generate-objects

Command Reference

Table of contents

1. Description

Generate an initial set of object data files from the type templates.

Reads type definitions and generates an initial set of object data files suitable to load in the server with <u>load-objects</u>.

Example: Bootstrap files containing deployment and setting object info

```
ctl -p project -t ProjectBuilder -o object -c generate-objects
```

Files will be written to directory: ${opts.basedir}/objects

2. Usage

```
ctl -t ProjectBuilder -o <objectname> -c generate-objects
[-basedir <>] [-basetype <all>] [-defaults <>] [-delimiter
<:>] [-format <projectxml>] [-name <default>] [-overwrite]
[-targetdir <>] [-templateDir <>] [-type <>] [-upload]
```

2.1. Options

Option	Description	Type	Default
basedir	*dir containing modules*	string	${entity.attribute.basedir}
basetype	*specifies the basetype of object data*Only relevant for the tabular format. **Note:** At this time, the tabulur format supports two base types: <u>Deployment</u> and Setting. Each type element in the type.xml must specify an *order* attribute (e.g, <type order="Deployment" ...) for the generate-objects command to understand how to format the tabular layout.	string	all

`defaults`	*file containing defaults data*	string	`/ ${entity.attribute.defaults}`
`delimiter`	*field delimiter. Only relevant if format is tabular. Defaults to ":" (colon)*	string	`:`
`format`	*data format. tabular or projectxml.*	string	`projectxml`
`name`	*name to give objects*	string	`default`
`overwrite`	*overwrite existing files*	boolean	
`targetdir`	*dir containing build*	string	`${entity.attribute.targetdir}`
`templateDir`	*file containing defaults data*	string	`/ ${entity.attribute.templateDir`
`type`	*type name* Only relevant for the tabular format. If type not specified, the object data for all types in basedir will be generated	string	
`upload`	*load objects after they are generated*	boolean	