NAME
>   mdtool – MonoDevelop Application Runner

SYNOPSIS
>   **mdtool** <applicationId> ...
>
>   **mdtool** setup ...
>
>   **mdtool** -q

DESCRIPTION
>   **mdtool** is a tool that can execute head-less MonoDevelop applications, this can be used for example to do batch compilations without starting up the MonoDevelop GUI.

USAGE
>   **applicationId ...**
>>   Runs the specified application.
>
>   **-q**   Lists available applications.

Available Applications
>   The list of applications returned by the option -q might be different depending on which add-ins are installed.
>
>   **gsetup**   Graphical add-in setup utility.
>
>   **build [options] [project-file]**
>>   Project build tool. Use this tool to build any solution supported by MonoDevelop (eg. .mds, .sln). If no solution/project is specified then it will pull the first solution it can find in the current directory. You can use this option to build packages in a head-less or batch fashion.
>>
>>   **Options**
>>>   **-c , --configuration:CONFIGURATION** Name of the solution configuration to build.
>>>
>>>   **-t , --target:TARGET** Name of the target: Clean or Build
>>>
>>>   **-p , --project:PROJECT** Name of the project to build.
>
>   **dbgen**   Parser database generation tool.
>
>   **project-export**
>>   Project conversion tool.
>
>   **IDE**   The MonoDevelop IDE.
>
>   **generate-makefiles <solution-file> [--simple-makefiles] [-d:default-config]**
>>   Makefile generator tool. This can be used to generate makefiles from any solution supported by MonoDevelop.
>>
>>   **Options**
>>>   **--simple-makefiles , -s** Generates set of Makefiles with the most common targets, and a configuration script that does a basic check of package dependencies. Default is to generate Makefile structure based on Autotools with the standard targets and configuration scripts.
>>>
>>>   **-d:default-config** Configuration that the Makefile will build by default. Other configurations can be selected via the '--config' or '--enable-*' option of the generated configure script.

setup command [arg1, .., argn2]
> Runs the MonoDevelop add-in setup command. You must specify a command to execute. If none
> is provided a list of commands is displayed

Add-in Commands

> **install (i) [package-name|package-file]** Installs an add-in or set of addins. The command
> argument is a list of files and/or package names. If a package name is provided the pack-
> age will be looked out in the registered repositories. A specific add-in version can be
> specified by appending it to. the package name using MonoDevelop.SourceEditor/0.9.1

> **uninstall (u) [package-name]**
> > Uninstalls an add-in. The command argument is the name of the add-in to unin-
> > stall.

> **check-install (ci)**
> > Checks if a package is installed. If it is not, it looks for the package in the regis-
> > tered repositories, and if found the package is downloaded and installed, includ-
> > ing all needed dependencies.

> **update (up)**
> > Downloads and installs available updates for installed add-ins.

> **list (l)**   Prints a list of all installed add-ins.

> **list-av (la)**
> > Prints a list of add-ins available to install in the registered repositories.

> **list-update (lu)**
> > Prints a list of available add-in updates in the registered repositories.

Repository Commands

> **rep-add (ra) url1 [url2 [url2]]**
> > Registers an add-in repository. Several URLs can be provided.

> **rep-remove (rr) url1 [url2 [url2]]**
> > Unregisters an add-in repository. Several URLs can be provided.

> **rep-update (ru)**
> > Updates the lists of addins available in all registered repositories.

> **rep-list (rl)**
> > Shows a list of all registered repositories.

Add-in Registry Commands:

> **reg-update (rgu)**
> > Looks for changes in add-in directories and updates the registry. New add-ins
> > will be added and deleted add-ins will be removed.

> **reg-build (rgu)**

> **info [filename|package-name]**
> > Prints information about an add-in.

Packaging Commands

> **rep-build (rb) path**
> > Scans the provided directory and generates a set of index files with entries for all
> > add-in packages found in the directory tree. The resulting file structure is an add-
> > in repository that can be published in a web site or a shared directory.

> **pack (p) file-path**
> > Creates an add-in package (.mpack file) which includes all files needed to deploy
> > an add-in. The command parameter is the path to the add-in's configuration file.

help (h) [command]
> Shows help for the given command

Debug Commands:

dump-file
> Prints the contents of a registry file for debugging.

SEE ALSO
monodevelop(1), mono(1), mcs(1)

COPYRIGHT
Copyright (C) 2007 Novell, Inc (http://www.novell.com)

MAILING LISTS
Visit http://lists.ximian.com/mailman/listinfo/monodevelop-list for details.

WEB SITE
Visit http://www.monodevelop.com for details