Click to Open

What It Is

Click to Open (C2O) allows you to open components in Eclipse directly from the running application in your browser! Click to Open appears in the lower left corner of browser as part of the pages of your running application. Clicking on this component, and then on an object in the browser window, opens the relevant WOComponent in Eclipse. This makes life easier for UI designers and for developers getting familiar with new projects. It also provides some other very slick debugging tools.

Check out the screencast at the mDimension build site

Click to Open is a browser based extension to WOLips found in Project Wonder. All of the Wonder ERD2W components support Click to Open.

Note that click-to-open support is expensive, because it has to dig around your component HTML quite a bit, so you will take a performance hit in development to have it enabled.

*Also note that if you use ERExcelLook or ERPDFGeneration that you will want to disable Click to Open in development. The former you can do in a rule:

```
*10 : pageConfiguration like '*Excel' => clickToOpenEnabled = "false" [com.webobjects.directtoweb.
BooleanAssignment]
```

The latter in your component:

```
public boolean clickToOpenEnabled(WOResponse response, WOContext context) {
     return false;
 }
```

What You Need

You need the **WOLips** framework that is part of Project Wonder. You also need the **ERExtensions** framework that is part of Project Wonder on the class path at runtime. If you already use Project Wonder, you are almost done.

If you are not using Project Wonder, you can download it from mDimension's site, untar the frameworks, and copy just the WOLips.framework and ERExtensions.framework to where you have the rest of your frameworks (usually /Library/Frameworks or ~/Library/Frameworks).

Getting Set Up

Add the WOLips.framework

Follow the tutorial on adding a framework to add WOLips.framework and ERXExtentions.framework to your application.

Add Support to Component Base Class

If your components extend Wonder's ERXComponent, you can skip this step. Once again, using Wonder makes your life easier.

If you don't have a custom component base class, you really should. Using com.webobjects.appserver.WOComponent as the super-class for your pages and components is just going to leave you doing the same things over and over. If you don't have one, you might want to start using the ClickToOpenComponent below.

You will need to add the appendToResponse(WOResponse, WOContext) method below to your component base class, or add this code to your appendToResponse method if you already have one.

Here is an example implementation of a component base class and of Click to Open support:

```
package net.com.foo.bar;

import com.webobjects.appserver.*;


/**
 * Support for "Click to Open" navigation from the browser to the template in Eclipse.  To enable this,
 * launch with:
 * <pre>
 * -Der.component.clickToOpen=true
 * </pre>
 */
public class ClickToOpenComponent extends com.webobjects.appserver.WOComponent {

    public static final boolean isClickToOpenEnabled = Boolean.parseBoolean(System.getProperty("er.component.
clickToOpen", "false"));

    public ClickToOpenComponent(WOContext context) {
        super(context);
    }

    public void appendToResponse(WOResponse response, WOContext context) {
        ERXClickToOpenSupport.preProcessResponse(response, context, isClickToOpenEnabled);
        super.appendToResponse(response, context);
        ERXClickToOpenSupport.postProcessResponse(getClass(), response, context, isClickToOpenEnabled);
    }
}
```

For components that can't sub-class ClickToOpenComponent (directly or indirectly), you can enable Click to Open by adding this method to your component:

```
public void appendToResponse(WOResponse response, WOContext context)
{
    ERXClickToOpenSupport.preProcessResponse(response, context, ClickToOpenComponent.isClickToOpenEnabled);
    super.appendToResponse(response, context);
    ERXClickToOpenSupport.postProcessResponse(getClass(), response, context, ClickToOpenComponent.
isClickToOpenEnabled);
}
```

Note that when isClickToOpenEnabled is false, the ERXClickToOpenSupport methods are no-ops.

