Creating Custom Registries

Allows custom registries to be plugged into the task system, which can provide shared tasks or augmented functionality. Registries are registered using registry().

Structure

In order to be accepted by gulp, custom registries must follow a specific format.

```
// as a function
function TestRegistry() {}
TestRegistry.prototype.init = function (gulpInst) {}
TestRegistry.prototype.get = function (name) {}
TestRegistry.prototype.set = function (name, fn) {}
TestRegistry.prototype.tasks = function () {}
// as a class
class TestRegistry {
init(gulpInst) {}
get(name) {}
set(name, fn) {}
tasks() {}
}
```

If a registry instance passed to registry() doesn't have all four methods, an error will be thrown.

Registration

If we want to register our example registry from above, we will need to pass an instance of it to registry().

```
const { registry } = require('gulp');
// ... TestRegistry setup code
// good!
registry(new TestRegistry())
// bad!
registry(TestRegistry())
// This will trigger an error: 'Custom registries must be instantiated, but it looks like you passed a constructor'
```

Methods

init(gulpInst)

The init() method of a registry is called at the very end of
the registry() function. The gulp instance passed as the only argument (gulpInst) can be used to pre-define tasks usinggulpInst.task(taskName, fn).

Parameters

parameter	type	note
gulpInst	object	Instance of gulp.

get(name)

The get() method receives a task name for the custom registry to resolve and return, or undefined if no task with that name exists.

Parameters

parameter	type	note
name	string	Name of the task to be retrieved.

set(name, fn)

The set() method receives a task name and fn. This is called internally by task() to provide user-registered tasks to custom registries.

Parameters

parameter	type	note
name	string	Name of the task to be set.
fn	function	Task function to be set.

tasks()

Must return an object listing all tasks in the registry.

Use Cases

Sharing Tasks

To share common tasks with all your projects, you can expose
an init method on the registry and it will receive the an instance of gulp as the only argument. You can then use gulpInst.task(name, fn) to register pre-defined tasks.

For example, you might want to share a clean task:

```
const fs = require('fs');
const util = require('util');
const DefaultRegistry = require('undertaker-registry');
const del = require('del');
function CommonRegistry(opts){
DefaultRegistry.call(this);
opts = opts || {};
this.buildDir = opts.buildDir || './build';
}
util.inherits(CommonRegistry, DefaultRegistry);
CommonRegistry.prototype.init = function(gulpInst) {
const buildDir = this.buildDir;
const exists = fs.existsSync(buildDir);
if(exists){
throw new Error('Cannot initialize common tasks. ' + buildDir + '
directory exists.');
}
gulpInst.task('clean', function(){
return del([buildDir]);
});
}
module.exports = CommonRegistry;
```

Then to use it in a project:

```
const { registry, series, task } = require('gulp');
const CommonRegistry = require('myorg-common-tasks');
registry(new CommonRegistry({ buildDir: '/dist' }));
task('build', series('clean', function build(cb) {
// do things
cb();
}));
```

Sharing Functionality

By controlling how tasks are added to the registry, you can decorate them. For example, if you wanted all tasks to share some data, you can use a custom registry to bind them to that data. Be sure to return the altered task, as per the description of registry methods above:

```
const { registry, series, task } = require('gulp');
const util = require('util');
const DefaultRegistry = require('undertaker-registry');
// Some task defined somewhere else
const BuildRegistry = require('./build.js');
const ServeRegistry = require('./serve.js');
function ConfigRegistry(config){
DefaultRegistry.call(this);
this.config = config;
}
util.inherits(ConfigRegistry, DefaultRegistry);
ConfigRegistry.prototype.set = function set(name, fn) {
// The `DefaultRegistry` uses `this._tasks` for storage.
var task = this._tasks[name] = fn.bind(this.config);
return task;
};
registry(new BuildRegistry());
registry(new ServeRegistry());
// `registry` will reset each task in the registry with
// `ConfigRegistry.prototype.set` which will bind them to the config
object.
registry(new ConfigRegistry({
src: './src',
build: './build',
bindTo: '0.0.0.0:8888'
}));
task('default', series('clean', 'build', 'serve', function(cb) {
console.log('Server bind to ' + this.bindTo);
console.log('Serving' + this.build);
cb();
}));
```

Examples

undertaker-registry: The Gulp 4 default registry.
undertaker-common-tasks: Proof-of-concept custom registry that pre-defines tasks.
undertaker-task-metadata: Proof-of-concept custom registry that attaches metadata to each task.