**TASK IN GRUNT**

Tasks are grunt's bread and butter. Every time Grunt is run, you specify one or more tasks to run, which tells Grunt what you'd like it to do.

If you don't specify a task, but a task named "default" has been defined, that task will run by default.

**THERE ARE TWO TYPES OF TASK:**

**1.BASIC TASK**

When a basic task is run, Grunt doesn't look at the configuration or environment—it just runs the specified task function, passing any specified colon-separated arguments in as function arguments.

grunt.registerTask(taskName, [description, ] taskFunction)

grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);

**2.Multiple TASK**

When a multi task is run, Grunt looks for a property of the same name in the Grunt configuration. Multi-tasks can have multiple configurations, defined using arbitrarily named "targets."

grunt.registerMultiTask(taskName, [description, ] taskFunction)

,Asynchronous task:

For simplicity's ,Grunt uses a synchronous coding style, which can be switched to asynchronous by calling this.async() within the task body.

Note that passing false to the done() function tells Grunt that the task has failed.

For example:

grunt.registerTask('asyncme', 'My asynchronous task.', function() {

var done = this.async();

doSomethingAsync(done);

});

Custom task

If u don’t want to use the multiple task then used

grunt.registerTask('default', 'My "default" task description.', function() {

grunt.log.writeln('Currently running the "default" task.');

});

Inside a task, you can run other tasks.

grunt.registerTask('foo', 'My "foo" task.', function() {

// Enqueue "bar" and "baz" tasks, to run after "foo" finishes, in-order.

grunt.task.run('bar', 'baz');

// Or:

grunt.task.run(['bar', 'baz']);

});

Tasks can access their own name and arguments.

grunt.registerTask('foo', 'My "foo" task.', function(a, b) {

grunt.log.writeln(this.name, a, b);

});