Mirage 2 Configuration and Customization

Introduction

Mirage has been the default XMLUI theme since DSpace 1.7 and has been used as base point for most custom themes. DSpace 5 includes Mirage 2, also developed by @mire, an upgrade to Mirage built on modern web technologies. The predominant improvement for the end user is the responsive design. Repository admins and developers will also benefit because of the tools available to make both simple and advanced customizations.

Responsive design

A responsive website is a website that rearranges its content to fit in different screen sizes. The main focus is to provide a better overall user experience whether you're browsing on a mobile phone, a tablet or desktop computer. As opposed to using a separate mobile theme, there's only one version of the website that will work everywhere. A simply way to find out what the differences are between a narrow screen and a larger screen in Mirage 2, go to any webpage and resize the browser window. You will notice the sidebar is only shown when the window is large enough, otherwise a menu button is displayed to get to the sidebar options. The theme provides a distinct look for each of the 3 different categories of screen sizes: mobile, tablet and desktop.

blocked URL

The build process and enabling Mirage 2

The modern web technologies that power Mirage 2 include a precompiler (Compass), a package manager (Bower) and a task runner (Grunt). These tools can only be installed when some prerequisites are present on the system. DSpace's Maven build process is capable of making a temporary installation of these dependencies just so the theme can be built. However the overall build time will be significantly shortened if these dependencies are manually installed on the system (see below for more info).

All of the Mirage 2 builds require git. Make sure to install git before starting any of the Mirage 2 builds.

- By default, DSpace does *not* build the Mirage 2 theme (as it lengthens the normal build process). However, you can easily tell DSpace to build Mirage 2 by running the following from your `[dspace-source]` directory:

```
mvn package -Dmirage2.on=true
```

    - If you wish to speed up the Mirage 2 build process, you can do so by pre-installing all of the Mirage 2 dependencies on your system (by default they will be downloaded each time you rebuild Mirage 2). This will *significantly shorten* the build process for Mirage 2. More information on installing these prerequisites can be found in the Developer Documentation for Mirage 2. Once these prerequisites have been installed on your local server, you can then build Mirage 2 more rapidly by running:

```
        # WARNING: This command will only work if you've manually installed *all* the
prerequisites for Mirage 2
mvn package -Dmirage2.on=true -Dmirage2.deps.included=false
```

- use a Mirage 2 build to fail. See: Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- nply re-running Ant from `[dspace-source]/dspace/target /dspace-installer/`:

```
ant update
```

- To enable Mirage 2, add the following to the `<themes>` section of your `xmlui.xconf`, replacing the currently active theme:

```
<theme name="Mirage 2" regex=".*" path="Mirage2/" />
```

- Finally, restart your Tomcat or servlet container, and you should see the Mirage 2 theme.

NOT building with Mirage2

As you get used to building with the mirage2.on=true property, if you ever need to again build without the Mirage2 theme enabled (for example, if you wish to test functionality not associated with the Mirage2 theme), you might be tempted to "turn off" Mirage2 building by treating the mirage2.on property as a flag, and setting it to false. However, if you look at the Maven pom.xml files, you'll see that the value of the property is never checked, just the *existence* of it is important. If you wish to build without Mirage2, the easiest thing to do is to simply omit the mirage2.on property from your mvn command. If you'd really like to ensure the mirage2 profile is not used, you can explicitly disable the dspace-mirage2 profile with:

```
mvn package -P-dspace-xmlui-mirage2
```

Common Build Issues

- Running the Mirage 2 build (`mvn package -Dmirage.on=true`) as the "root" user (or via sudo) will result in the following error from "Bower". This will result in a broken Mirage 2 build. The fix is to ensure you are building DSpace as a **non-root** user account. For more information on this Bower error, see: http://serverfault.com/questions/548537/cant-get-bower-working-bower-esudo-cannot-be-run-with-sudo

```
bower ESUDO   Cannot be run with sudo

Additional error details:
Since bower is a user command, there is no need to execute it with superuser permissions.
```

- The Mirage 2 build requires git. Ensure that git is installed before you launch the Mirage 2 build.
- The Mirage 2 build process will attempt to retrieve some dependencies from GitHub via the "git" protocol. This requires outgoing access to github.com, port 9418. If the machine on which you're running the build has access restrictions in place for that port but outgoing access via HTTPS (port 443) is allowed, you can substitute the https protocol by running (with the same user account that will run the maven step):

```
git config --global url."https://github.com/".insteadOf git://github.com/
```

on on this issue, see Unable to locate Jira server for this macro. It may be due to Application Link configuration.

