Your First Stateful Project

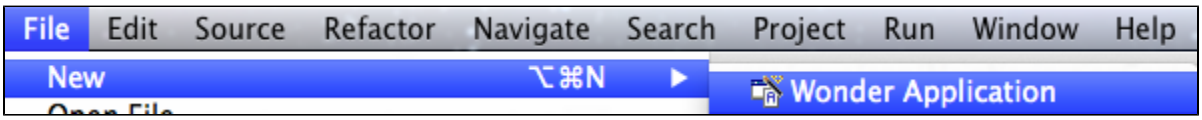
So far, we have seen two of the technologies, D2W and ERRest, that Project Wonder offers for viewing and managing the data. In this tutorial, we will show how to do it with the "stateful" way of doing things. Stateful have been around since the beginning of WebObjects in 1996, so it's the oldest way of presenting data and constructing pages.

Stateful means that you don't have to worry about creating sessions and keeping track of data coming from HTML input fields and controls. In fact, D2W is also stateful.

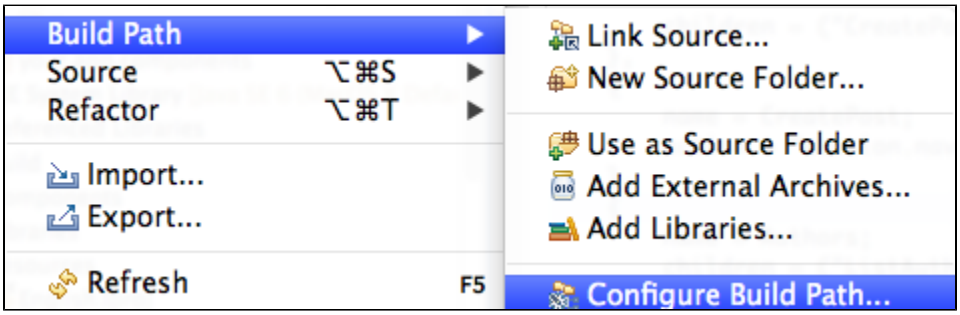
In this tutorial, we are also going to use the Ajax framework, who is stateful too. We will replicate the functionalities of the two other tutorials, but by creating pages ourselves. The application will have the following pages:

- The main page will display a list of blog entries, with a link to see the blog entry.
- The main page will have a link to an "admin" page that will show a login form.
- After login, a list of blog entries with links to edit, delete and create blog entries will be show.
- We need a form to edit/create blog entries.