Add Support to Application

If your Application.java class extends Wonder's ERXApplication, you can skip this step too. Otherwise, add a developmentMode() method like this:

```
private Boolean isDevelopmentMode;
public boolean developmentMode() {
    if (isDevelopmentMode == null) {
            isDevelopmentMode = new Boolean(System.getProperty("er.extensions.ERXApplication.developmentMode",
"false"));
    }
    return isDevelopmentMode.booleanValue();
}
```

Set Application Properties

In your application's Properties file, add this line:

```
wolips.password=yourpassword
```

If you need to, you can also change these default values:

```
wolips.host=localhost
wolips.port=9485
```

Provide prototype.js

WOLips.framework needs a prototype.js. If you are using Ajax framework, you don't need to do anything, because it will default to use Ajax.framework's prototype.js. However, if you are not, add this to your application's Properties file (as an example, change the values as appropriate):

```
wolips.prototype.framework=app
wolips.prototype.fileName=prototype.js
```

Add WOLToolBar to Your Pages

In your page wrapper's HTML template, add

```
<wo:WOLToolBar/>
```

and you're done. If you don't have a page wrapper, you will have to add this to every page. Hint: page wrappers make your life easier.

If you are using the old WO template syntax, add this to the .html file:

```
<webobject name="WOLToolBar" />
```

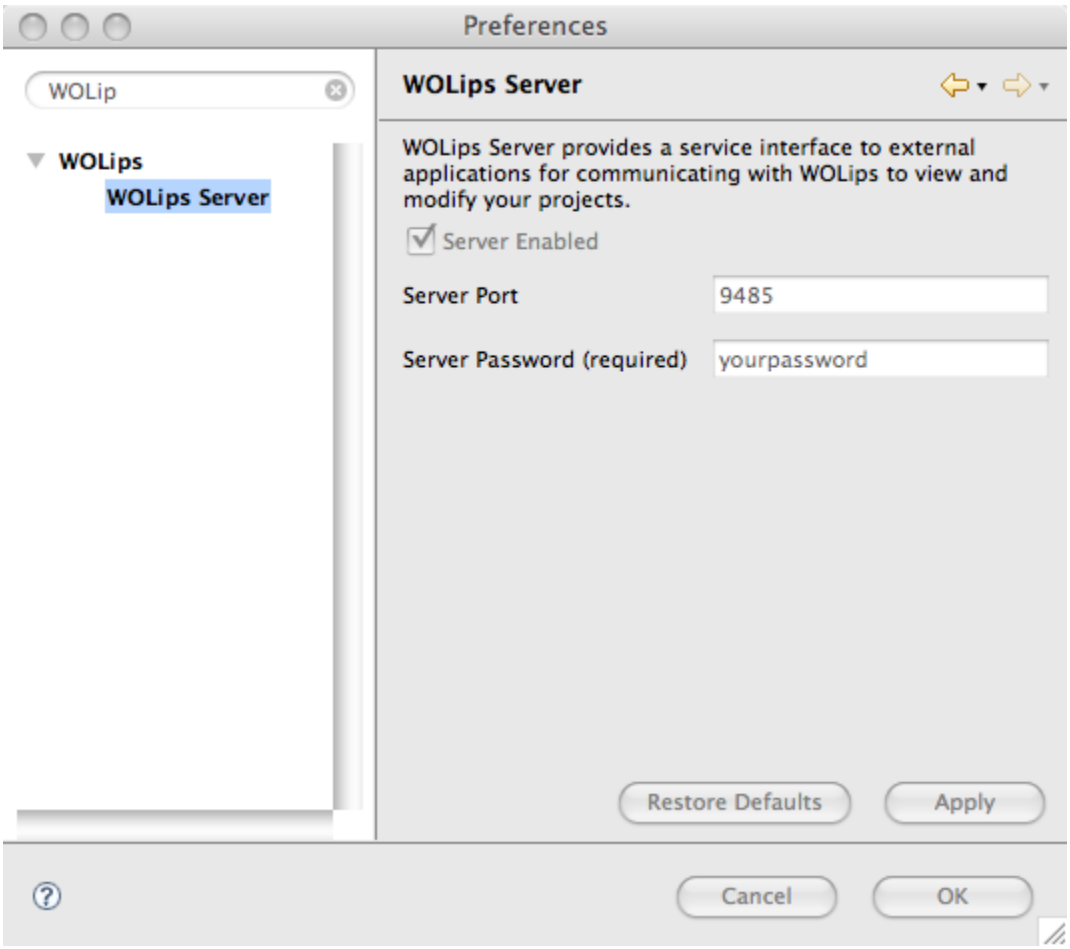
And add this to the .wod file:

```
WOLToolBar: WOLToolBar{
}
```