- ll the build dependencies locally, see Unable to locate Jira server for this macro. It may be
- Application Link configuration. . The problem was fixed for DSp Unable to locate Jira server for this macro. It may be due to
- may need extra quotes

```
mvn package '-Dmirage2.on=true'
```

Configuration options

Mirage 2 adds two configuration options to dspace.cfg that affect the rendering of bitstream labels on item pages:

As an administrator, you can choose between displaying the file name (title) or the description (label). Because bitstream description is an optional value, you can also define a fallback value. The default configuration will use the label as the first choice, and fall back to the title field.

```
    ### Settings for the Item page in Mirage2 theme ###
# Whether the title or the label of a file should be used to display it on the item page
mirage2.item-view.bitstream.href.label.1 = label

# Whether the title or the label of a file should be used as a fallback to display it on the item page
mirage2.item-view.bitstream.href.label.2 = title
```

There are other configuration properties that affect the theme. These aren't new but we mention them here for the sake of completeness.

- When METSRIGHT is included in `plugin.named.org.dspace.containt.crosswalk.DisseminationCrosswalk` the item page will display those rights.
- The property `xmlui.theme.mirage.item-list.emphasis` defines the style of the item lists. When the value is 'file' another style is used.
- The property `webui.browse.render-scientific-formulas` includes a javascript library to render scientific formulas.
- The properties `thumbnail.maxheight` and `thumbnail.maxwidth` define the outer bounds for the dimensions of the item thumbnails in the item lists.

Customizing Mirage 2

Do not attempt the following

Do not attempt to manage local customizations to Mirage 2 in:

- `[dspace-source]/dspace/modules/`**`xmlui`**`/src/main/webapp/themes`
  - This is where you would put standard XMLUI Themes or customizations. However, because of the Mirage 2 build process, this won't work.
- `[dspace-source]/dspace-xmlui-mirage2`
  - This is the place where the community, committers and contributors manage the STANDARD version of Mirage 2. You could change files there if your intention is to create a contribution that would benefit everyone. But in this case, we are not talking about a local customization.

Recommended approach

Manage your local Mirage 2 customizations or derived themes in:

`[dspace-source]/dspace/modules/`**`xmlui-mirage2`**

Managing your local customizations in this folder comes with the advantage that you ONLY need to keep files you have changed, compared to the standard Mirage 2 folder. To get you started, the contributors have added a `_style.scss` file where you can make local scss customizations:

dspace/modules/xmlui-mirage2/src/main/webapp/themes/Mirage2/styles

The Mirage 2 color scheme

The style sheet of Mirage 2 is written in sass and relies on the bootstrap framework. A big advantages of this is the ease of changing the color scheme. By default Mirage 2 has the colors that are familiar from the classic Mirage theme, but another color scheme with only the standard bootstrap colors is also ready and available. In fact this color scheme can be activated by building DSpace using one extra maven profile, `mirage2_bootstrap_color_scheme`.

The classic mirage theme is a customization of the bootstrap theme. Thanks to the sass variables, a complete color scheme can be conceived by modifying one or two variables. These variables are set in the theme's `/styles/classic_mirage_color_scheme/_bootstrap_variables.scss`. Copy this file into `[dspace-source]/dspace/modules/xmlui-mirage2/src/main/webapp/themes/Mirage2` and see what happens when you change `$brand-primary`. More detailed information on how to customize this file can be found in the Mirage 2 readme.

How to reuse an existing bootstrap theme is also explained in that section.

Simple styling customization

Simple customizations imply that they only require custom css, e.g. changing the font or the logo. The theme's `_style.scss` file is the right place for this. All the lines of css in this file will be included in the theme's style sheet. Even though it's a file with the scss extention, the usual lines of css will work just as well.

Advanced styling customizations

For guidelines on how to include javascript, please read the the Additional Developer documentation.

Automatically retrieving the latest versions of Mirage 2 dependencies

Mirage 2 dependencies are automatically pulled in during the Bower step of the build process. For official DSpace releases, the committers lock the dependencies on a specific version in order to make the behaviour of the theme predictable.

For development purposes however, it is recommended that set the dependencies to "latest" so you can benefit from the most recent updates and bugfixes in Mirage 2's dependencies.

You can make these changes in the bower.json file: bower.json

As mentioned in the previous section, make sure you manage this file and any changes you make to it in `[dspace-source]/dspace/`**`modules`** (e.g. `[dspace-source]/dspace/modules/xmlui-mirage2/src/main/webapp/themes/Mirage2`). It is not recommended to update the officially distributed bower.json file directly in `[dspace-source]/dspace-xmlui-mirage2`

Additional Developer documentation

Specific guidelines and technical details about Mirage 2 are part of the Readme.MD file in the Mirage 2 sourcetree.