Tapestry5SubmitContextComponent

This is especially useful in loops, where the component name is dynamic.

To use it, add this to your page template:

```
<input t:type="SubmitContext" t:id="theSubmit" t:context="idString"/>
```

add this to your page class:

```
void onSelectedFromTheSubmit(String idString)
{
    // do something based on idString
}
```

Source:

(Modified from T5.0.4. Put it into yourapp.components package. Updated to work with 5.0.12 by 9902468.)

Check here for a version based on the Tapestry5.1 Submit component

```
// Copyright 2007 The Apache Software Foundation
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
// http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

package yourapp.components;

import org.apache.tapestry.ComponentResources;
import org.apache.tapestry.MarkupWriter;
import org.apache.tapestry.annotations.Environmental;
import org.apache.tapestry.annotations.Inject;
import org.apache.tapestry.annotations.Parameter;
import org.apache.tapestry.corelib.base.AbstractField;
import org.apache.tapestry.services.FormSupport;
import org.apache.tapestry.services.Heartbeat;
import org.apache.tapestry.services.Request;

/**
  * Corresponds to &lt;input type="submit"&gt;, a client-side element that can force the
  * enclosing form to submit. The submit responsible for the form submission will post a
  * notification that allows the application to know that it was the responsible entity. The
  * notification is named "selected" and has a String context.
  */
public final class SubmitContext extends AbstractField
{
     static final String SELECTED_EVENT = "selected";

     /**
      * If true, then any notification sent by the component will be deferred until the end of
      * the form submission (this is usually desirable).
      */
     @Parameter
     private boolean _defer = true;

     @Parameter
     private String _context;

     @Environmental
```

```
private FormSupport _formSupport;

@Environmental
private Heartbeat _heartbeat;

@Inject
private ComponentResources _resources;

@Inject
private Request _request;

public SubmitContext()
{
}

SubmitContext(Request request)
{
    _request = request;
}

void beginRender(MarkupWriter writer)
{
    // write a hidden input for the context
    //String elementName = getElementName();
    String elementName = getControlName(); //Modified to work with 5.0.12
    writer.element("input", "type", "hidden", "name", elementName + "X", "value", _context);
    writer.end();

    // now the submit
    writer.element("input", "type", "submit", "name", elementName, "id", getClientId());
    _resources.renderInformalParameters(writer);
}

void afterRender(MarkupWriter writer)
{
    writer.end();
}

//protected void processSubmission(FormSupport formSupport, String elementName)
@Override
protected void processSubmission(String elementName) //Modified to work with 5.0.12
{
    String value = _request.getParameter(elementName);
    final String context = _request.getParameter(elementName + "X");

    if (value == null)
        return;

    Runnable sendNotification = new Runnable()
    {
        public void run()
        {
            _resources.triggerEvent(SELECTED_EVENT, new Object[] {context}, null);
        }
    };

    // When not deferred, don't wait, fire the event now (actually, at the end of the current
    // heartbeat). This is most likely because the Submit is inside a Loop and some contextual
    // information will change if we defer. Another option might be to wait until the next
    // heartbeak?

    if (_defer)
        _formSupport.defer(sendNotification);
    else
        _heartbeat.defer(sendNotification);

}

// For testing:

void setDefer(boolean defer)
```

```
    {
        _defer = defer;
    }

    void setup(ComponentResources resources, FormSupport support, Heartbeat heartbeat)
    {
        _resources = resources;
        _formSupport = support;
        _heartbeat = heartbeat;
    }
}
```