createproject.groovy to maven archetype requirements discussion

notes

- would like to create a maven archetype from what we do with createproject.groovy
- idea is that Travis would do this as a contribution but would like requirements to speed things along
- Jerry and others have attempted to clean this up with and since Rice 2.0
    - Erik - Current createproject works on the command line but doesn't get imported into Eclipse with its current version (4)
    - Jerry - When you import the project it won't start up due to errors in the project being created.
    - Eric - if you don't use the docclass profile it seems to work even though you can't get rid of all the javascipt errors (calendar stuff) and couldn't get the validation to shut off.
    - Jerry - seems that the launch script was missing.
        - Eric - Eclipse has a WPET (?) tool but the directions are well out of date and should be updated.
    - Missing launch scripts, needs to work with latest eclipse, and has a few bugs
    - Jerry - seems that it makes you setup everything even if you just want to work on just KRAD
- Claus and Erik need it for the pre-conference workshop as well for KD2012
- What are the basic requirements? Phased approach and when to go with it?
- maven archetype is run from the command line, there are archetypes for everything you want to do basically
- it's a project template setup with options to specify thing
- if you're smart in how you version things, you can download and pull in necessary items to keep things consistent
- Jerry - but it couldn't be interactive?
    - seems like it could be done, but not sure
    - need to know if you can rerun it to modify the project
    - Jessica - there is an interactive mode option. need to find what it does
- Jeff - wouldn't have to start from scratch, could use one that's out there already and modify what's needed
- Eric - Put a maven wrapper around the RDO tool and such that we've created out there without having to check it out online
- Jerry - plug-ins would still just be maven wrappers (MOJOs) around the java code we already have.
- This will all be done on 2.1.x

1. Get createproject.groovy working so that it can be imported into newest version of Eclipse w/out error
    a. Eric W - will share his notes on how it works
    b. Erik - Wrap createproject.groovy in a maven command so that there's a seamless experience when we move from groovy to maven
2. Recreate createproject.groovy as maven archetype (Travis)

Travis' notes

- maven archetype plugin is designed for creating sample projects.
- it is the best way to create maven-based projects from a template
- you can use property replacement within files as well as on file & directory names
- it can be fully interactive or not
    - in interactive mode the tool will ask you to fill in each required property
- different project type should be separate archetypes rather than one large script like we have now
- works from the command line & from IDEs
- using the archetype plugin will give us IDE integration automatically (eclipse & intellij, not sure about netbeans)
    - ex: in intellij you can select File -> new project -> create project from scratch -> Maven module -> (and select a kuali rice archetype)

- an archetype should be easier to maintain because we will just be maintaining the templates, not the logic for moving files or directories
- i would start with a simple archetype (named quickstart) which would create an empty bundled rice app using maven overlays for the web content
- we could create other archetypes as well such as non-overlayed versions or thin client versions
- plan to create IT tests for the archetype by generating a project and then executing mvn clean install on the generated project to make sure it at least can compile
- ideally we would want to also test that the generated project can startup as well - could do this easier with embedded DB support
- i'll commit to having it done & tested well before Kuali Days
- we should not be providing launch scripts any longer as they are kind of a hack and pretty strange as well
    - instead we support 2 main ways of launching rice
        - the command line via various maven plugins configured in the pom
        - IDE appserver integration
            - intellij's works fine
            - eclipse's WTP works well (for KFS development). Previously is had trouble with war overlays but this looks like it might be fixed: http://stackoverflow.com/questions/8491308/how-to-handle-maven-war-overlays-in-eclipse

Documentation

The following is the documentation for Kuali Rice archetypes and the replacement of createproject.groovy.

Motivation

The createproject.groovy script is a groovy script that constructs a maven based project which uses Kuali Rice. Over time it has become difficult to maintain and even more difficult to test. In addition, it is a non-standard tool. This means that users must familiarize themselves with the tool which could be considered a barrier to entry.

