Release Notes 2.1.0

This release contains several bug fixes as well REST support, upgrades (JPA 2, CXF, JSF 2, Tapestry 5, Spring 3) and restructured archetypes that make everything **much** faster to run. In addition, there's a new **appfuse-ws** archetype that leverages Enunciate. Below is a highlighted list of significant enhancements since 2.0.2:

AppFuse

- [APF-1221] - Upgraded to JSF 2
- [APF-1114] - Upgraded to Tapestry 5
- [APF-1125] - Upgraded from XFire to CXF
- [APF-897] - Added appfuse-ws as an archetype
- [APF-1244] - Added support for RESTful Services
- [APF-1193] - Upgraded to Spring 3.0.5 and Spring Security 3.0.5
- [APF-1171] - Upgraded Compass to 2.2.0
- [APF-267] - Added search feature to list screen generation
- [APF-1212] - Changed all web frameworks to use Extensionless URLs
- [APF-1218] - Upgraded to JPA 2.0
- [APF-1195] - Upgraded to Hibernate 3.6.1
- [APF-1105] - Changed to use Spring annotations (@Repository, @Service and @Autowired) in service and data modules
- [APF-984] - Upgraded Spring MVC Controllers to use annotations
- [APF-1130] - Upgraded to Struts 2.1
- [APF-1131] - Removed Clickstream
- [APF-1164] - Removed OSCache
- [APF-1166] - Upgraded to Canoo WebTest 3.0
- [APF-970] - Fixed appfuse:full-source when spaces in project path
- [APF-1201] - Upgraded to Tomcat 6.0.29 for Memory Protection
- [APF-1012] - Extended appfuse maven plugin to allow sub-packages
- [APF-1119] and APF-1245 - Upgraded to JUnit 4 for all tests

AppFuse Light

- [EQX-68] - Created archetypes of AppFuse Light combinations
- [EQX-197] - Changed to use AppFuse Backend (a.k.a. remove OJB, Spring JDBC and JDO)
- [EQX-197] - Added Ajaxified Body to all Web Frameworks
- [EQX-200] - Moved from Ant to a Maven Modular project
- [EQX-209] - Upgraded Wicket to 1.4.15
- [EQX-213] - Upgraded to JSF 2
- [EQX-214] - Integrated Extensionless URLs
- [EQX-210] - Integrated AMP into AppFuse Light

In addition, a number of blog posts were written about features that went into this release while it was being developed:

- Adding Search to AppFuse with Compass
- Upgrading to JSF 2
- Fixing XSS in JSP 2
- Implementing Extensionless URLs with Tapestry, Spring MVC, Struts 2 and JSF
- AppFuse 2.1 Milestone 2 Released
- Using JRebel with IntelliJ IDEA on OS X
- AppFuse 2.1 Milestone 1 Released
- AppFuse Light converted to Maven modules, upgraded to Tapestry 5 and Stripes 1.5
- My Experience with Java REST Frameworks (specifically Jersey and CXF)
- Upgrading Hibernate to 3.4.0 and AppFuse for Tapestry 5

Demos for this release can be viewed at http://demo.appfuse.org. If you don't see a list of AppFuse and AppFuse Light demos, try clearing your browser's cache (shift + reload).

Please see the Upgrade Guide below or the QuickStart Guide to get started with this release. Individual issues fixed can be seen in the changelog , as well as the release notes for 2.1.0 M1 and 2.1.0 M2.

Upgrade Guide

There are many things that changed between AppFuse 2.0.2 and AppFuse 2.1. Most significant, archetypes now include all the source for the web modules so using jetty:run and your IDE will work much smoother now. The backend is still embedded in JARs, enabling you to choose with persistence framework (Hibernate, iBATIS or JPA) you'd like to use. If you want to modify the source for that, add the core classes to your project or run **appfuse:full-source**.

Diff is your friend
The easiest way to upgrade is likely to create a new project using 2.1.0, then compare the top-level directory of your project with the new one.

- Beyond Compare is a fabulous diff tool for Windows users.
- WinMerge is a great open source visual diff and merging tool for Windows.
- SmartSynchronize is a multi-platform file and directory compare tool.

The tutorial applications have been upgraded from 2.0.2 to 2.1.0.

The AppFuse Maven Plugin

This plugin currently does two things: 1) code generation for CRUD and 2) allows you to convert your project to use AppFuse's source instead of using its binary dependencies.

Generating CRUD with AMP

You can run the following command to generate CRUD screens/classes for a POJO:

```
mvn appfuse:gen -Dentity=Name
```

Web Tests Generation Bug
We used incorrect logic when deciding if WebTest support was included in your project. For this reason, you may see the following error when generating code:

```
[info] [AppFuse] Project doesn't use Canoo WebTest, disabling UI test
generation.
[info] [AppFuse] Support for jWebUnit will be added in a future release.
[info] [AppFuse] See http://issues.appfuse.org/browse/EQX-215 for more
information.
```

To fix this problem, perform one of the following steps:

- Create an empty file at target/appfuse/generated-sources/src/test/resources/{Class}-web-tests.xml.
- Upgrade the appfuse-maven-plugin in your project to use version 2.1.1-SNAPSHOT.

After doing this, you should be able to run **mvn appfuse:gen -Dentity={Class}** successfully.

If you don't specify the entity name, you will be prompted for it. Currently, if a @Column has "nullable = false", a "required field" validation rule will be generated as part of the web framework's validation configuration. This command will also install the generated code, unless you specify `-DdisableInstallation=true`.

If your entity is not defined in hibernate.cfg.xml, it will be added.

In a modular project, these commands must be run in the "core" and "web" modules. The plugin is smart enough to figure out when it should/should not generate stuff based on the packaging type (jar vs. war). If you want to generate specific code in a "war" project, you can use **gen-core** or **gen-web**.

Removing Code
You can run **mvn appfuse:remove** to remove the artifacts installed by **appfuse:gen**.

There's also a goal that allows you to generate model objects from database tables.

```
appfuse:gen-model
```

This goal will install the generated files into your source tree, unless you specify `-DdisableInstallation=true`. After running this command, you can use **appfuse:gen** to generate CRUD classes/tests/screens for this object.

If you want to customize the code generation templates (written in FreeMarker), you can copy them into your project using the following command:

```
appfuse:copy-templates
```

Installing AppFuse's source into your project

Creating a project with no dependencies on AppFuse is very easy. After you've create a new project, run the following command:

```
mvn appfuse:full-source
```

Issues with Maven 3
This command won't work with Maven 3. Please use Maven 2.2.1 to run this command (you only have to do it once per project).

This goal will convert your project to use all of AppFuse's source and remove all dependencies on AppFuse.

What the full-source plugin does:

1. Exports all sources from Subversion into your project. It reads the dao.framework and web.framework properties to determine what you need.
2. Calculates dependencies by reading pom.xml files form the various AppFuse modules. It replaces your dependencies with these new ones. The order of the dependencies added is alphabetical based on groupId.
3. Reads properties from the root AppFuse pom.xml and adds the ones that don't exist to your project. The order of the properties added is alphabetical.
4. Renames packages from `org.appfuse` to your project's groupId.

Detailed Changelog

Also see the release notes from 2.1.0 M1 and 2.1.0 M2 for previous improvements as part of this release.

AppFuse Issues (${entries.size()} issues)

T	Key	Summary	Status	Resolution

AppFuse Light Issues (${entries.size()} issues)

T	Key	Summary	Status	Resolution