import org.apache.tapestry.IRequestCycle;
import org.apache.tapestry.IScript;
import org.apache.tapestry.PageRenderSupport;
import org.apache.tapestry.TapestryUtils;
import org.apache.tapestry.annotations.InjectObject;
import org.apache.tapestry.engine.ExternalServiceParameter;
import org.apache.tapestry.engine.IEngineService;
import org.apache.tapestry.engine.ILink;
import org.apache.tapestry.form.FormConstants;
import org.apache.tapestry.link.DirectLink;

import actualis.web.tapestry.components.form.FormMessages;

/**
 * Implements a component that submits its enclosing form via a JavaScript link. [<a
 * href="../../../../../ComponentReference/LinkSubmit.html">Component Reference</a>]
 *
 * @author Richard Lewis-Shell
 * @version $Id$
 */

public abstract class PopupLinkSubmit extends CustomSubmit {
  /**
   * The name of an {@link org.apache.tapestry.IRequestCycle}attribute in which
   * the current submit link is stored. LinkSubmits do not nest.
   */

  public static final String ATTRIBUTE_NAME = "actualis.web.tapestry.PopupLinkSubmit";

  public static final String ATTRIBUTE_NAME_SCRIPT = "actualis.web.tapestry.PopupLinkSubmitScript";

  /**
   * The name of an {@link org.apache.tapestry.IRequestCycle}attribute in which
   * the link submit component that generates the javascript function is stored.
   * The function is only required once per page (containing a form with a
   * non-disabled LinkSubmit)
   */
  public static final String ATTRIBUTE_FUNCTION_NAME = "actualis.web.tapestry.PopupLinkSubmit_function";

  @InjectObject("engine-service:external")
  public abstract IEngineService getExternalService();
```

```
/**
 * Checks the submit name ({@link FormConstants#SUBMIT_NAME_PARAMETER}) to
 * see if it matches this LinkSubmit's assigned element name.
 */
protected boolean isClicked(IRequestCycle cycle, String name) {
  String value = cycle.getParameter(FormConstants.SUBMIT_NAME_PARAMETER);

  return name.equals(value);
}

public ILink getLink(IRequestCycle cycle) {
  Object[] pageParameters = DirectLink.constructServiceParameters(getPopupParameters());
  ExternalServiceParameter esp = new ExternalServiceParameter(getPopupPage(),
      pageParameters);

  return getExternalService().getLink(false, esp);
}

public abstract Object getPopupParameters();

public abstract String getPopupPage();

/**
 * @see org.apache.tapestry.form.AbstractFormComponent#renderFormComponent(org.apache.tapestry.IMarkupWriter,
 *      org.apache.tapestry.IRequestCycle)
 */
@Override
protected void renderFormComponent(IMarkupWriter writer, IRequestCycle cycle) {
  boolean disabled = isDisabled();

  if (!disabled) {
    PageRenderSupport pageRenderSupport = TapestryUtils.getPageRenderSupport(cycle,
        this);

    if (cycle.getAttribute(ATTRIBUTE_NAME_SCRIPT) == null) {
      BodyBuilder builder = new BodyBuilder();

      builder.addln("function {0}(form, elementId, url)", "submitPopupLink");
      builder.begin();
      builder.addln("var form = document.getElementById(form);");
      builder.addln("form.events.submit(elementId);");
      builder.addln("aWindow = window.open(url, '" + PopupDirectLink.POPUP_WINDOWNAME
          + "', " + PopupDirectLink.POPUP_FEATURES + ", false);");
      builder.addln("aWindow.focus();");
      builder.end();

      pageRenderSupport.addBodyScript(builder.toString());
      cycle.setAttribute(ATTRIBUTE_NAME_SCRIPT, this);
    }

    IForm form = getForm();

    String slink = getLink(cycle).getURL(null, true);
    Map symbols = new HashMap();
    symbols.put("form", form);
    symbols.put("name", getName());
    symbols.put("popupLink", slink);

    getScript().execute(cycle, pageRenderSupport, symbols);

    writer.begin("a");
    writer.attribute("href", (String) symbols.get("href"));