Goal

Immediate

We want to have a tool that is standard, well documented, and easy to maintain and test. This tool should enable users to quickly create Kuali Rice based applications (either on the command line or through an IDE). The maven archetype plugin is the standard mechanism for generating maven-based projects. It is well documented on the maven archetype website. It is well supported by all major IDEs. We will initially support a simple Kuali Rice bundled archetype. We will not remove the createproject.groovy script but modify it to use the newly created archetype.

Long Term

We should remove the createproject.groovy script altogether. We may want to create other archetypes such as thin clients, archetypes using certain parts of rice, or more complex examples of KRAD, KIM, Workflow, or other modules. We may even want to wrap these archetypes in a large Kuali maven plugin ecosystem if desired.

Maven Archetype

From http://maven.apache.org/guides/introduction/introduction-to-archetypes.html

> *In short, Archetype is a Maven project templating toolkit. An archetype is defined as an original pattern or model from which all other things of the same kind are made. The names fits as we are trying to provide a system that provides a consistent means of generating Maven projects. Archetype will help authors create Maven project templates for users, and provides users with the means to generate parameterized versions of those project templates.*
>
> *Using archetypes provides a great way to enable developers quickly in a way consistent with best practices employed by your project or organization. Within the Maven project we use archetypes to try and get our users up and running as quickly as possible by providing a sample project that demonstrates many of the features of Maven while introducing new users to the best practices employed by Maven. In a matter of seconds a new user can have a working Maven project to use as a jumping board for investigating more of the features in Maven. We have also tried to make the Archetype mechanism additive and by that we mean allowing portions of a project to be captured in an archetype so that pieces or aspects of a project can be added to existing projects. A good example of this is the Maven site archetype. If, for example, you have used the quick start archetype to generate a working project you can then quickly create a site for that project by using the site archetype within that existing project. You can do anything like this with archetypes.*
>
> *You may want to standardize J2EE development within your organization so you may want to provide archetypes for EJBs, or WARs, or for your web services. Once these archetypes are created and deployed in your organization's repository they are available for use by all developers within your organization.*

Note that maven archetypes have changed over the course of time. Much of the documentation for creating archetypes is outdated. All archetypes in rice are using the new method of creating archetypes. All templates are found under the archetype-resources folder with a archetype-metadata.xml under the META-INF/maven directory.

Rice Archetype Quickstart Project

This is Kuali Rice's first archetype. It is located under the following directory inside of the rice project: rice/config/archetype/quickstart. In maven terms, the archetype directory is a new parent module holding all Kuali Rice archetypes. The only archetype module at this time is called quickstart.

The quickstart archetype has the following features:

1. demonstrates Spring Service creation
2. demonstrates unit testing
3. demonstrates integration testing
4. contains an internal BootStrap Spring file to load Kuali Rice
5. contains an internal BootStrap configuration file to configure Kuali Rice
    a. bundled configuration
6. uses war overlay to bring in web resources from the Kuali Rice project (portal, tag files, etc)
7. overrides portal tag files to customize the Rice Portal
8. paramterized to support different project configurations (ex: mysql, versus oracle)
9. fully supports the entire application lifecycle through maven. You can:
    a. fully install the application (mvn clean install). This cleans, compiles and runs unit tests.
    b. selectively run integration tests (mvn clean install -Dmaven.failsafe.skip=false). This cleans, compiles, runs unit & integration tests.
    c. start up the application using jetty (mvn jetty:run)
10. fully documented
11. fully integration tested. The tests do the following:
    a. generates a new project in a temporary directory
    b. compiles the new project running all unit and integration tests
    c. starts up jetty. Confirms that application launched correctly in Jetty
12. velocity based logic to support more advanced templating needs
13. tested in:
    a. maven 3.0.4 on the command line
    b. eclipse 3.7.2
    c. Intellij IDEA Ultimate 11.1

The quickstart archetype does NOT have the following features:

1. examples of specific rice features such as KRAD, Workflow.
    a. mainly because forcing custom database changes from what ships with rice is complex espcially since rice does not support in memory DBs