Configure WOLips Server

You must be using a recent version of WOLips that supports the **WOLips Server**. In your WOLips preferences, you must enable the WOLips Server and set the communication password. This password must match the `wolips.password` in the **Set Application Properties** section above.

Turning on the WOLips Server requires Eclipse to be restarted.



You can optionally change the port number. If you do change the port number, see `wolips.port` in the **Set Application Properties** section above.

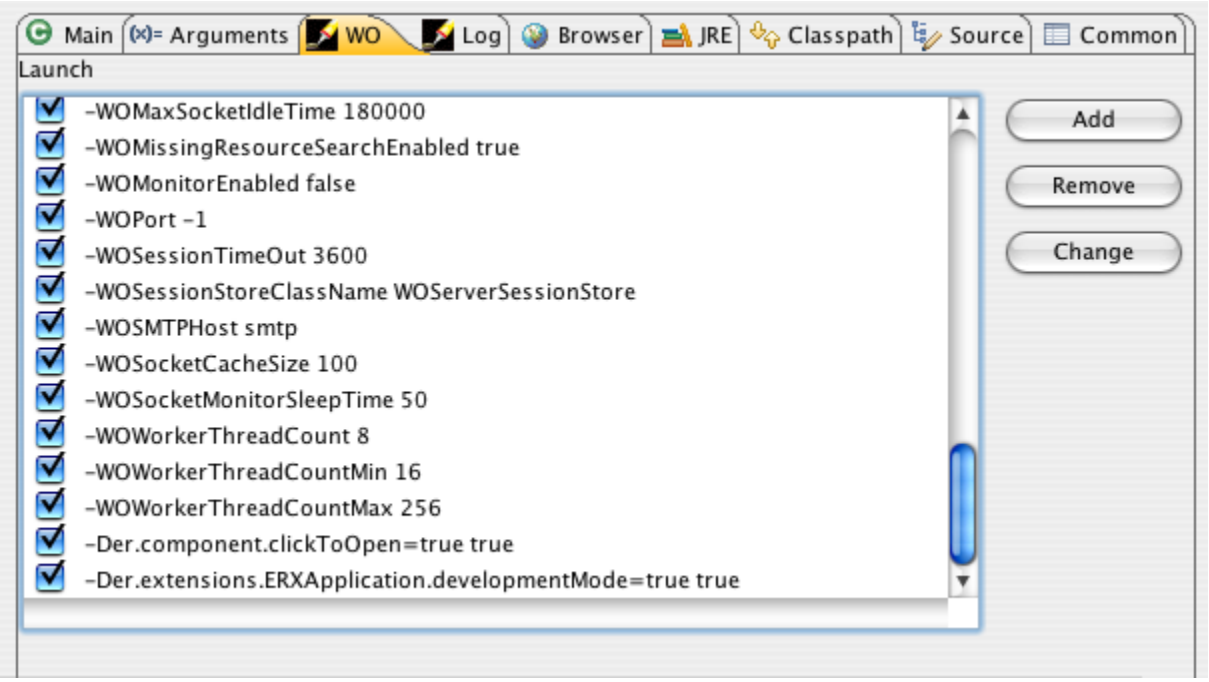
Enable Click to Open

If you are using Wonder, add the following to your Properties file:

- **er.component.clickToOpen=true**
- **er.extensions.ERXApplication.developmentMode=true**

If you are not using Wonder, add the Launch parameters as follows:

- Parameter = **-Der.component.clickToOpen=true**, Argument = **true**
- Parameter = **-Der.extensions.ERXApplication.developmentMode=true** Argument = **true**

