IDE Integration - DSpace and NetBeans

Instructions for DSpace 1.5 and above

- These instructions assume you are using DSpace 1.5.x (or higher) in the NetBeans IDE.
- DSpace 1.4.x and previous are not covered.

Table of Contents:

Video Walkthrough

João Rocha da Silva kindly posted a video walkthrough of setting up DSpace + Netbeans on YouTube:

Installing NetBeans

NetBeans 7.1 (or latest version) is recommended for both performance and features (plus, >7.1 includes Git support out-of-the-box). Download it from http://www.netbeans.org/.

Choose either the 'Java SE' or 'Java' version. The 'Java' version has more features (such as editing JSPs), and although the GlassFish / Tomcat runtimes will be downloaded, you don't have to install them. If you choose the 'Java SE' version, you can always install those extra features later (as plugins).

Now simply run the downloaded installer.

If you are running on Linux/Unix, you will need to make the downloaded script executable (e.g. `chmod +x netbeans-[version]-full-linux.sh`).

Git Plugin / Support

As long as you are running NetBeans 7.1 or above, Git support comes pre-installed in NetBeans.

However, if you are running NetBeans 7.0, you can install the "Git" plugin by going to "*Tools* -> *Plugins*" and finding the plugin named "Git" under the list of "Available Plugins".

If you plan to also occasionally run Git commands from your command-line, you may wish to install Git on your operating system:

- Git Homepage (includes info on downloading & installing on major platforms)

Using a Local Subversion Repository instead?

If your institution plans to instead use a local Subversion repository for your local development, you can do so (you will just need to download the DSpace Source Code and import into your local subversion repository).
Subversion support is already included in NetBeans. However, it requires that the command line client is installed and available on your local machine.

- For most Linux distributions, you should be able to just install the Subversion client available in your distribution's repositories.
- For Windows, there is now a 'Bundled Subversion Client for Windows' Plugin which you can install from within NetBeans
    - Go to *Tools* -> *Plugins* to install the plugin.
- For Windows/RedHat/Solaris, CollabNet also offers free Subversion downloads that are specifically tested with NetBeans: http://www.collab.net/downloads/netbeans/ (Free to download, but they require that you register first. Registration is also free, but requires a valid email address)

Maven Support

As long as you are on a recent version of NetBeans (> 6.7), Maven support is included out-of-the-box.

Integrate an External Installation of Maven

Although not required, it is *recommended* to install an external version of Apache Maven. This will ensure that you are using a version of Maven which DSpace supports. (You should check the latest pre-requisites if you are using a later version of DSpace) However, if the version of Maven that is bundled with NetBeans is a supported version, you are welcome to use it.

If you wish to integrate an external install of Maven:

- First, install Maven:
  - For most Linux distributions, you should be able to just install the Maven client available in your distribution's repositories.
  - For all other operating systems, you can install the latest version from the Apache Maven site.
- Configure Netbeans to use your external Maven:
  - Go to the "*Tools*" menu, and select "*Options*" (on OS/X, this is *"NetBeans" -> "Preferences"*)
  - Select "*Java*" section.
  - Select the "*Maven*" tab.
  - Modify the "*Maven Home*" field. It's possible that Netbeans will already see your external version of Maven (if it's in your PATH). However, if it says that it is using the "Bundled" Maven, then you'll want to fill out the "*Maven Home*" field with the location of your external installation of Maven.
  - In the middle of this page, you may also need to fill out the location of your "*Local Repository*". This should be the location of the ".m2" directory under your user's home directory.
    - On Linux and OS/X, it should be located at `~/.m2/`
    - On Windows, it should be located at `C:\Documents and Settings\username\.m2` (Windows XP) or `C:\Users\username\.m2` (Windows 7)
  - All the other Maven settings should be fine as their defaults. Press "*OK*" to save your changes.

Checkout DSpace from GitHub

NetBeans Git Guide

For more generic information about working with Git/GitHub in NetBeans, see the NetBeans Git Guide

Need a tutorial on Git/GitHub?