Usage

Here are several examples of different method of using the quickstart archetype. Note in order for the application to startup a database must be configured and running. See **Load Impex Data via Maven**. Currently if database parameters are not set it defaults to Oracle @ jdbc:oracle:thin:@localhost:1521:XE with username and password RICE/RICE

Maven CLI

1. Interactive mode

```
mvn archetype:generate -DarchetypeGroupId=org.kuali.rice
-DarchetypeArtifactId=rice-archetype-quickstart
-DarchetypeVersion=2.2.0-M4-SNAPSHOT
```

2. Automated

```
mvn archetype:generate -DarchetypeGroupId=org.kuali.rice
-DarchetypeArtifactId=rice-archetype-quickstart
-DarchetypeVersion=2.2.0-M4-SNAPSHOT
-DgroupId=org.foo
-DartifactId=bar
-Dversion=1.0-SNAPSHOT
-Dpackage=org.foo.bar
-Ddatasource_ojb_platform=Oracle
-Ddatasource_url=jdbc:oracle:thin:@localhost:1521:XE
-Ddatasource_username=RICE
-Ddatasource_password=RICE
```

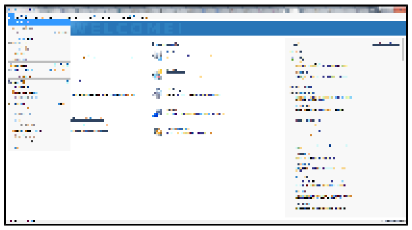
Not that the following parameters have suggested defaults and do not need to be specified: version, package, datasource_ojb_platform, datasource_url, datasource_username, datasource_password.

> *C:\project> cd bar*
> *C:\projects\bar> mvn clean install -Dmaven.failsafe.skip=false jetty:run*

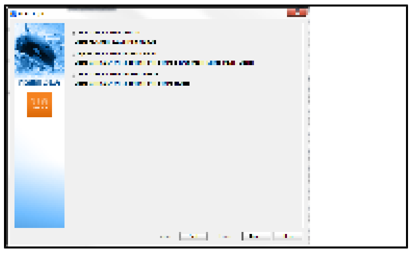
This will clean, compile, run unit and integration tests, and startup the project in jetty
Navigate to http://localhost:8080/bar

Intellij IDEA Ultimate 11.1

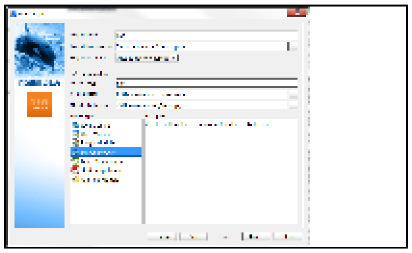
File -> New Project



Create project from scratch -> Next

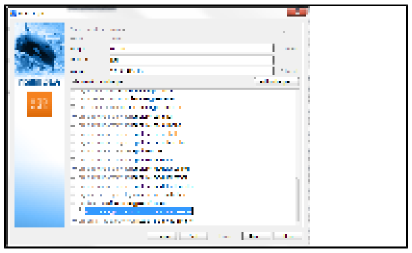


Project Name: bar
Module Name: bar
Select Maven Module -> Next

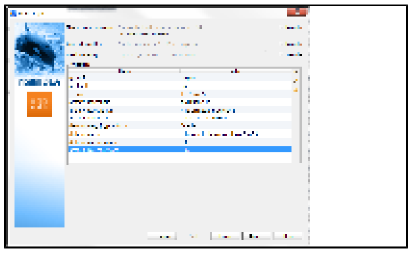


GroupId: org.foo
ArtifactId: bar
Version: 1.0-SNAPSHOT
Click create project from achetype
{optional} Add Archetype if quickstart does not exist in the list

