SimpleAddonMechanism

WARNING: This page is still under development

The addon-mechanism for DSpace is a compile time modularity mechanism that allows code level separation between DSpace components. The main goal is to be able to extend DSpace without requiring patches and re-patches that lock developers into a single version of DSpace with no path to upgrade. This addon-mechanism moves DSpace closer to a binary distribution. These changes made by this proposal will make all compiled distributions (war files) non unique, so in the future we could just post binaries for DSpace user's to download and deploy. However the installation and upgrade processes will need to be refactored for this to be fully realized, this moves us a bit closer to that goal.

Contents

Preparation

To prepare for the AddonMechanism, the dspace trunk needs to be reorganized to separate the codebase into separate projects. There will be 4 initial separate projects:

- dspace: Configuration, Installation, and Release Managment support to create a dspace isntallation.
- dspace-api: The DSpace "kernel" codebase (basically everything but oai and webui).
- dspace-jspui: The DSpace jsp based webapplication.
- dspace-oai: The DSpace oai gateway webapplication.

The process that was completed on the trunk and is outlined here

SimpleAddonMechanism CodeReorganization

Maven based Addon Mechanism

Maven has good build and distribution management capabilities and provides an ideal build tool for managing DSpace modules/addons and doing release management for the entire project. We can achieve a high degree of build autonomy for each addon as well as control assembly of addons into "distributions" for both DSpace endorsed releases and as a tool to manage and maintain site specific customization our users may require.

What is an Addon?

All DSpace Addons are Maven projects. They are configured via a <your-project>/pom.xml and should be capable of being compiled independently of the full build process.

How does one "Compile" and "package" a distribution of DSpace?

The simplest set of instructions to date (assumes understanding of SVN checkout/export):

1.) Install a copy of Maven 2.x from your OS vendor or download it here and install its "bin" directory on your OS's PATH

2.) Checkout the current SVN trunk

```
~% svn checkout https://dspace.svn.sourceforge.net/svnroot/dspace/trunk
```

3.) Change Directory to trunk/dspace

```
~% cd trunk/dspace
```

4.) Execute the build process using Maven, default will assemble the API, OAI, and JSPUI Addons into a "target" dir within the dspace dir.

```
~% mvn package assembly:assembly
```

5.) Navigate to target/dspace-1.5-SNAPSHOT.dir

```
~% cd target/dspace-1.5-SNAPSHOT.dir
```

And you should see the following directory structure:

```
dspace-1.5-SNAPSHOT.dir
  |-- bin
  |-- config
  |-- docs
  |-- etc
  |-- lib
  |-- webapps
  |     |-- dspace-jspui.war
  |     `-- dspace-oai.war
  |
  |-- CHANGES
  |-- KNOWN_BUGS
  |-- LICENSE
  |-- README
  `-- build.xml
```

6.) Currently, the configuration process is still Ant driven for the moment. Installation and Upgrade can be done via the original targets.

```
~% ant [-Dconfig=-path-to-your-dspace.cfg] [fresh_install]
```

How does one **decide which addons to use**?

Addons can be included into the main dspace/pom.xml as modules, and/or configured in the pom via "profiles" which can be activated via command-line or configuration properies. Addons are included into the build process either via being added as a dependency to the "appropriate" dspace webapplication project, or by inclusion as a dependency into the main dspace/pom.xml project descriptor.

