ERPrototaculous

Introduction

ERPrototaculous was developed to provide the ajax functionality in ERDivaLook. It's a *pseudo-stateless* Ajax framework for WebObjects.

What is ERPrototaculous?

Features include:

- 'Organic' support for Prototype and Scriptaculous in WebObjects.
  - Light-weight dynamic elements to support Prototype
  - Transparent API that doesn't hide or abstract Prototype and Scriptaculous
  - Sans patches or extensions to Prototype
- Use of Unobtrusive Javascript.
- Pseudo-stateless ajax responses
- A set of widgets in the Prototype + Scriptaculous family.

Unobtrusive Javascript in ERPrototaculous

Unable to render {include} The included page could not be found.

```
er.prototaculous.useUnobtrusively = true
```

Prototype WebObjects Elements

`Ajax.Updater` and `Ajax.Request` have been implemented as WebObjects elements.

Ajax.Updater

Support for Prototype's Ajax.Updater is in the form of three elements:

- AjaxUpdaterLink
- AjaxUpdaterButton
- AjaxUpdaterForm (with `onsubmit` for ajax form submission)

These components will update *any* **container** on the page:

```
new Ajax.Updater('container', url, {options});
```

Ajax.Request

Prototype's Ajax.Request is in the form of:

- AjaxRequestLink
- AjaxRequestButton

These are used for strictly **background** ajax communication:

```
new Ajax.Request(url, {options});
```

Prototype + Scriptaculous Widgets

- Accordion
- LightWindow
- ModalBox
- CalendarDateSelect
- FileUpload

Support for these widgets have been similarly implemented as **button** and **link** variants, depending whether it is used inside a form or not.

Ajax Forms in ERPrototaculous

Differences from using forms in WebObjects. i.e `WOForm`:

1. All ajax form controls must be named. This includes text fields, selects and buttons.
   (WebObjects dynamically assigned names are not compatible with ERPrototaculous).
2. All ajax forms in an ERPrototaculous app need to be instances of `AjaxUpdaterForm`.
3. Ajax form submit buttons can be a:
   - Static `<button>`.
   - WOSubmitButton (if the result is to update whole page/app).
   - AjaxUpdaterButton (to update a **container**). Or
   - AjaxRequestButton (for a **background** ajax request).

So forms are different in Ajax.framework and ERPrototaculous.

In ERPrototaculous you may still have typical WebObjects forms (i.e WOForm) as well as use ajax forms (i.e **AjaxUpdaterForm**).
They behave differently in that an AjaxUpdaterForm will update the contents of a container as opposed to the entire page.

Warning

You must **name your form controls**, otherwise under certain circumstances Prototype (ajax) form submission will break.

Embrace Statelessness!

Unable to render {include} The included page could not be found.

ERPrototaculous embraces the stateless model Ajax offers in exchange for simplifying the work WebObjects has to do - it's a win-win!

So you may observe one notable difference between the ERPrototaculous and Ajax frameworks is in the way they handle ajax responses.
In ERPrototaculous, updates and actions break with typical WebObjects behavior by being pseudo-stateless.

Ajax Response Handling in ERPrototaculous

In a typical WebObjects application, when a user navigates to the previous page using the browser back button and subsequently clicks on a link in on that page, WebObjects needs to remember how to handle that action and to return the correct page. This is no longer necessary for ajax.

A user never travels backwards or forwards through the ajax application history.
(i.e there is no forward/back buttons for ajax requests - just as there aren't forward/back buttons on desktop apps).

So for ajax, the current state of the page fragment component is all that is necessary.

No More "You backtracked too far"

A 100% ajax WO-app (like an ERDivaLook app) is no longer plagued by the well known *limitation* of WebObjects - the browser **backtra ck problem**.

Ajax Actions in ERPrototaculous

Typically, in a WebObjects application, an action would return the contents of the entire page.

Ajax responses are mostly page fragments or just part of a page.
So you should make sure the actions in ERPrototaculous (or AjaxUpdaterButton and AjaxRequestButton) return the proper page fragment as opposed to the entire page.
This is different to how WebObjects normally works, so this is where you need to be careful.

Note

ERPrototaculous actions **differ** from Ajax.framework actions - *They must return only the page fragment*.

To assist with page fragments inside **forms**, you can make your ajax action responses a subclass of **WXPageFragment** (a utility to handle the forms processing between Prototype and WebObjects).

Update Containers in ERPrototaculous

Perhaps the only real similarity to Ajax.framework is the ajax update container.
WXGenericContainer is the ERPrototaculous update container.
It is similar to an AjaxUpdateContainer when it has the binding `ajax = true`, otherwise it's pretty much like **WOGenericContainer**.

It has been implemented as a utility or *convenience* for Prototype's `Ajax.Updater`.
That is it can be used to update from a DOM *event* like popup *onchange*, or from a Prototype *callback* like *onComplete*, etc.

Compatibility

ERPrototaculous can not be used with WebObjects 5.3 as it is dependent on the hooks for ajax added to WebObjects with version 5.4.

ERPrototaculous is WebObjects *5.4* compatible only

External Links

WOWODC '10 (Slides) - DirectToWeb 2.0