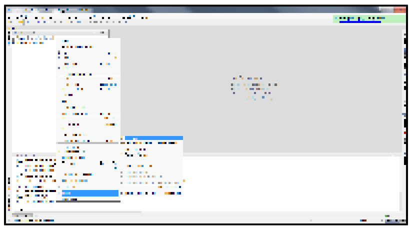
{optional} GroupId: org.kuali.rice
{optional} ArtifactId: rice-archetype-quickstart
{optional} Version: 2.2.0-M4-SNAPSHOT
select org.kuali.rice:rice-archetype-quickstart:2.2.0-M4-SNAPSHOT
Next



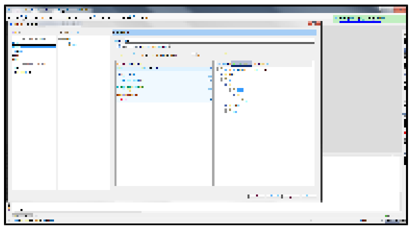
{optional} set the desired properties
Finish



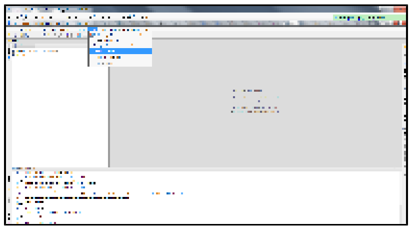
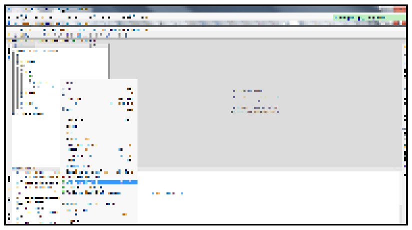
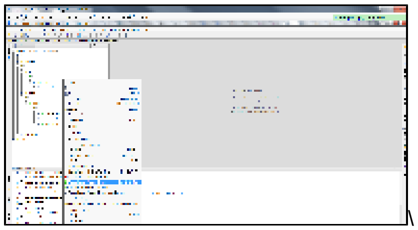
Force a "reimport" of the project
Right Click on Project -> Maven -> Reimport



Add the integration test folder as a source folder
Right Click on Project -> Open Module Settings
Navigate to src/it/java folder and click Test Sources -> Ok

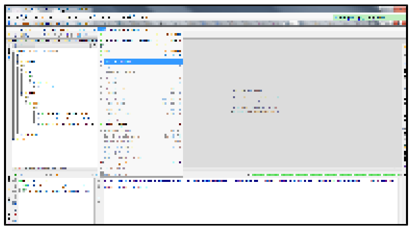


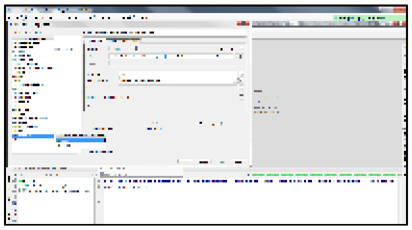
At this point it is a good idea to rebuild the entire project and make sure there are no errors. Then run all the unit and integration tests through Intellij.

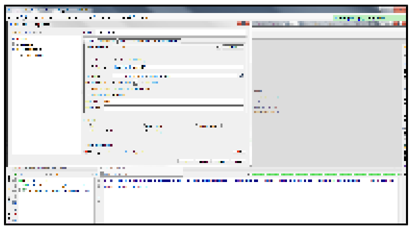




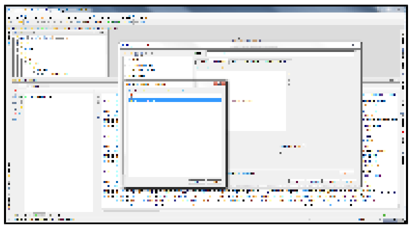
Now to launch in an Application Server through Intellij