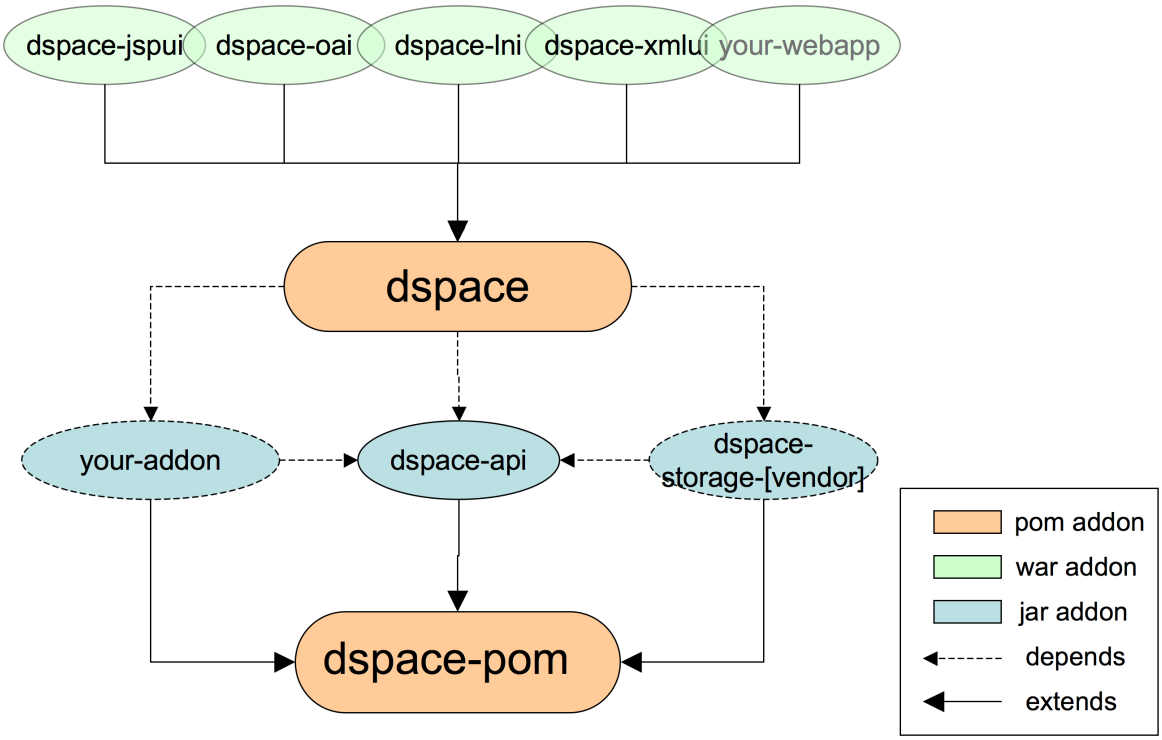
'Detail maven command-line build options.' --Mark Diggory 15:08, 30 May 2007 (EDT)

How does one make an Addon?

Addons fall into three primary "types" which have been outlined below.

'Detail creating a DSpace Addon further down in this document' --Mark Diggory 15:08, 30 May 2007 (EDT)

Maven based Addon dependecy scenarios



Simple Addon (A)

Addon with dependency on dspace-api, picked up as dependency by other dspace webapplications.

1.) Create a maven project with a dependency on dspace-api. For example your-addon/pom.xml might contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>org.your-org</groupId>
   <artifactId>your-addon</artifactId>
   <packaging>jar</packaging>
   <name>Your Addon</name>
   <version>1.0</version>
   <description>Your Addon To DSpace</description>
   <url>http://www.dspace.org</url>

   <dependencies>
      <dependency>
         <groupId>org.dspace</groupId>
         <artifactId>dspace-api</artifactId>
         <version>1.5-SNAPSHOT</version>
      </dependency>
   </dependencies>

</project>
```

2.) Add dependency on the addon to the webapplication projects you wish to have it included into:

```
<?xml version="1.0" encoding="UTF-8"?>
<project ...>
   <groupId>org.dspace</groupId>
   <artifactId>dspace-jspui</artifactId>
   <packaging>war</packaging>
   ...

   <dependencies>
      ...
      <dependency>
         <groupId>org.dspace</groupId>
         <artifactId>dspace-api</artifactId>
         <version>1.5-SNAPSHOT</version>
      </dependency>
  ~~> <dependency>
         <groupId>org.your-org</groupId>
         <artifactId>your-addon</artifactId>
         <version>1.0</version>
  ~~> </dependency>
   </dependencies>

</project>
```

3.) If you wish to have it included into the dspace/lib directory include it as a module in dspace/pom.xml

```
 <modules>
     <module>../dspace-api</module>
     <module>../dspace-oai</module>
     <module>../dspace-jspui</module>
     <module>../dspace-xmlui</module>
     <module>../dspace-lni</module>
 ~~> <module>../your-addon</module> <~~
 </modules>
```

Standalone Webapplication (B)

Webapplication with dependency on other dspace addons and dspace-api. Add your project to the list of Modules in the dspace/pom.xml

```
<modules>
    <module>../dspace-api</module>
    <module>../dspace-oai</module>
    <module>../dspace-jspui</module>
    <module>../dspace-xmlui</module>
    <module>../dspace-lni</module>
~~> <module>../your-webapp</module> <~~
</modules>
```

Customized Webapplication (C)

Webapplication which depends on dspace webapp and extends and overrides its presentation and behavior.

1. Add your customized webapp and remove the existing one.

