Project WONDER-Frameworks-Ajax-AjaxUpdateContainer

AjaxUpdateContainer

AjaxUpdateContainer is a particularly cool component that allows any component on your page to be refreshed in the background without a full page refresh. AjaxUpdateContainer uses a WOComponentContent. All content between its webobjects tags will be refreshed when it is ordered.

If your AjaxUpdateContainer contains component actions, you **must** include the ERExtensions framework in your application and extend ERXApplication and ERXSession as described in the Project Wonder Quickstart guide. If all of your actions are Direct Actions inside the update container, this is not necessary.

Additionally, if you intend to use the "observeFieldID" binding, you must also include the ERExtensions framework (as described above). The requirement is because a partial form submit is necessary (to send the modified observed field back to the server), and WOForm required a change to be able to recognize that it should takeValuesFromRequest for the observed value.

A convenience Javascript function is autogenerated for each AjaxUpdateContainer of the form "<id>Update()". For instance, if your id is "messageBox", then the Javascript function would be named "messageBoxUpdate()". Calling this function will cause the container to refresh.

A series of demos of AjaxUpdateContainer is available in the AjaxExamples project in Project Wonder.

Wonder Bindings

- id (required) - the ID of the generated HTML element that will contain your content
- elementName - the HTML element name of the container ("div" or "span", defaults to "span")
- observeFieldID - This provides support similar to the observe_field helper on Ruby on Rails. The value of observeFieldID should be the id of the form element you want to observe (for instance, set the id = "Something" on a WOTextField and set observeFieldID = "Something"; on your update container). When the field changes, this will cause the field you were watching to submit its new value and the update container will refresh. This allows you to do things like dependent WOPopUpButtons where one popup button changes, causing the second popup button to refresh with new values. I've only tested this where both are in the same form.

Prototype Bindings

Prototype Documentation

- frequency - when specified, the update container will automatically periodically refresh at this rate (in seconds)
- decay - if the container updates and the contents do not change, slow down the rate of update by this factor each request
- evalScripts - whether or not script tags should be evaluated when the container updates
- insertion
- onComplete
- onSuccess
- onFailure
- onException