Source Code Organization and Build System

Overview

What are the most important facts that a developer should know about this source code organization and build system?

> Briefy highlight the most important things that another developer should know about your source code organization and build system. Focus on unconventional or difficult to understand concepts, rules that have been broken in the past, or things that have changed.

What are the ranked goals of this source code organization and build system?

- Separation of files by type
- Platform independence

Key Directories and Files in Developer Working Copies

Path	VC	Description
web/css/	Yes	Css file , if any
web/images/	Yes	Image files, if any
www/	Yes	Project documents (e.g., overview, plan, requirements, and design)

Build Targets

Target	Description
init	Creates the directories which are used by the other build targets. A direct compliment to the clean target, the directories created in this target will be removed through the clean target.
compile = default	Compiles python source code and django under the "build" directory

Build Configuration Options

Property	**Description**
app.name	The name of this application. This should be one short word. Used in the name of resulting package files
app.version	Version number of this release. Used in the name of resulting package files.

Source Code Organization and Build System Checklist

- Separation of files by type: Are files separated by type?
  Yes. Except that application JSP and HTML files are in the same directory, which is convenient because sometimes we change an HTML file to be a JSP file.
- Separation of version-controlled and non-version controlled files: To what extent has this been achieved?
  It has been achieved. Everything is under version control except for the "build" directory. No step in the build process should create or modify any file in any other directory.
- Compatibility with standard build processes: To what extent has this been achieved ?
  So far, so good. We can use build.xml files that are very close to the examples that come with Ant. One difference is that we keep our technical documentation under "www" rather than under "docs". Also, we have avoided the use of custom ant tasks.
- Platform independence: To what extent has this been achieved ?
  We are using Ant, which is itself platform independent. The names of the files and directories should work across platforms because they do not rely on case-sensitive names. We assume that the utility scripts in the "scripts" directory support all needed platforms and we have not created directories for different versions of these files aimed at specific platforms.
- Have these implementation decisions been communicated to the development team and other stakeholders ?
  Yes, everyone understands. Feedback is welcome.