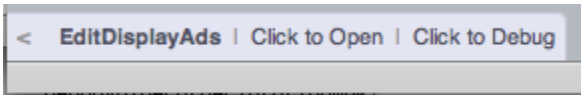


Using Click to Open

Run your application and look in the lower, left hand corner. You should see a link like this:

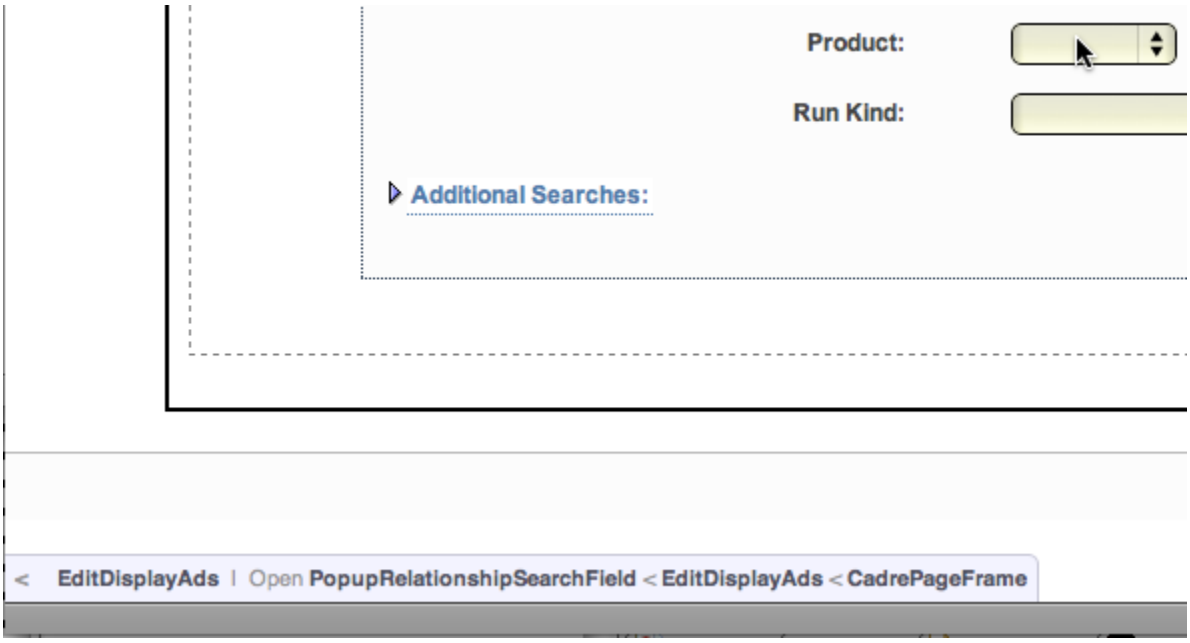
If you don't, check that the page has the WOLToolBar on it and that the `er.component.clickToOpen` property is set to true and the `er.extensions.ERXApplication.developmentMode` property is set to true.

Click on this component to open the Click to Open UI:



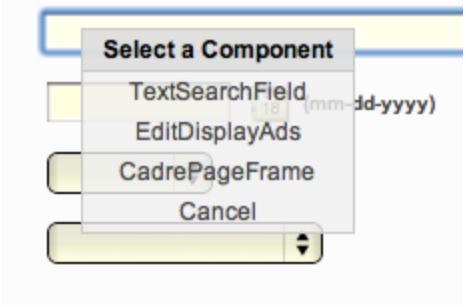
EditDisplayAd is the page in the browser. Click on this link to open this page in Eclipse.

If you are looking for a sub-component of this page, click on the **Click to Open** link. As you move your mouse over the page, the bread crumb of components will change to show you where you are. Just click to open the component under the mouse in Eclipse. It is that easy!



Additional Functionality

- Esc is a shortcut for getting out of click-to-open mode
- hold down the Cmd key while you move the mouse around and it will highlight the component
- Cmd-click it will popup the stack of components and you can pick from the stack:



- you can set the binding `expanded=true;` on WOLToolBar so it's open by default, instead of closed