If you need help/tips/resources on DSpace development with Git/GitHub, or just tutorials on Git in general, you may want to check out our DSpace Development with Git page.

NetBeans makes the checkout and configure process easy, as it does nearly everything for you.

- Under the '*Team*' menu, go to '*Git*' -> '*Clone*'. (In Git terminology, "clone" just means you are downloading an exact copy of the remote code repository to your local machine.) Enter the URL for the DSpace Repository in GitHub which you wish to work with (e.g. https://github.com/DSpace/DSpace.gitOR if you have your own fork: https://github.com/\[your-username\]/DSpace.git), optionally enter in a GitHub username/password, and click on the "Next" button.
  - ***IMPORTANT NOTE:*** If you plan to do a larger amount of DSpace development or local changes, you may wish to first "fork" the DSpace GitHub Repository (https://github.com/DSpace/DSpace) to your own GitHub account. This will create your own copy of the DSpace source code under your GitHub account (e.g. https://github.com/\[your-username\]/DSpace). You can then checkout your own forked repository to work from and commit local changes to (push changes to). For more information, see the GitHub help page on "Forking a Repo".
- Next, select the "Remote Branch" you wish to develop on. A few hints:
  - Branches named "dspace-#_#_x" (e.g. dspace-1_8_x) are Bug Fix / Maintenance branches. So, the latest code in that bug-fix or maintenance release will be available on that branch. This code tends to be more stable overall. **As such, we recommend most developers use the appropriate Bug Fix / Maintenance branch for their local development.**
  - The branch named **master** is roughly equivalent to the old SVN Trunk. As such, it may not be as stable, but it includes the latest & greatest code which is being prepared for the next major release. *Unless you know what you are doing, we do NOT recommend running this code in Production. It is essentially "unsupported" until it is officially released.*
  - If you wish to work from a "tagged" (official release) version of DSpace (e.g. 1.8.2), you can download those releases as Tarballs/Zips from: https://github.com/DSpace/DSpace/tags You could then use that Tarball/Zip to import it into your own Git/GitHub or SVN repository as you see fit.
- Next, choose a local parent folder to use and the "Clone Name" (actual folder name for the source code), and leave the 'Scan for NetBeans projects after Clone' option selected, and click on 'Finish'. (All other options you should be able to leave as their default values.)
- When NetBeans completes the clone, it will pop-up a dialog telling you that it found a number of DSpace projects (Maven projects) during the checkout. Choose 'Open Project' from the dialog. Select all the projects (and/or subprojects) that you wish to open (hold down **ctrl** or **shift** to select multiple), and click the "Open" button. (Don't worry you can always Open or Close projects later if you notice you opened up too few or too many)

After the clone has completed, you'll notice NetBeans considers each DSpace Maven "module" to be a separate project. So, you'll see separate projects for "DSpace XML-UI" and "DSpace JSP-UI", even though these are all cloned from the same source code.

Git/GitHub Hints & Tips

The following are a few hints/tips which you may want to utilize to ease your development processes with NetBeans and GitHub. Much more info about DSpace + GitHub development at Development with Git.

1. **Fork your own Repo to store your local changes:** As recommended above, you really should think about forking your own copy of the DSpace GitHub repository. As GitHub describes in their "Fork a Repo" guide, forking lets you create your own personal copy of the codebase. It not only provides you a place to put your local customizations. It also provides an easier way to contribute your work back to the DSpace community (via a GitHub Pull Request).

2. **For easier Fetch/Merge, setup an "upstream" repository location:** This is only really relevant if you have your own personal "fork" (see #1). If you have forked the DSpace GitHub repository, then you may want to setup an "upstream" remote that points at the central DSpace GitHub repository. This is described in more detail in the GitHub "Fork a Repo"guide. Perform the following:
    - On the command-line, change directory to your local machines' cloned DSpace git repository, which is also your DSpace source directory (e.g. `cd [dspace-src]`)
    - Run the following 'git' command from that directory:

      ```
      git remote add upstream git://github.com/DSpace/DSpace.git
      ```

      (Technically you can name it something other than "upstream". But, "upstream" is just the GitHub recommended naming convention).
    - For more information about how this comes in handy, see the section below on "#Fetch & Merge Example via NetBeans".