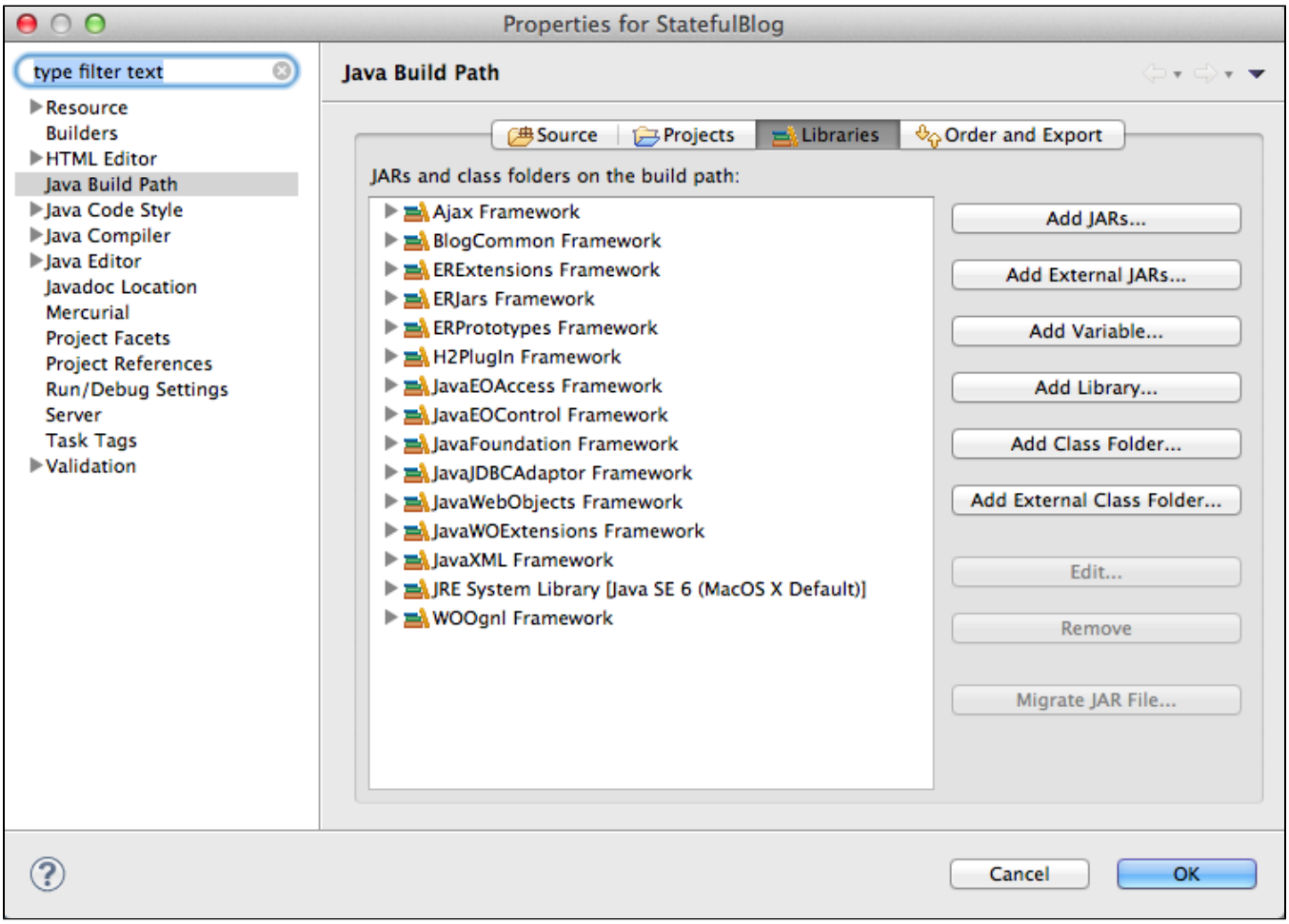
Let's start by creating a new project in Eclipse. You need to create a **Wonder Application** project type, and name it **StatefulBlog**.



Just like the D2W tutorial, you need to link the application with the **BlogCommon**, **Ajax** and **H2PlugIn** frameworks. To do so, right-click on **StatefulBlog** and select **Build Path** -> **Configure Build Path**.



In the **Libraries** tab, click on **Add Library**. Select **WebObjects Frameworks** and click **Next**. Check **Ajax**, **BlogCommon** and **H2PlugIn** from the list and click **Finish**. The **Libraries** tab should look like this:



We are ready to code! Open the **Components** folder of the project, and open **Main WO**. In the **Related** view (bottom-right), you see that all related files of the component are listed, and we need to open the Java code associated with the component. To do so, in the **Related** view, double-click on **Main.java** to open the Java class into an editor.

In **Main.java**, we need some Java code to get the list of blog entries so that we can show that list into the component. The following code will do what we need:

```
import your.app.model.BlogEntry;

import com.webobjects.appserver.WOContext;
import com.webobjects.eoaccess.EODatabaseDataSource;
import com.webobjects.eocontrol.EOEditingContext;

import er.extensions.batching.ERXBatchingDisplayGroup;
import er.extensions.components.ERXComponent;
import er.extensions.eof.ERXEC;

public class Main extends ERXComponent {

  private EOEditingContext _ec;
  private BlogEntry _blogEntryItem;

  public Main(WOContext context) {
    super(context);
    EODatabaseDataSource dataSource = new EODatabaseDataSource(editingContext(), BlogEntry.ENTITY_NAME);
    ERXBatchingDisplayGroup<BlogEntry> dg = new ERXBatchingDisplayGroup<BlogEntry>();
    dg.setNumberOfObjectsPerBatch(20);
    dg.setDataSource(dataSource);
    dg.setObjectArray(BlogEntry.fetchAllBlogEntries(editingContext(), BlogEntry.LAST_MODIFIED.descs()));
  }

  private EOEditingContext editingContext() {
    if (_ec == null) {
      _ec = ERXEC.newEditingContext();
    }
    return _ec;
  }

  public void setBlogEntryItem(BlogEntry blogEntryItem) {
    this._blogEntryItem = blogEntryItem;
  }

  public BlogEntry blogEntryItem() {
    return this._blogEntryItem;
  }

}
```