```
<modules>
<module>../dspace-api</module>
<module>../dspace-oai</module>
<module>../dspace-jspui</module>
<s><module>../dspace-xmlui</module></s>
<module>../dspace-lni</module>
~> <module>../your-xmlui-webapp</module> <~
</modules>
```

2. Create your webapp project with a dependency on existing dspace-xmlui webapp.

```
<?xml version="1.0" encoding="UTF-8"?>
<project
   xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/maven-v4_0_0.xsd">

   <modelVersion>4.0.0</modelVersion>
   <groupId>org.your-org</groupId>
   <artifactId>your-xmlui-webapp</artifactId>
   <packaging>war</packaging>
   <version>1.0.0</version>
   <description>Your Customized Manakin Webapplication</description>

   <dependencies>
      <dependency>
        <groupId>org.dspace</groupId>
        <artifactId>dspace-xmlui</artifactId>
        <version>1.5-SNAPSHOT</version>
        <type>war</type>
      </dependency>
   </dependencies>

</project>
```

Dependency Management

Because Maven provides a high degree of dependency management, we are able to do the following at development time, release creation time or at "site customization" time.

- Resolve any and all necessary dependencies from maven repositories (jars and wars) and make them available locally for compilation and packaging.
- Publish binary versions of our addons into ours http://maven.dspace.org/ and others maven repositories to make them available for the previous process.

Project Inheritance

- Provides a simple mechanism to bring common configuration settings under control of one pom.xml managed in the "dspace" assembly project.

Project Assembly / Modules

- Provides an excellent and highly configurable mechanism for aggregating built addon artifacts under one assembled distribution directory that can then either be packaged into a zip/tar.gz/tar.bz2 archive the current structure of this distribution directory structure can be seen below:

```
 dspace
   |-- pom.xml  <-- Maven project file that defines what modules are to be added to the assembly.
   |-- src
   |    `-- assemble
   |        `-- assembly.xml  <-- descriptor of files/modules to include into distribution and any filtering
that is required.
   |
   `-- target
        `-- dspace-<version>
              |-- bin
              |-- config
              |-- lib
              |-- logs
              |-- logs
              |-- search
              `-- webapps
                    |-- dspace-jspui.war
                    |-- dspace-oai.war
                    `-- ...
```

- In the main dspace projects pom file (

  ```
  trunk/dspace/pom.xml
  ```

) one can outline the modules that are to be included into the projects build process.

```
  <project>
       <modelVersion>4.0.0</modelVersion>
       <groupId>org.dspace</groupId>
       <artifactId>dspace</artifactId>
       <name>DSpace Installation Project</name>
       <version>1.5-SNAPSHOT</version>
       <packaging>pom</packaging>
       ...
       <modules>
            <module>../dspace-oai</module>
            <module>../dspace-jspui</module>
            <module>../dspace-xmlui</module>
            <module>../dspace-lni</module>
       </modules>
       ...
```

Supporting DSpace Project Resources

If you're interested in how we can generate websites/documentation see the current published http__--projects.dspace.org maven sites for the above branch

External Resources

To learn about the basic usage of Maven and its build lifecycle, please consult the following documentation sources.

- Maven 2 Home Page
- Maven plugins
- Maven 2 codehaus tutorial
- Maven 2 tutorial on java.net
- Totally Free (no registration) Maven2 Book
- Free Maven2 Book

Things left to do

1. Update DSpace's documentation
2. make sure fresh installs / upgrades work.
3. Create simple example addons that are highly documented.
4. Write Eclipse instructions for developers

</html>