Fetch & Merge Example via NetBeans

This assumes you've followed the #Git/GitHub Hints & Tips listed above, and have forked your own personal copy of DSpace's GitHub as well as setup an "upstream" remote link. **This is just one example of how you can perform these tasks.**

1. **Fetch changes from DSpace Main GitHub:**You fetch (and later merge) changes that have occurred in the central DSpace GitHub Repository:
    - *From NetBeans*: Right-click on the "DSpace Parent Project" (root project) and then select: *Git -> Remote -> Fetch*. This will pop up a window that will allow you to easily select the "upstream" configured repository to fetch the latest changes from, and allow you to choose the "master" branch to apply them to. Once you click "Finish", a new "upstream/master" branch will be created locally with the latest changes to be merged.
    - *From Command-line* in your DSpace source directory (e.g. `cd [dspace-src]`): `git fetch upstream`
2. **Merge changes into your Local Git Repo:**Remember, "fetching" changes just brings them into your local-machine's copy of the Git repository. You'll then need to merge those changes with yours and push the changes back to your personal public GitHub repository.
    - *From NetBeans*: Right-click on the "DSpace Parent Project" (root project) and then select: *Git -> Merge Revision*. This will pop up a window to let you select which "branch" to merge into your currently checked out code. If press "Select", you'll see a new branch called "upstream/master" under "Branches -> Remote". Selecting that branch will merge the latest code from "upstream/master" into whatever branch you currently are working with (e.g. "master").
    - *From Command-line* in that same directory: `git merge upstream/master`
3. **Quick Status of Local Git Repo:**If you want to see what happened, you can look at the "Status" information:
    - *From NetBeans*: Right-click on the "DSpace Parent Project" (root project) and then select: *Git -> Show History*. Click the "Search" button (without entering any search info). It will bring back results that will show you where the HEAD pointer is (latest commit in your local machine's git repo) versus where the 'origin/master' is (latest commit in your personal GitHub repo).
    - *From Command-line* in that same directory: `git status` (will tell you how many "commits ahead" of 'origin/master' you now are)
4. **Push Merged Code up to your Personal GitHub Repo:**Finally, assuming all went well, you can push your changes back up to GitHub into your public personal repository:
    - *From NetBeans*: Right-click on the "DSpace Parent Project" (root project) and then select: *Git -> Remote -> Push*. This will pop up a window that will allow you to select the "origin" repository (your personal fork in GitHub), and allow you to choose the "master" branch to push to.
    - *From Command-line* in that same directory: `git push origin master`

Build DSpace from NetBeans

Two Ways to Build DSpace

As of DSpace 1.8, there are now two options to building DSpace Source Code as detailed in the Advanced Customisation Documentation. They are as follows:

- **Full Build:** Running `mvn package` from the root `[dspace-src]` directory (Or run "Build" on the "DSpace Parent Project" in NetBeans). This option will rebuild all DSpace modules from their Java Source code, then apply any Maven WAR Overlays. In other words, all subdirectories of `[dspace-src]` are recompiled/rebuilt.
- **Quick Build:** Running `mvn package` from the `[dspace-src]/dspace/` directory (Or run "Build" on the "DSpace Assembly and Configuration" project in NetBeans). This option performs a "quick build". It does not recompile/rebuild all DSpace modules. All it does is rebuild and re-apply any Maven WAR Overlays to the previously compiled source code. In other words, the ONLY code that will be recompiled/rebuilt is code that exists in `[dspace-src]/dspace/modules/*` (the Maven WAR Overlay directories)

As described in the note above, as of DSpace 1.8, there are two options to build DSpace. Traditionally (before 1.8), you could only build DSpace from the *dspace* sub-folder (e.g. `[dspace-src]/dspace`). In NetBeans, this project is named "DSpace Assembly and Configuration", based on the name specified in its Maven configuration file (pom.xml).