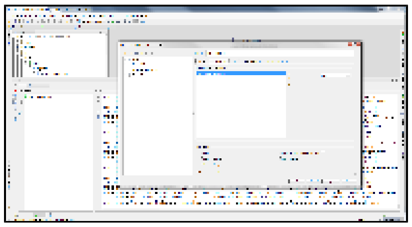
Run -> Edit Configurations

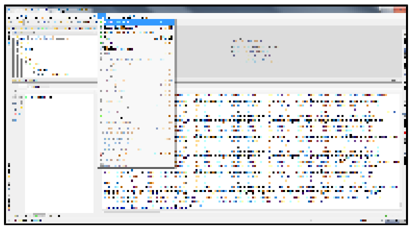

+ -> Tomcat Server (or Jetty if you prefer) -> local

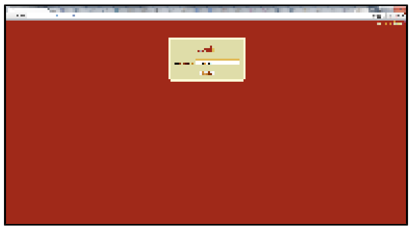
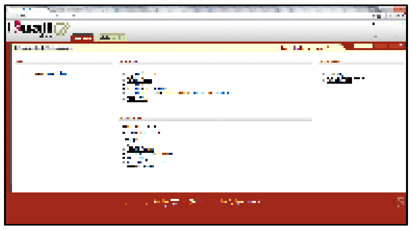

Name: Local Tomcat 6 (or 7)
Select Tomcat Server
VM Options: Add more memory (ex: -Xms512m -Xmx2g -XX:MaxPermSize=512m)


Click Deployment Tab -> + artifact -> bar:war (exploded) -> Ok


Application Context: /bar
Ok


Click Run -> Run Local Tomcat 6

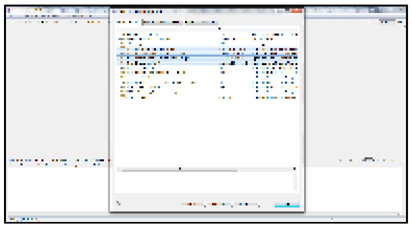

Watch the Application Start up!
Navigate to http://localhost:8080/bar

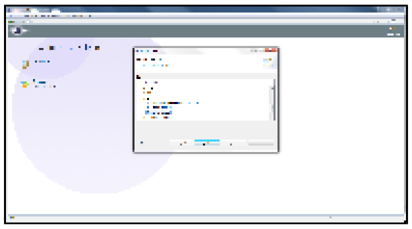
Eclipse 3.7.2 JEE Edition

Before you can start with eclipse the following plugins must be installed:

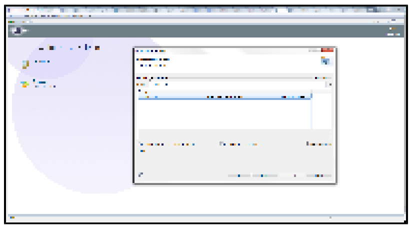
1. m2e - this is the core maven plugin for eclipse
2. m2e connector for buildhelper plugin - this allows m2e to recognize the buildhelper plugin.
    a. If you don't install this ahead of time it is ok. m2e is smart enough to show an error and will allow you to grab the plugin from the eclipse marketplace.
3. Maven Integration for WTP - this allows maven to integrate with WTP (Eclipse Web Tools Patform. It now support war overlays!!



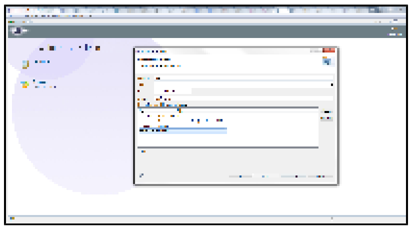
New -> Project -> Maven -> Maven Project -> Next -> Next



{optional} Add Archetype if quickstart does not exist in the list
{optional} Archetype Group Id: org.kuali.rice
{optional} Archetype Artifact Id: rice-archetype-quickstart
{optional} Archetype Version: 2.2.0-M4-SNAPSHOT
{optional} Ok
{optional} Include snapshot archetypes
select org.kuali.rice:rice-archetype-quickstart:2.2.0-M4-SNAPSHOT -> Next



