ajax div template

⚠ The Ajax theme is experimental. Feedback is appreciated.

The ajax div template provides a much more interesting div rendering option that the other themes do. Rather than simply rendering a `<div>` tag, this template relies on advanced AJAX features provided by the Dojo Toolkit. While the div tag could be used outside of the ajax theme, it is usually not very useful. See the div tag for more information on what features are provided.

Features

The remote div has a few features, some of which can be combined with the a tag and the ajax a template. These uses are:

- Retrieve remote data
- Initialize the div with content before the remote data is retrieved
- Display appropriate error and loading messages
- Refresh data on a timed cycle
- Listen for events and refresh data
- JavaScript control support

Retrieve Remote Data

The simplest way to use the div tag is to provide an *href* attribute. For example:

```
<saf:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"/>
```

What this does after the HTML page is completely loaded, the specified URL will be retrieved asynchronously in the browser. The entire contents returned by that URL will be injected in to the div.

Initializing the Div

Because the remote data isn't loaded immediately, it is sometimes useful to have some placeholder content that exists before the remote data is retrieved. The content is essentially just the body of the div element. For example:

```
<saf:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239">
    Placeholder...
</saf:div>
```

If you wish to load more complex initial data, you can use the action tag and the *executeResult* attribute:

```
<saf:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239">
    <ww:action id="weather" name="weatherBean" executeResult="true">
        <ww:param name="zip" value="97239"/>
    </ww:action>
</saf:div>
```

Loading and Error Messages

If you'd like to display special messages when the data is being retrieved or when the data cannot be retrieved, you can use the *errorText* and *loadingText* attributes:

```
<saf:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"
        loadingText="Loading weather information..."
        errorText="Unable to contact weather server">
    Placeholder...
</saf:div>
```

Refresh Timers

Another feature this div template provides is the ability to refresh data on a timed basis. Using the *updateFreq* and the *delay*attributes, you can specify how often the timer goes off and when the timer starts (times in milliseconds). For example, the following will update every minute after a two second delay:

```
<saf:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"
        loadingText="Loading weather information..."
        errorText="Unable to contact weather server">
        delay="2000"
        updateFreq="60000"
    Placeholder...
</saf:div>
```

Listening for Events

The a tag (specifically the ajax a template) and the div tag support an ajax event system, providing the ability to broadcast events to topics. You can specify the **topics** to listen to using a comma separated list in the *listenTopics* attribute. What this means is that when a topic is published, usually through the ajax a template, the URL specified in the *href* attribute will be re-requested.

```
<saf:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"
        loadingText="Loading weather information..."
        errorText="Unable to contact weather server"
        listenTopics="weather_topic,some_topic">
    Placeholder...
</saf:div>
<saf:a id="link1"
      theme="ajax"
      href="refreshWeather.action"
      notifyTopics="weather_topic,other_topic"
      errorText="An Error ocurred">Refresh</saf:a>
```

JavaScript Support

There are also javascript functions to refresh the content and stop/start the refreshing of the component. For the remote div with the component id "remotediv1":

To start refreshing use the javascript:

```
remotediv1.startTimer();
```

To stop refreshing use the javascript:

```
remotediv1.stopTimer();
```

To refresh the content use the javascript:

```
remotediv1.refresh();
```

JavaScript Examples:

To further illustrate these concepts here is an example. Say you want to change the url of a div at runtime via javascript. Here is what you need to do: What you will need to do is add a JS function that listens to a JS event that publishes the id from the select box that was selected. It will modify the URL for the div (adding the id so the correct data is obtained) and then bind() the AJAX div so it refreshes.

```
<saf:head theme="ajax" />

<script type="text/javascript">
    function updateReports(id) {
       var reportDiv= window['reportDivId'];
       reportDiv.href = '/../reportListRemote.action?selectedId='+id;
       reportDiv.refresh();
    }
    dojo.event.topic.getTopic("updateReportsListTopic").subscribe(null, "updateReports");
</script>

<form ... >
<saf:select .... onchange="javascript: dojo.event.topic.publish("updateReportsListTopic", this.value); " />

<saf:div id="reportDivId" theme="ajax" href="/.../reportListRemote.action" >
  Loading reports...
</saf:div>
</form>
```