If you do not see the proper project opened (NetBeans may not have opened it by default), then open it manually:

- Right click in your "Projects" window
- Select "Open Project", and browse to either the `[dspace-src]` OR `[dspace-src]/dspace/` directory (based on the build option you wish to use). You should see the Project Name (on the right) specified as either "DSpace Parent Project" or "DSpace Assembly and Configuration", respectively.
- Click the "Open Project" button.

Once that project is opened, you can build DSpace by doing the following:

1. Right-click on the project
2. Select the "Build" option (alternatively you may select "Clean & Build" to first clean out previous builds).
   You should be able to watch the status in the "Output" window at the bottom of NetBeans. The end result is that DSpace is built into the `[dspace-src]/dspace/target/dspace-[version]-build.dir/` directory (you can verify this from the "Files" window in NetBeans, if you wish).

After building DSpace for the first time, you may still see red exclamation point icons (!) next to some projects. In most cases, this is caused by NetBeans being unable to locate some of the DSpace third-party dependencies on your local file system. To fix this problem, do the follow for **each project** which has a red icon next to it:

- Right click on the project
- Select "Show and Resolve Problems..." (near bottom of pop-up menu)
- Click on "Download Libraries" (assuming the problem is that "Some dependency artifacts are not in the local repository").
  NetBeans should then use Maven to find all the DSpace dependencies and download them to your local Maven repository (in your user's `~/.m2/` folder)

By default NetBeans builds using "mvn install" instead of "mvn package"

By default, NetBeans builds all Maven-based projects using `mvn install` instead of `mvn package` (which DSpace recommends). The resulting build is identical, except that the "install" command will take longer as it also attempts to verify that all source file headers match our DSpace license. If you'd like to speed up your NetBeans build & avoid these license checks you can do the following:

1. Right click on the project you are attempting to Build (e.g. "DSpace Assembly and Configuration"), and select "Properties"
2. Go to the "Actions" category on the left
3. Select the "Build project" action.
4. In the "Execute goals" change "install" to "package"
5. Now select the "Clean and Build Project" action
6. In the "Execute goals" change "clean install" to "clean package"
7. Finally, select the "Build with Dependencies" action
8. In the "Execute goals" change "install" to "package"
   Now, for that project, each time you run any of the 3 build commands (Build, Clean & Build, or Build with Dependencies) NetBeans will use the Maven 'package' command instead of the 'install' command. You'd have to do this customization for any NetBeans project that you want to build using those commands.

As an alternative, you could also create your own custom NetBeans commands (which can be run across any/all projects) by doing the following:

1. Right click on **any** project.
2. Select "Custom -> Goals..."
3. In the "Goals:" field type "package"
4. Select "Remember as:" (at bottom) and enter in "Package" (or some name you will remember)
5. Click OK
   Now, you can right click on any project, and go to "Custom -> Package" to run a Maven "package" of that project. You can use this new custom command instead of the default "Build" command.

Install DSpace normally

After building DSpace, we need to use Apache Ant to install it. Unfortunately, this is an area where NetBeans is not very helpful (as we built DSpace using Maven, it will continue to assume all of our projects are Maven-based projects).

You have two options here:

1. The easiest way to install DSpace may be to just follow the normal installation procedure in the DSpace System Documentation. NetBeans doesn't seem to have an efficient way to perform this installation, so it's easiest to just do it from the command line, as normal.
2. Alternatively, you can use NetBeans to run the `ant fresh_install`command as follows: (You need to create a database and a database owner first.)
   - Go to your start browser, open pgAdmin (Assuming you install PostGreSQL)
   - Connect to PostGreSQL 9.0
   - Right click login roles, select new login role
   - Type dspace for the following fields: Role name/ password/ password again
   - Click Ok
   - Right click database, select new database
   - Type dspace in the following field: Name. Select dspace from dspace owner field
   - Click Ok
   - Return to project folder in Netbeans
   - Click over to the "Files" tab in the left hand window
   - Browse under the "DSpace Assembly and Configuration" project. You are looking for the `target/dspace-[version]-build.dir/build.xml` Ant configuration file.
   - Right click on the `build.xml`file and select "Run Target" => "fresh_install"
     - The above command will perform a fresh install of DSpace based on the settings specified in your `target/dspace-[version]-build.dir/config/dspace.cfg` file.
     - The default installation url is C:\dspace.
   - After installation, open cmd
   - Navigate to your installation folder\bin. Default is C:\dspace. Example command: cd C:\dspace\bin
   - Type the following command: dspace.bat create-administrator
   - Key in the information specified by the command prompt instructions

