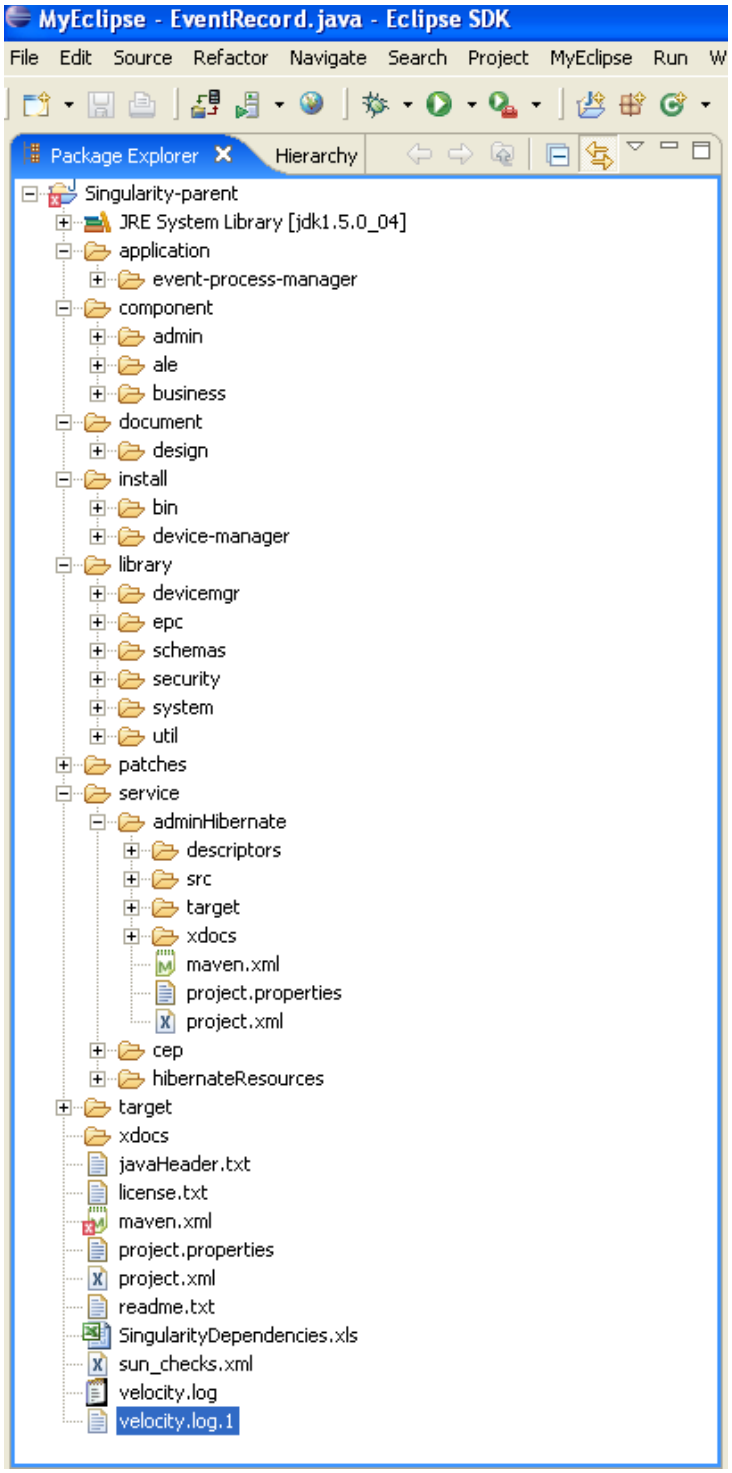
Below is a graphic to show the project layout:

Once you download the source you will have to download maven 1.1 beta2 as well http://maven.apache.org/start/download.html. Place MAVE_HOME./bin in your path(or use the full path to maven). Then type maven in the top level directory to build Singularity.



For the first time Singularity is built, some of the dependent libraries/plug-ins do not get downloaded properly. A work around for this issue is to start maven in the sub-project that failed to build; it should then download dependencies and build the sub-project correctly. Then the build can be restarted from top level. I have only had this happen on the first build in a clean environment.

The built components are in their respective folders “target” directory. The top level is a project, the next level are just categories to organize the artifacts, and the next level of directories are sub-projects. Sub-project has not direct meaning in Maven, just my term for describing the child projects under the top level project. Maven allows for a hierarchy of projects, the top-level being what it calls a multi-project. Maven docs explain this nicely.

The category folder such as “application” will contain all the applications such as EARs, or Rich Client Applications. Event-process-manager is a sub-project, and an EAR is built (Event-process-manager is the middleware component that contains the ALE service).

In maven there are three significant things you should be aware of for each project, or sub-project. Project.xml (names the project, and its dependencies, and where to find the src, and JUNIT test src), maven.xml (explains how to build the project and what goalsshould be run (Maven.xml looks a lot like Ant, goals are similar to Ant targets), project.properties (properties for the various maven plugins being used in the build).

Singularity has a few categories:

Library

Utility

System

Device Manager

Schema

Contains all compoents such as Utility (Util), system, Device Manager (devicemgr), schema, etc.

Schema - ALE schema definition and the build generates java classes that represent the schema, those classes can then convert themselves to DOM or XML Strings, as well as parse DOM or XML strings into java objects. I have not fully implanted this yet. Only in reports, and client ALE usage. I need to convert ECSpec over the xmlbeans for the next release.

Component

any ejb, webapp is considered a component and are built here. The components each are a project, so you can pick and choose what you want to assemble.

Service

Any MBean service is built here. Two services are hibernate services, one is a HAR, and one is a SAR (trying them both out), the third is the rules service for complex event processing.

Install

Hold install staging area. Device-manager is a Jini Install image for the device managers, and expect to have a JBoss, or whatever other install images should be created for easy deployment. Of course you can just deploy the component if you already have these environments up, but if you don't we will provide an install image for quick startup. We are still working on the install piece of the build but the vision is we can install artifacts into whatever environment you have remotely or locally, and also install them into the install images so you have a contained environment. Expect the contained environment mostly to be used by developers. The bin directory is jus the startup scripts for JBoss so it starts with a security manager turned on.

The current approach is that you can use what is already built for applications, or create a new application, and you assemble just the artifacts you need, leveraging off as much or as little of Singularity as you need. Also building your project within this environment provides a nice, structured environment to maintain your project.