Click to Debug

What It Is

Click to Debug is a Click to Open extension that allows you to toggle binding debugging from the UI while your application is running.

What You Need

See the What You Need section of the Click to Open documentation if you are not already using Click to to Open.

Getting Set Up

See the Getting Set Up section of the Click to Open documentation if you are not already using Click to to Open.

Add this line to the `Properties` file in your application:

```
ognl.debugSupport=true
```

Add Support to Application

If your Application.java class extends (directly or indirectly) Wonder's ERXApplication, you can skip this step. Otherwise, add this to your Application() constructor, or a method that runs before requests are processed::

```
NSNotificationCenter.defaultCenter().addObserver(this,
                                                 ERXSelectorUtilities.notificationSelector
("applicationDidHandleRequest"),
                                                 WOApplication.ApplicationDidDispatchRequestNotification, null);
```

Then add this method to handle this notification:

```
/**
 * When request is finished, we remove the context from thread local storage.
 *
 * @see #createContextForRequest(WORequest)
 * @param n notification
 */
public void applicationDidHandleRequest(NSNotification n) {
    ERXWOContext.setCurrentContext(null);
    ERXThreadStorage.removeValueForKey(ERXWOContext.CONTEXT_DICTIONARY_KEY);
}
```

The next part is a method that sets ERXWOContext.currentContext() when a request is dispatched:

```
/**
 * When a context is created we push it into thread local storage.
 *
 * @see #applicationDidHandleRequest(NSNotification)
 * @param request the request
 * @return the newly created context
 */
public WOContext createContextForRequest(WORequest request) {
        WOContext context = super.createContextForRequest(request);
        // We only want to push in the context the first time it is
        // created, ie we don't want to lose the current context
        // when we create a context for an error page.
        if (ERXWOContext.currentContext() == null) {
                ERXWOContext.setCurrentContext(context);
        }
        return context;
}
```

And finally, add this code to support Click to Debug:

```
protected void _debugValueForDeclarationNamed(WOComponent component, String verb, String aDeclarationName,
                                              String aDeclarationType, String aBindingName,
                                              String anAssociationDescription, Object aValue) {
        if (aValue instanceof String) {
                StringBuffer stringbuffer = new StringBuffer(((String) aValue).length() + 2);
                stringbuffer.append('"');
                stringbuffer.append(aValue);
                stringbuffer.append('"');
                aValue = stringbuffer;
        }
        if (aDeclarationName.startsWith("_")) {
                aDeclarationName = "[inline]";
        }

        StringBuffer sb = new StringBuffer();

        //NSArray<WOComponent> componentPath = ERXWOContext._componentPath(ERXWOContext.currentContext());
        //componentPath.lastObject()
        //WOComponent lastComponent = ERXWOContext.currentContext().component();
        String lastComponentName = component.name().replaceFirst(".*\\.", "");
        sb.append(lastComponentName);

        sb.append(verb);

        if (!aDeclarationName.startsWith("_")) {
                sb.append(aDeclarationName);
                sb.append(":");
        }
        sb.append(aDeclarationType);

        sb.append(" { ");
        sb.append(aBindingName);
        sb.append("=");

        String valueStr = aValue != null ? aValue.toString() : "null";
        if (anAssociationDescription.startsWith("class ")) {
                sb.append(valueStr);
                sb.append("; }");
        }
        else {
                sb.append(anAssociationDescription);
                sb.append("; } value ");
                sb.append(valueStr);
        }

        NSLog.debug.appendln(sb.toString());
```

