ContentBuilder

Builds a ContentPackage archive

Table of contents

1. Overview

ContentBuilder: *Builds a ContentPackage archive*

A ContentBuilder is responsible for building ContentPackage packages. Normally, users of this type will run two commands: scmChanges to check if the files in the local checkout tree are different from those on the repository, and Build to package those changes and store the resulting package archive in the repository.

2. Design

Super Type
Builder

Role	**Concrete**. (Objects can be created.)
Instance Names	**Unique**
Notification	**false**
Template Directory	`${modules.dir}/ContentBuilder/templates`
Data View	Children, proximity: **1**
Logger Name	ContentBuilder

3. Constraints

3.1. Allowed Child Dependencies

- BuilderBuildFile1
- BuilderBuildTarget1
- BuilderImportMax1
- BuilderImportMin1
- BuilderPackageExtension1
- BuilderPackageFilebase1
- BuilderPackageInstallroot1
- BuilderPackageType1
- BuilderPackageVersion1
- BuilderScmBinding1
- BuilderScmConnection1
- BuilderScmLabel1
- BuilderScmModule1
- ContentBuilderScmChangesOutput 1

1: These types have a *Singleton* constraint. Only one instance may be added as a resource.

3.2. Allowed Parent Dependencies

- ContentUpdater
- Node

4. Attributes

4.1. Exported Attributes

Name	Property
`basedir`	deployment-basedir
`targetdir`	deployment-install-root

4.2. Defaults for Imported Attributes

Name	Default	Description
`buildFile`	${modules.dir}/ContentBuilder/lib	The purpose of the ContentBuilder build.xml is to generate a new ContentPackage tgz archive.
`buildTarget`	package	target to call from runBuildscript command
`importMax`	1	maximum number of expected ContentPackage files
`importMin`	1	minimum number of expected ContentPackage files
`packageExtension`	tgz	
`packageFilebase`	.*?	
`packageInstallroot`		directory where package archive should be extracted
`packageType`	ContentPackage	Name of type to use to store and registrate new packages. Do not change unless subtyping ContentPackage.

packageVersion	${opts.buildstamp}	
scmBinding	svn	
scmChangesOutput	${entity.instance.dir}/var/scmCha	location of scmChanges output
scmConnection		connection string to SCM repository.
scmLabel		
scmModule		Name of module to checkout from SCM repository.

5. Commands

Note:

Commandline options displayed in square brackets "[]" are optional. If an option expects arguments, then angle brackets are shown after the option "<>". Any default value is shown within the brackets.

5.1. Build

Run the build cycle.

The purpose of the ContentBuilder build life cycle is to package any incremental changes between the SCM repository and the local workspace.

Usage

```
Build [-buildstamp <>]
```

5.1.1. Workflow

1. scmChanges
2. scmCheckout
3. generateManifest
4. generateTarIncludes
5. runBuildScript
6. repoImport

5.1.2. Success Handler

Email	
Subject	[${context.type}:${context.name} @ ${framework.node}] ${command.name} - SUCCESS

File	`${modules.dir}/Deployment/templates/notice.html`

5.1.3. Options

Option	Description
`buildstamp`	*build identifier*

5.2. generateManifest

Generate a ContentPackage manifest.

Creates a manfiest.xml file from the results of the scmChanges command

Usage

```
generateManifest [-buildstamp <>] [-manifest
<${entity.instance.dir}/var/manifest.xml>] [-module <>]
[-print] [-scmChangesOutput <>]
```

5.2.1. Options

Option	Description
`buildstamp`	*build identifier*
`manifest`	*output file*
`module`	*module name*
`print`	*print results to console*
`scmChangesOutput`	*file list*

5.3. generateTarIncludes

Generate a file for tarset includes.

The format of the tar includes file lists one file per line in plain text.

Usage

```
generateTarIncludes [-buildstamp <>] [-module <>] [-print]
[-scmChangesOutput <>] [-tarincludes
<${entity.instance.dir}/var/tarincludes.txt>]
```

5.3.1. Options

Option	Description

buildstamp	*build identifier*
module	*module name*
print	*print result to console*
scmChangesOutput	*file list*
tarincludes	*file to write tar includes*

5.4. runBuildScript

Build a new ContentPackage archive.

Usage

```
runBuildScript [-basedir <>] [-buildfile <>] -buildstamp <>
[-manifest <${entity.instance.dir}/var/manifest.xml>]
[-target <package>] [-targetdir <>] [-tarincludes
<${entity.instance.dir}/var/tarincludes.txt>]
```

Execution	bash
Arguments	${ant.home}/bin/ant -f ${opts.buildfile} ${opts.target} -Dopts.basedir=${opts.basedir}/${entity.attribute.scmModule} -Dopts.tarincludes=${opts.tarincludes} -Dopts.targetdir=${opts.targetdir} -Dopts.tarfilename=${context.name}-${opts.buildstamp}.tgz -Dopts.manifest=${opts.manifest};

5.4.1. Options

Option	Description
basedir	*directory where build resources reside*
buildfile	*build file to execute*
buildstamp	*build identifier*
manifest	*package manifest file*
target	*build target to evaluate*
targetdir	*directory build artifacts will be written*
tarincludes	*tar file includes*

5.5. scmChanges

Report the SCM status.

Usage

```
scmChanges [-basedir <>] [-binding <svn>] -buildstamp <>
[-connection <>] [-format <xml>] [-label <HEAD>] [-module
<>] [-quiet] [-scmChangesOutput <>]
```

5.5.1. Options

Option	Description
`basedir`	*base directory*
`binding`	*scm binding*
`buildstamp`	*build identifier*
`connection`	*connections string*
`format`	*output format. plain, markdown, xml, xmlproperty* • plain: lists one file per line w/o any adornment • markdown: Uses simple text formatting rules. See: Daring Fireball: Markdown for reference. • xml: Uses xml format intrinsic to SCM tool. • xmlproperty: Transformation of xml format to one suitable for loading by XmlProperty task.
`label`	*revision identifier*
`module`	*scm module*
`quiet`	*do not print output*
`scmChangesOutput`	*file list*

6. Related Types

The following types are defined for use with ContentBuilder.

6.1. ContentBuilderSetting

6.1.1. Overview

ContentBuilderSetting: *A ContentBuilder setting.*

6.1.2. Design

Super Type
Setting

Role	**Abstract**. (Objects cannot be created.)
Instance Names	**Unique**

6.1.3. Constraints

6.1.3.1. Allowed Parent Dependencies

- Builder

6.2. ContentBuilderScmChangesOutput

6.2.1. Overview

ContentBuilderScmChangesOutput: *The results of the scmChanges command is stored in this file.*

The results file should contain a list of changes between the local basedir and the SCM repository

6.2.2. Design

Super Type
ContentBuilderSetting

Role	**Concrete**. (Objects can be created.)
Instance Names	**Unique**

6.2.3. Attributes

6.2.3.1. Exported Attributes

Name	Property
`scmChangesOutput`	settingValue