Run DSpace from NetBeans with Tomcat

Once you have DSpace installed, you can run any DSpace web application (XMLUI, JSPUI, SWORD, LNI, etc.) from NetBeans after you integrate NetBeans with Apache Tomcat.

Integrate an External Installation of Tomcat

If you have Tomcat installed separately from NetBeans, you'll need to tell NetBeans where it's located.

- First, you'll need to install the "*Tomcat*" plugin for NetBeans, if it isn't already installed.
  - Go to the "*Tools*" menu and select "*Plugins*".
  - Select the "*Available Plugins*" tab. This should list an entry for "*Tomcat*" or "Java Web Applications"
  - Check the box next to it, and click on 'install'.
  - Restart NetBeans
- You'll now want to tell NetBeans where your Tomcat installation is located.
  - Go to the "*Tools*" menu and select "*Servers*".
  - Click the "Add Server.." button to add a new server
  - Select the type of Server (e.g. Tomcat 6.0/7.0) and click "Next >"
  - NetBeans will ask you for the "*Catalina Home*" location of this Tomcat Server. This is the location where Tomcat is installed (e.g. On Windows it may be "C:\Program Files\Apache Software Foundation\Tomcat-6.0"). You'll also need to give NetBeans your credentials for the "manager" role in Tomcat (which can be configured in the tomcat-users.xml configuration file).
- After configuring your Tomcat 6.0 server, you may want to reconfigure a few default settings.
  - Go to the "*Tools*" menu and select "*Servers*".
  - Select your "*Tomcat 6.0*" server.
  - On the "*Connection*" tab, you may wish to enable the *HTTP Monitor* option
    - This will start up HTTP Monitoring (in a new tab) in NetBeans, whenever you start this server. HTTP Monitoring may be useful to developers who wish to view all HTTP requests/responses that occur during their development process.

Run DSpace WebApp using Tomcat Integration

Once NetBeans knows about Tomcat, you can run your DSpace webapps through this Tomcat integration and even perform debugging of your DSpace web application. To do this, we'll need to perform some basic configuration of the web application project in NetBeans.

- Select the project for the Web Application you wish to run through Tomcat (e.g. "*DSpace XML-UI (Manakin)*" for the XMLUI). This project **must** be a "*war*" based project, as Tomcat only runs WAR files.
- Right click on your selected project and click "*Properties*"
- From the Properties window, select the "*Run*" category.
  - From these Run settings, you'll want to specify the Server which this application should run on. Select your newly configured Tomcat server.
  - Also, specify a "*Context Path*". This should be the ending path on the URL. For example, specifying "/xmlui" will mean your web application will be available from "http://localhost:8080/xmlui"
- Now, click over to the "*Actions*" category on the left.
  - You'll see a list of Actions on the right. Click on the "*Run Project*" action (as this is the one used to run your web application).
  - In that Action's "*Set Properties*" section, add a property to point it to your DSpace installation. The following is an example on Windows, assuming that you've installed DSpace to "*C:/dspace*":
    - For DSpace 3.0 or above, set the property: `dspace.dir=C:/dspace` (make sure to include this entire line – also, do NOT include quotes around the file path)
    - For DSpace 1.8.x or lower, set the property: `dspace.config=C:/dspace/config/dspace.cfg` (make sure to include this entire line – also, do NOT include quotes around the file path)
- Finally, click "*OK*" at the bottom to save all your new project settings.