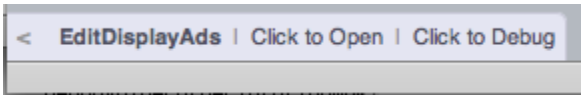
```
}

/**
 * The set of component names that have binding debug enabled
 */
private NSMutableSet<String> _debugComponents = new NSMutableSet<String>();

/**
 * Little bit better binding debug output than the original.
 */
@Override
public void logTakeValueForDeclarationNamed(String aDeclarationName, String aDeclarationType,
                                            String aBindingName, String anAssociationDescription, Object
aValue) {
        WOComponent component = ERXWOContext.currentContext().component();
        if (component.parent() != null) {
                component = component.parent();
        }
        _debugValueForDeclarationNamed(component, " ==> ", aDeclarationName,
                                       aDeclarationType, aBindingName, anAssociationDescription, aValue);
}

/**
 * Little bit better binding debug output than the original.
 */
@Override
public void logSetValueForDeclarationNamed(String aDeclarationName, String aDeclarationType,
                                           String aBindingName, String anAssociationDescription, Object aValue)
{
        WOComponent component = ERXWOContext.currentContext().component();
        if (component.parent() != null) {
                component = component.parent();
        }
        _debugValueForDeclarationNamed(component, " <== ", aDeclarationName, aDeclarationType,
                                       aBindingName, anAssociationDescription, aValue);
}

/**
 * Turns on/off binding debugging for the given component.  Binding debugging requires using the WOOgnl
 * template parser and setting ognl.debugSupport=true.
 *
 * @param debugEnabled whether or not to enable debugging
 * @param componentName the component name to enable debugging for
 */
public void setDebugEnabledForComponent(boolean debugEnabled, String componentName) {
        if (debugEnabled) {
                _debugComponents.addObject(componentName);
        }
        else {
                _debugComponents.removeObject(componentName);
        }
}

/**
 * Returns whether or not binding debugging is enabled for the given component
 *
 * @param componentName the component name
 * @return whether or not binding debugging is enabled for the given componen
 */
public boolean debugEnabledForComponent(String componentName) {
        return _debugComponents.containsObject(componentName);
}

/**
 * Turns off binding debugging for all components.
 */
public void clearDebugEnabledForAllComponents() {
        _debugComponents.removeAllObjects();
}
```

Using Click to Debug

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

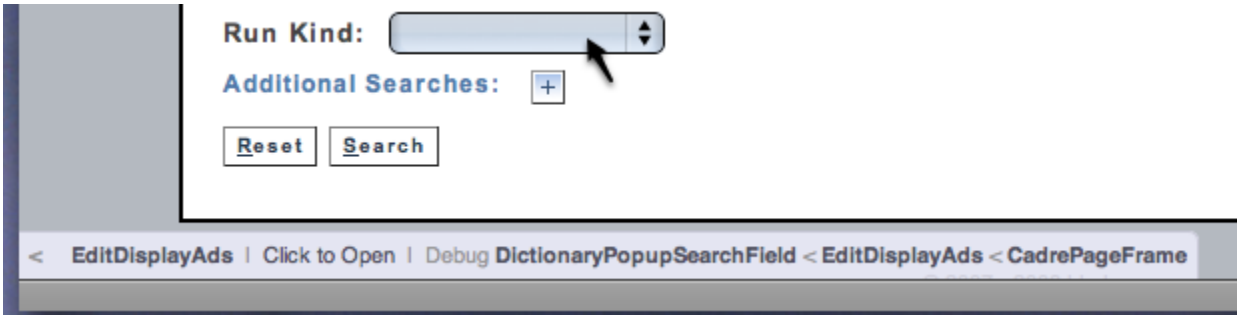
If you don't, check that the page has the WOLToolBar on it and that the `er.component.clickToOpen` property is set to true and the `er.extensions.ERXApplication.developmentMode` property is set to true.

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 Debug** 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 turn binding debug on (or off if it is on) for the component under the mouse in Eclipse. It is that easy!



With binding debug on, you'll get output like:

```
DEBUG NSLog  - HatchViewTaskPage ==> [inline]:HatchEditTask { task=task; }  value <com.mdimension.mdtask.model.
Task pk:"1027787">
DEBUG NSLog  - HatchViewTaskPage <== [inline]:HatchEditTask { task=task; }  value <com.mdimension.mdtask.model.
Task pk:"1027787">
DEBUG NSLog  - HatchViewTaskPage ==> [inline]:HatchEditTask { task=task; }  value <com.mdimension.mdtask.model.
Task pk:"1027787">
DEBUG NSLog  - HatchViewTaskPage <== [inline]:HatchEditTask { task=task; }  value <com.mdimension.mdtask.model.
Task pk:"1027787">
DEBUG NSLog  - HatchViewTaskPage ==> [inline]:HatchEditTask { task=task; }  value <com.mdimension.mdtask.model.
Task pk:"1027787">
DEBUG NSLog  - HatchViewTaskPage <== [inline]:HatchEditTask { task=task; }  value null
DEBUG NSLog  - HatchViewTaskPage ==> [inline]:HatchEditTask { task=task; }  value null
DEBUG NSLog  - HatchViewTaskPage <== [inline]:HatchEditTask { task=task; }  value null
```

Note: A prefix like `Dec 13 11:00:29 MDTask[WOL:62934] (ERXNSLogLog4jBridge.java:46)` was removed from each line above to make this easier to read.

So what that's showing is HatchViewTaskPage component is pushing the task binding into the HatchEditTask component with the value Task 1027787, then HatchEditTask component is pushing the binding back up. So you can see here that 6th line, HatchEditTask component is pushing a null binding back out (which, in this case, was causing a problem I was trying to find).