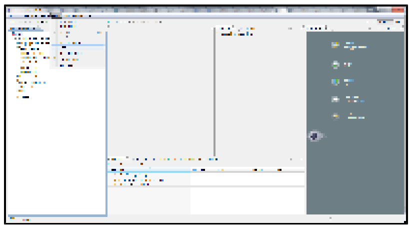
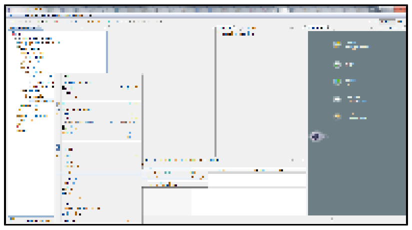
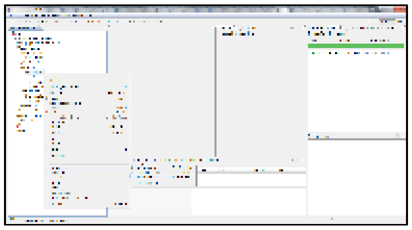
Group Id: org.foo
Artifact Id: bar
Version: 1.0-SNAPSHOT
package: org.foo.bar
{optional} set the desired properties
Finish



If the project's pom.xml contains and error related to the buildhelper plugin open the pom in eclipse. On the overview tab click on the error. This will find the required plugin in the eclipse marketplace to resolve this error. The plugin needed is the m2e connector for buildhelper plugin. After installing restart eclipse.

You may get JSP, JavaScript or other validation errors. Just ignore them. Eclipse's various validations creates many false positives. They can be disabled within eclipse.

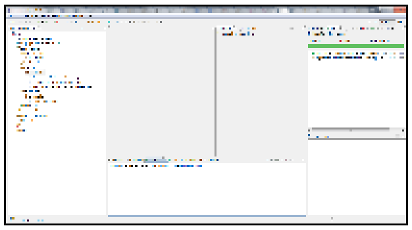
At this point it is a good idea to rebuild the entire project and make sure there are no errors. Then run all the unit and integration tests through eclipse.

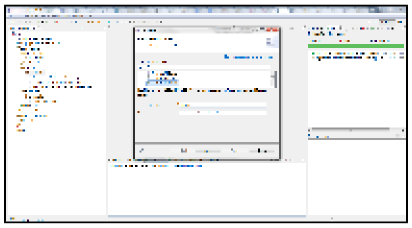
Now to launch in an Application Server through Eclipse WTP plugin

From the JEE perspective:
Click on the Servers tab.

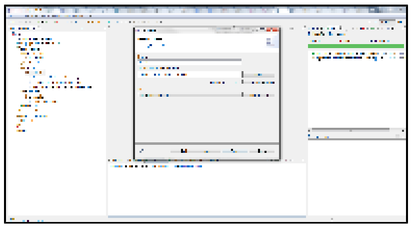
Click new server wizard



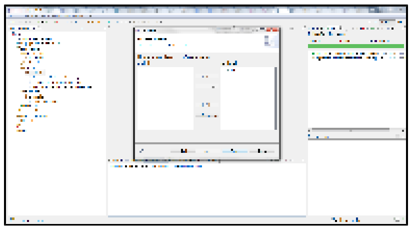
Select Apache Tomcat 6 (or the app server of your choice) -> Next



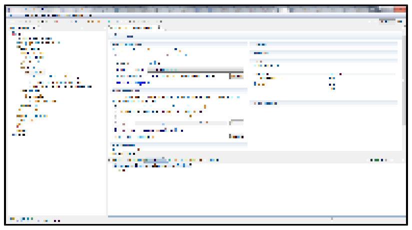
Point to a local install of tomcat -> Next



