Installer Prototype

"Easy" Installer for DSpace

Primary Goals of this work

> **Prototype Only**
>
> This work is an initial prototype. It should not be considered stable until formally released as a part of DSpace

Main Goals:

- To create an easier installation/upgrade process, which does not require familiarity with Maven or Ant.
  - NOTE: However, it should still be possible for developers to build DSpace from source code using current Maven and/or Ant tools
- Installer should not require recompiling code, or pulling down anything via Maven. Installer should be able to be complete successfully with no internet connection whatsoever (i.e. the installer will need to have all third party dependencies & DSpace code within it)
- Installer should guide users by asking questions which will fill out the basic settings in dspace.cfg file.
  - Basic settings include the installation location and all database configuration settings

Potential features requiring further investigation:

- Can the installer also be used to perform an "upgrade" of DSpace?
  - Obviously, it would **not** be able to automatically update their customizations. But, could potentially help users to more easily upgrade to the latest 'out-of-the-box' version of DSpace (and copy their customizations to a backup location, where the user can reapply them as needed once upgrade is complete).
  - Can the installer help users upgrade their DSpace database? (This would require the installer to know what version user is currently running, and then call the necessary SQL upgrade scripts in proper order.)

Initial Usage Details

Where is Code Available At?

The code is available from SVN prototype branch at: http://scm.dspace.org/svn/repo/sandbox/installer-prototype/

This code **only** includes a customized version of the normal 'dspace' Assembly module from Trunk (1.8.0-SNAPSHOT).

How do I build the installer?

After downloading the code, run the following from `[installer-prototype]/dspace/` directory:

```
mvn package -Pdspace-installer
```

> **May need to first build Trunk**
>
> Currently, to keep the 'installer-prototype' SVN copy very minimal, I have **not** copied over an entire version of Trunk. The 'installer-prototype' only includes the 'dspace' Assembly directory. So, if you run into Maven build errors, you may first need to build Trunk, in order to ensure your local Maven repository (`~/.m2/`) has a copy of all 1.8.0-SNAPSHOT dependencies.

How do I run the installer?

After the build completes, you'll see a JAR installer created at `[installer-prototype]/dspace/target/dspace-installer-1.8.0-SNAPSHOT.jar`

You can execute this Installer by running the following from the `[installer-prototype]/dspace/target/` directory:

```
java -jar dspace-installer-1.8.0-SNAPSHOT.jar
```

After you run the installer, it will ask a series of questions around where you wish to install a copy of DSpace.

> **Currently only works for DSpace Install process**
>
> Currently, the Installer only works fully for a clean (fresh) install of DSpace. It is not fully functional in terms of updating or upgrading DSpace (some parts may work, but some may still be buggy).

Initial Implementation Details

How does the Installer work? How is it built/implemented?

- The initial Installer is packaged using One-Jar. One-Jar essentially provides us with an easy way to create an executable JAR file.
- When a user runs the JAR file, the One-JAR 'Boot' class is automatically called
- The One-JAR 'Boot' class automatically calls whatever is located at `/main/main.jar` within the installer.jar file. In our case, `/main/main.jar` calls the new dspace-install-api.jar file (see `[installer-prototype]/dspace/src/assemble/installer-assembly.xml` for details)
- The dspace-install-api.jar is what actually performs the installation
  - This JAR actually embeds Apache Ant within it, and also contains a custom Ant script (`installer-build.xml`) which creates the DSpace installation directory similar to how it is created in past versions of DSpace. More notes on this below.
- NOTE: The 'dspace-installer.jar' actually includes a full copy of all third party dependencies (JARs) as well as a copy of the DSpace install directory. See the `[installer-prototype]/dspace/src/assemble/installer-assembly.xml` for details.

Overview of dspace-install-api

The heart of this Installer is the new `/dspace/dspace-install-api/` module. This module currently only includes a few new files:

- `org.dspace.install.Installer` - This is the main executable Installer class. Currently, it essentially just uses the Apache Ant API to call a custom Installer 'installer-build.xml' file (which is based off the default DSpace `[installer-prototype]/dspace/src/main/config/build.xml` file). NOTE: Even though this installer uses the Ant API, Ant is **not** required to be installed on the local system. The Ant API is included within the Installer itself.
- `/src/main/resources/installer-build.xml` - This is the Ant Build file which actually tells Ant what it needs to do to actually perform the Install process.

Why embedding Maven into Installer won't work

This section is just a note on implementation details that have unfortunately ended in failure.

Initially, I thought: "If I can embed Ant in the installer to actually create the [dspace] installation directory, why not go one step further and embed Maven, so that the Installer.jar just auto-builds DSpace for you via Maven". The main reason for potentially embedding Maven was to allow for a smaller Installer.jar overall (less duplication of JAR dependencies, for each of the various DSpace WARs), and to allow Maven to do what it does best (namely managing dependencies).

Unfortunately, that is not as simple as it may sound. To properly embed Maven into the Installer.jar, you'd need to do the following:

1. Embed a complete copy of an offline Maven Repository into the Installer.jar (this would ensure Maven didn't need to go and download all dependencies for you – which obviously slows down the installation process, and wouldn't be much of an improvement over the current DSpace build process)
2. Embed a copy of Maven into Installer.jar
3. Ensure that when Installer.jar is run, it kicks off embedded Maven, points it at the embedded offline Maven Repository and then builds & installs DSpace.

Although #2 and #3 above seem to be possible, there seems to be no easy way to do #1 (embedding a copy of an offline maven repo). Unfortunately, Maven does not come with a plugin which can successfully create an entire offline repository, and ensure all dependencies are written there. A few more notes on this:

- There is a 'dependency plugin' which comes with a 'dependency:go-offline' option.
- Unfortunately, this 'go-offline' option fails to download all **maven plugins** which are dependencies of the build process. (Or at least, from my basic testing it seems to have difficulty with this in multi-module maven projects, like DSpace)
- In addition, this 'go-offline' mode cannot be pointed at a custom repository location (so you cannot specify the location of the offline repository). Instead, it will always use your local Maven repository defined in your 'settings.xml' file.
- The only way to take Maven **completely offline** seems to be what is detailed in this blog post. Unfortunately, it's not an easy process to script, so it's not really plausible to use for the building of our Installer. It's possible this may be easier in the future, but it's just not good enough yet.
  - http://stubbisms.wordpress.com/2008/08/28/maven-is-to-ant-as-a-nail-gun-is-to-hammer-and-nails-you-need-to-move-on/
- This thread details more frustrations around trying to use the 'go-offline' option: http://maven.40175.n5.nabble.com/Creating-repository-for-offline-building-with-dependency-go-offline-fails-td125614.html

The Conclusion: At least at this point in time, embedding Maven into an Installer is not really a plausible solution. We'll need to find a better way of avoiding an ever growing Installer.jar file (which is already rather large as some basic dependencies, e.g. dspace-api.jar, are duplicated 7 times in that one Installer.jar, once for each of the six webapps and once in [dspace]/lib).