Wisej Extensions

AspNetControl

1 Overview

The *AspNetControl* class is a generic wrapper based on *Wisej.Web.IFramePanel*. The component cannot be used by itself, you must wrap a specific AspNet control class and create a new usable class.

```
public class ReportViewer :
AspNetWrapper<Microsoft.Reporting.WebForms.ReportViewer> {

}
```

You may also register the *Wisej.Web.Ext.AspNetControl.HttpModule* with *Web.config* in order to enable the wrapper. *Wisej.Web.Ext.AspNetControl.HttpModule* is a simple virtual path provider that is able to serve the embedded page wrapper Wisej.AspNetHost.aspx without having to save it to a local file.

Add it to Web.config like this:

```
<system.webServer>
<modules>
<add name = "WisejAspNetControl" type="Wisej.Web.Ext.AspNetControl.HttpModule,
Wisej.Web.Ext.AspNetControl" />

…
```

Starting from build 1.4.50 it is no longer necessary to add the HttpModule to Web.config. The assembly self-registers using the `PreApplicationStartMethod` attribute.

2 Usage

To complete the wrapper implementation, override *OnInit* or *OnLoad* – they correspond to the PageInit and PageLoad events in the Asp.Net page life cycle – and initialize the wrapped control *this.WrappedControl*: attach events, set properties, etc.

The property *this.WrappedControl* is typed to the type of the wrapped AspNet control. You can also use *this.IsPostback*, or *this.IsCallback* to determine the type of initialization that you need to perform.

3 Wrapping Up

It’s up to your wrapper class to provide events, methods and properties to the Wisej application using it.

You may fire any event while handling the events from the wrapper control (if you attached the handlers in OnInit or OnLoad), store properties and update the wrapped control when at the next OnInit or OnLoad.

Note that nothing can be done to the wrapped control outside of the processing of OnInit or OnLoad. That’s the nature of AspNet.