  }

  renderBody(writer, cycle);

  if (!disabled)
    writer.end();
}
```

```
  public abstract IScript getScript();

  /**
   * @see org.apache.tapestry.AbstractComponent#prepareForRender(org.apache.tapestry.IRequestCycle)
   */
  protected void prepareForRender(IRequestCycle cycle) {
    IComponent outer = (IComponent) cycle.getAttribute(ATTRIBUTE_NAME);

    if (outer != null)
      throw new ApplicationRuntimeException(FormMessages
          .linkSubmitMayNotNest(this, outer), this, getLocation(), null);

    cycle.setAttribute(ATTRIBUTE_NAME, this);
  }

  /**
   * @see org.apache.tapestry.AbstractComponent#cleanupAfterRender(org.apache.tapestry.IRequestCycle)
   */
  protected void cleanupAfterRender(IRequestCycle cycle) {
    cycle.removeAttribute(ATTRIBUTE_NAME);
  }

  /**
   * Links can not take focus, ever.
   */
  protected boolean getCanTakeFocus() {
    return false;
  }

  /**
   * Returns true; the LinkSubmit's body should render during a rewind, even if
   * the component is itself disabled.
   */
  protected boolean getRenderBodyOnRewind() {
    return true;
  }

}
```

AbstractSubmit has package visibility! So you need to copy/paste the whole thing.

```
package actualis.web.tapestry.components;

import java.util.Collection;

import org.apache.tapestry.IActionListener;
import org.apache.tapestry.IForm;
import org.apache.tapestry.IMarkupWriter;
import org.apache.tapestry.IRequestCycle;
import org.apache.tapestry.form.AbstractFormComponent;
import org.apache.tapestry.listener.ListenerInvoker;

/**
 * Copy of AbstractSubmit
 */
public abstract class CustomSubmit extends AbstractFormComponent {

  /**
   * Determine if this submit component was clicked.
   *
   * @param cycle
   * @param name
   * @return true if this submit was clicked
   */
  protected abstract boolean isClicked(IRequestCycle cycle, String name);

  /**
   * @see org.apache.tapestry.form.AbstractFormComponent#rewindFormComponent(org.apache.tapestry.IMarkupWriter,
org.apache.tapestry.IRequestCycle)
```

```
 */
protected void rewindFormComponent(IMarkupWriter writer, IRequestCycle cycle)
{
    if (isClicked(cycle, getName()))
        handleClick(cycle, getForm());
}

void handleClick(final IRequestCycle cycle, IForm form)
{
    if (isParameterBound("selected"))
        setSelected(getTag());

    final IActionListener listener = getListener();
    final IActionListener action = getAction();

    if (listener == null && action == null)
        return;

    final ListenerInvoker listenerInvoker = getListenerInvoker();

    Object parameters = getParameters();
    if (parameters != null)
    {
        if (parameters instanceof Collection)
        {
            cycle.setListenerParameters(((Collection) parameters).toArray());
        }
        else
        {
            cycle.setListenerParameters(new Object[]
            { parameters });
        }
    }

    // Invoke 'listener' now, but defer 'action' for later
    if (listener != null)
      listenerInvoker.invokeListener(listener, CustomSubmit.this, cycle);

    if (action != null) {
          Runnable notify = new Runnable()
          {
              public void run()
              {
                  listenerInvoker.invokeListener(action, CustomSubmit.this, cycle);
              }
          };

        form.addDeferredRunnable(notify);
    }
}

/** parameter */
public abstract IActionListener getListener();

/** parameter */
public abstract IActionListener getAction();

/** parameter */
public abstract Object getTag();

/** parameter */
public abstract void setSelected(Object tag);

/** parameter */
public abstract boolean getDefer();

/** parameter */
public abstract Object getParameters();

/** Injected */
```

```
  public abstract ListenerInvoker getListenerInvoker();
}
```