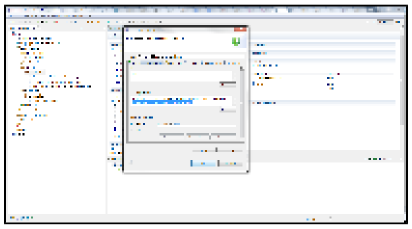
Select bar -> Add -> Finish

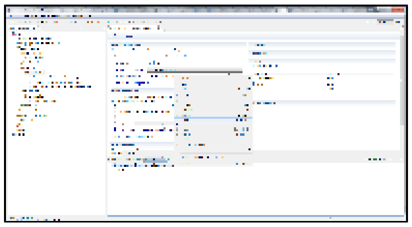


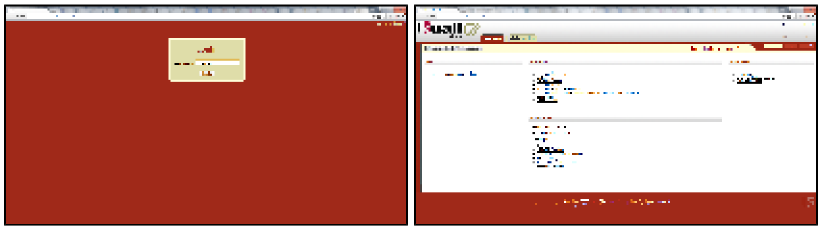
Double Click on the newly created Server under the Servers tab


Open Launch Configuration

Click Arguments tab
VM Arguments: Add more memory (ex: -Xms512m -Xmx2g -XX:MaxPermSize=512m) - Ok


Right Click on Server under Servers tab -> Start


Watch the Application Start up!
Navigate to http://localhost:8080/bar



Archetype Maintenance

There is an integration test in a new module in rice located at rice/it/config. This test is called QuickStartTest. This should be the first line of defense to keeping the quickstart archetype working. The integration test works by calling maven commands. As a result, maven must be available on the system running our integration tests.

The quickstart archetype project will likely need to be maintained if the following changes:

1. web content files that are overrriden in in the archetype: web.xml, portal files
2. rice maven structure
3. anything referenced in the BootStrapConfig.xml or BootStrapSpringBeans.xml
    a. did rice add any require config params? any required Spring Beans?
4. rice pom.xml configuration (dependencies, plugins, etc)

attendees

- Jeff
- Jerry
- Eric
- Erik
- Jessica
- Claus
- Matt

tasks

task	description	phase	reporter	assignment	JIRA	notes
1	allow creation of just modules needed via a walkthrough process or by having it create the basics first then pick the add ons after	2	jerry			First archetype does not do this right now. Suggest creating other archetypes for common configurations
2	give options to setup/tweak parameters after the initial setup	2	jerry			not supported with archetypes
3	should follow the process of the groovy as far as being a command line tool to start off with	1	eric	travis	KFs-333	basically works the same. this is done and documented
4	figure out what interactive mode in maven archetype works	1	jessica	jessica	KFS-333	done and documented
5	need to ensure archetype available for standalone & embedded and to include the sample app or not (maybe 4?), not sure if this means we have separate archetypes or if the process can be told what to do	1	jerry	travis	KFS-333	only supported bundled. Consider creating another archetype for this or you could support embedded with a few tweaks (need to have cli options to kick off standalone rice, need to use velocity to twaek BootStrap config)
6	ensure the project doesn't have to be checked out to run it	1	jerry	travis	KFS-333	just need to have the archetype in a maven repo
7	well documented directions on what it does and what you can specify to do this or that	1	jerry	travis	KFS-333	done
8	POM file has all the tooling in the project already so that the rest of the tools are there for the command line experience	2	eric	travis	KFS-333	of course I would support command line ;-)

9	fix createproject.groovy and wrap it in maven so it's seamless and works on Eclipse 4+	1	erik	erik & claus		do we really want to continue on with createproject.groovy even if it just delegates to the archetype?
10	ensure the name for the archetype is correct and common for the groovy and maven work	1	jeff	jeff	KFS-333	
11	update groovy documentation with what Eric has noted	1	matt	erik & claus		do we really want to continue on with createproject.groovy even if it just delegates to the archetype?