Now, test it out!

Right click on the project, and select "*Run*". This should re-build the project, start-up Tomcat, and open up your application in your default web browser.

BONUS: If you haven't already noticed, NetBeans will auto-rebuild your project in real-time while Tomcat is running. This means, if you want to edit a CSS or XSLT (for the XMLUI), you can edit it and just refresh your web browser. Your changes should show up in the browser almost immediately. (That being said, there are times when you will still have to restart Tomcat in NetBeans – usually after you haven't restarted it in a long time, or after an update to Java source code)

Debug DSpace WebApp using Tomcat Integration

Debugging a DSpace web application involves mostly the same setup as running it. The main difference here is that you need to configure the "*Debug Project*" settings (rather than the "*Run Project*" settings). So, similar to above, do the following:

- Select the project for the Web Application you wish to run through Tomcat (e.g. "*DSpace XML-UI (Manakin)*" for the XMLUI). This project **must** be a "*war*" based project.
- Right click on your selected project and click "*Properties*"
- From the Properties window, select the "*Actions*" category on the left.
  - You'll see a list of Actions on the right. Click on the "*Debug Project*" action (as this is the one used to debug your web application).
  - In that Action's "*Set Properties*" section, add a property to point it to your DSpace installation. Make sure to keep all existing properties in tact, and just add your new property. The following is an example on Windows, assuming that you've installed DSpace to "*C:/dspace/*":
    - For DSpace 3.0 or above, set the property: `dspace.dir=C:/dspace` (make sure to include this entire line – also, do NOT include quotes around the file path)
    - For DSpace 1.8.x or lower, set the property: `dspace.config=C:/dspace/config/dspace.cfg` (make sure to include this entire line – also, do NOT include quotes around the file path)
- Finally, click "*OK*" at the bottom to save all your new project settings.

Now, we'll start our web application in Debug-mode. (Make sure to stop Tomcat first, if it is currently running)

- Right click on the project, and select "*Debug*". This should re-build the project, start-up Tomcat in debug-mode, and open up your application in your default web browser. You should see a "Debugger Console" appear.
- You can now add breakpoints to areas of your code. The debugger should automatically stop at those points and let you step through your code line-by-line.
- **Note:** Occasionally, the first time you perform debugging, the debugger doesn't connect properly with your Tomcat Server. If you find it's *not* stopping at your breakpoints, you may wish to "Attach" the debugger manually:
    - From the "*Debug*" menu, select "*Attach Debugger..*"
        - For the "*Connector*", specify "SharedMemoryAttach".
        - For the "*Name*", specify "tomcat_shared_memory_id" (without the quotes).
        - Click "*OK*" to save these settings
    - Finally, verify that Tomcat is specifying this "tomcat_shared_memory_id" field.
        - Go to the "*Tools*" menu and select "*Servers*"
        - Click on your Tomcat Server, and visit the "*Startup*" tab.
        - Make sure the "*Shared Memory Name*" setting is selected, and that the value is also "tomcat_shared_memory_id".

Run DSpace's bundled Solr server from NetBeans

As noted in the examples above, for most modules you should be running the web application via the main source code directory (e.g. `[src]/dspace-xml/` for XMLUI).

However, for the bundled Solr module to function properly, you need to run the "DSpace SOLR :: Local Customizations" project (i.e. [src]/dspace/modules/solr/) INSTEAD. You'll still need to customize its "Actions" (like detailed above) to set the property `dspace.dir=[installation-directory].`

Integrate DSpace Javadoc within Netbeans

This section provides instructions for generating the DSpace javadoc and its integration within Netbeans. Its aim is to allow developers to refer to the current DSpace Library API calls, and understanding its uses from within the IDE.

- Right click on the "DSpace Parent Project" (root project) - If it isn't opened, you may need to open it first.
- Select "Generate Javadoc"
- Javadoc for DSpace should be available within Netbeans. (It will appear during auto-complete functionality – if you hover over a method, the Javadocs will be displayed if any exists)