ERXBatchingDisplayGroup is a subclass of WODisplayGroup, a utility that adds multiple actions and logic to a list of objects. One of the best features of ERXBatchingDisplayGroup is that it does real batching (if the RDBMS that you use supports it), so that means that if we specify a batch of 20 objects (*dg.setNumberOfObjectsPerBatch(20)*), it will fetch only the first 20 objects from the database, and if you switch to the next batch, the display group will go to the database to get the next 20 objects. ERXBatchingDisplayGroup is useful if you known that your list will contains hundred of objects.

Let's edit the component. Open **Main.wo** and edit the content in the top panel to be:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>untitled</title>
  </head>
  <body>
    <h1>Our super blog</h1>
    <wo:if condition="$displayGroup.hasMultipleBatches">
      <div>
        <wo:link action="$displayGroup.displayPreviousBatch">Previous</wo:link>
        | Batch
        <wo:str value="$displayGroup.currentBatchIndex" />
        of
        <wo:str value="$displayGroup.batchCount" />
        |
        <wo:link action="$displayGroup.displayNextBatch">Next</wo:link>
      </div>
    </wo:if>
    <wo:loop list="$displayGroup.displayedObjects" item="$blogEntryItem">
      <p><wo:str value="$blogEntryItem.title" /></p>
      <p><wo:str value="$blogEntryItem.content" escapeHTML="false" /></p>
      <p>Created by
        <wo:str value="$blogEntryItem.author.fullName" />
        on
        <wo:str value="$blogEntryItem.creationDate" />
      </p>
      <hr />
    </wo:loop>
  </body>
</html>
```

The condition *displayGroup.hasMultipleBatches* checks if we have more than 20 objects. If that's true, we will show a HTML div with links to navigate between batches. After that condition, we loop over the displayedObjects (the current batch of objects) and we use WOString elements to display details about each blog entry.

You can run the application to check if everything is working. If everything is ok, terminate the app.

The next steps is to build the administration part of the application. The first thing we need to do is to add a variable and two methods in **Session.java** that will store the logged user. Open **Session.java** and add the following code:

```
private Author _loggedAuthor;

        public Session() {
          this._loggedAuthor = null;
        }

        public Author loggedAuthor() {
          return this._loggedAuthor;
        }

        public void setAuthor(Author loggedAuthor) {
          this._loggedAuthor = loggedAuthor;
        }
```

Save the file. Now open Author.java and add the following method:

```
public static Author validateLogin(EOEditingContext editingContext, String _emailAddress) {
     Author user = Author.fetchAuthor(ERXEC.newEditingContext(), Author.EMAIL.eq(_emailAddress));
     return user;
}
```

Next: we need to add a component to present the login form to the user. Right-click on the **Components** folder in the project, and select **New** -> **WOComponent**. Change the name of the component to be **AdminMainPage** and change the superclass to **er.extensions.components.ERXComponent**.

After the component have been created, open **AdminMainPage.java** and override the content of the class with the following code:

```
package your.app.components;

import your.app.Session;
import your.app.model.Author;
import your.app.model.BlogEntry;

import com.webobjects.appserver.WOActionResults;
import com.webobjects.appserver.WOContext;
import com.webobjects.eoaccess.EODatabaseDataSource;
import com.webobjects.eocontrol.EOEditingContext;

import er.extensions.batching.ERXBatchingDisplayGroup;
import er.extensions.components.ERXComponent;
import er.extensions.eof.ERXEC;

public class AdminMainPage extends ERXComponent {

  private ERXBatchingDisplayGroup<BlogEntry> _dg;

  public AdminMainPage(WOContext context) {
    super(context);
    EODatabaseDataSource dataSource = new EODatabaseDataSource(editingContext(), BlogEntry.ENTITY_NAME);
    _dg = new ERXBatchingDisplayGroup<BlogEntry>();
    _dg.setNumberOfObjectsPerBatch(20);
    _dg.setDataSource(dataSource);
    _dg.setObjectArray(BlogEntry.fetchBlogEntries(editingContext(), BlogEntry.AUTHOR.eq(session().
loggedAuthor()), BlogEntry.LAST_MODIFIED.descs()));
  }

  public ERXBatchingDisplayGroup<BlogEntry> displayGroup() {
    return this._dg;
  }

  private String _emailAddress;

  public String emailAddress() {
    return this._emailAddress;
  }

  public void setEmailAddress(String emailAddress) {
    this._emailAddress = emailAddress;
  }

  private BlogEntry _blogEntryItem;

  public void setBlogEntryItem(BlogEntry blogEntryItem) {
    this._blogEntryItem = blogEntryItem;
  }

  public BlogEntry blogEntryItem() {
    return this._blogEntryItem;
  }

  @Override
  public Session session() {
    return ((Session)super.session());
  }

  public boolean isLogged() {
    return ((session()).loggedAuthor() == null) ? false: true;
  }

  private EOEditingContext _ec;

  public EOEditingContext editingContext() {
    if (_ec == null) {
      _ec = ERXEC.newEditingContext();
```

```
    }
    return _ec;
  }

  private String _errorMessage = null;

  public String errorMessage() {
    return this._errorMessage;
  }

  public WOActionResults login() {
    Author loggedAuthor = Author.validateLogin(editingContext(), _emailAddress);
    if (loggedAuthor != null) {
      session().setAuthor(loggedAuthor);
    } else {
      _errorMessage = "Invalid email address";
    }
    return null;
  }
}
```

You will notice that we are using a ERXBatchingDisplayGroup again. But this time, when we call *_dg.setObjectArray*, we set the array of objects so that only the blog entries created by the logged author are displayed.

Open **AdminMainPage.wo** and override all the content between the <body> tag to be:

```
<wo:AjaxUpdateContainer id="main">
      <wo:if condition="$isLogged">
        <wo:if condition="$displayGroup.hasMultipleBatches">
          <div>
            <wo:link action="$displayGroup.displayPreviousBatch">Previous</wo:link>
            | Batch
            <wo:str value="$displayGroup.currentBatchIndex" />
            of
            <wo:str value="$displayGroup.batchCount" />
            |
            <wo:link action="$displayGroup.displayNextBatch">Next</wo:link>
          </div>
        </wo:if>
        <table>
          <tr>
            <th><wo:WOSortOrder displayGroup="$displayGroup" key="title" /> Title</th>
            <th>Author</th>
            <th><wo:WOSortOrder displayGroup="$displayGroup" key="creationDate" /> Created on</th>
            <th><wo:WOSortOrder displayGroup="$displayGroup" key="lastModified" /> Last modified</th>
          </tr>
          <wo:loop list="$displayGroup.displayedObjects" item="$blogEntryItem">
            <tr>
              <td>
                <wo:str value="$blogEntryItem.title" />
              </td>
              <td> <wo:str value="$blogEntryItem.author.fullName" /> </td>
              <td> <wo:str value="$blogEntryItem.creationDate" dateformat="%Y/%m/%d" /> </td>
              <td> <wo:str value="$blogEntryItem.lastModified" dateformat="%Y/%m/%d" /> </td>
            </tr>
          </wo:loop>
        </table>
      </wo:if>
      <wo:else>
        <wo:if condition="$errorMessage">
          <span style="text-color: red;">Error: <wo:str value="$errorMessage" /></span>
        </wo:if>
        <wo:form>
          <div>
            <label>Email address:</label>
            <wo:textField value="$emailAddress" />
          </div>
          <div><wo:AjaxSubmitButton updateContainerID="main" action="$login" value="Login" /></div>
        </wo:form>
      </wo:else>
    </wo:AjaxUpdateContainer>
```

Last step: we need a link to the admin page. Open **Main.wo** and just before the </body> tag, add the following:

```
<wo:link pageName="AdminMainPage">Admin</wo:link>
```

Save everything, run the app and try to login. If login is not successful, you will get an error message. If login is valid, you will see the blog entries that you created.

For the last part of this tutorial, we are going to add a link on each blog entry in the list that will bring us to a edit page where we can modify a blog entry. We are also going to add a link to create a new blog entry.

Create a new component, and name it **EditBlogEntry**. Open **EditBlogEntry.java** and override the code with:

```
package your.app.components;

import your.app.Session;
import your.app.model.Author;
import your.app.model.BlogEntry;

import com.webobjects.appserver.WOActionResults;
import com.webobjects.appserver.WOContext;
import com.webobjects.eocontrol.EOEditingContext;

import er.extensions.components.ERXComponent;
import er.extensions.eof.ERXEC;
import er.extensions.eof.ERXEOControlUtilities;

public class EditBlogEntry extends ERXComponent {

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

  private BlogEntry _blogEntry;

  public BlogEntry blogEntry() {
    return this._blogEntry;
  }

  public void setBlogEntry(BlogEntry blogEntry) {
    if (blogEntry == null) {
      this._blogEntry = ERXEOControlUtilities.createAndInsertObject(editingContext(), BlogEntry.class);
      Author localUser = ERXEOControlUtilities.localInstanceOfObject(editingContext(), session().
loggedAuthor());
      this._blogEntry.setAuthorRelationship(localUser);
    } else {
      this._blogEntry = ERXEOControlUtilities.localInstanceOfObject(editingContext(), blogEntry);
    }
  }

  private EOEditingContext _ec;

  public EOEditingContext editingContext() {
    if (_ec == null) {
      _ec = ERXEC.newEditingContext();
    }
    return _ec;
  }

  @Override
  public Session session() {
    return ((Session)super.session());
  }

  public WOActionResults save() {
    editingContext().saveChanges();
    return pageWithName(AdminMainPage.class);
  }
}
```

Open **EditBlogEntry.wo** and between the <body> tag, add the following:

```
<wo:form>
      <div>
        <label>Title:</label>
        <wo:textfield value="$blogEntry.title" />
      </div>
      <div>
        <label>Content:</label>
        <wo:text value="$blogEntry.content" rows="20" cols="80" />
      </div>
      <div>Author: <wo:str value="$session.loggedAuthor.fullName" /></div>
      <div><wo:submitButton action="$save" value="Save changes" /></div>
    </wo:form>
```

We now have a form to edit or create a blog entry. Save the component and the Java class, and open **AdminMainPage.java** to add the following code:

```
public WOActionResults editBlogEntry() {
    EditBlogEntry nextPage = pageWithName(EditBlogEntry.class);
    nextPage.setBlogEntry(_blogEntryItem);
    return nextPage;
  }

  public WOActionResults createBlogEntry() {
    EditBlogEntry nextPage = pageWithName(EditBlogEntry.class);
    nextPage.setBlogEntry(null);
    return nextPage;
  }
```

Open **AdminMainPage.wo** and just after <wo:if condition="$isLogged">, add the following line:

```
<div><wo:link action="$createBlogEntry">Create a new blog entry</wo:link></div>
```

Find this line:

```
<wo:str value="$blogEntryItem.title" />
```

and replace it with:

```
<wo:link action="$editBlogEntry"><wo:str value="$blogEntryItem.title" /></wo:link>
```

Save everything, run the app, click on the "admin" link, login and check if you can create or edit a blog entry. Everything should be working, and just created your first stateful Project Wonder application! It's time to deploy an application.