

Controllers

<code>Controller Instance</code>	<code>var controller = this;</code>	returns instance
<code>controller.body</code>	Gets the parsed data from the request body.	returns object
<code>controller.ip</code>	Gets the HTTP host address	returns string
<code>controller.language</code>	Gets/Sets the controller language for the localization	returns string
<code>controller.params</code>	Gets a dynamic params from URL	returns object
<code>controller.query</code>	Gets the parsed data from the URL address query	returns object
<code>controller.req</code>	Gets the HTTP request	returns object
<code>controller.res</code>	Gets the HTTP response	returns object
<code>controller.url</code>	Gets a relative lower-case URL	returns string
<code>controller.user</code>	Gets the user instance if exists	returns object
<code>controller.callBack([view_name])</code>	Returns data as JSON or view	returns Function
<code>controller.sendFile(filename, [callback])</code>	Sends the file into response	returns controller instance
<code>controller.sendFilefs(name, id, [callback])</code>	Sends a stored file in FileStorage	returns controller instance
<code>controller.hostname([path])</code>	Gets the hostname	returns string
<code>controller.layout(name)</code>	Sets a layout before the current view	returns controller instance
<code>controller.invalidate(error)</code>	Creates an ErrorBuilder instance	returns ErrorBuilder
<code>controller.plain(text, [headers])</code>	Sends the plain text	returns controller instance
<code>controller.proxy(opts/url)</code>	Creates a proxy for the current request	returns object
<code>controller.redirect(url, [permanent])</code>	Creates a redirection	returns controller instance
<code>controller.succeed([success], [value])</code>	Alias for <code>controller.json(SUCCESS (true))</code>	returns controller instance
<code>controller.throw404([problem])</code>	Sends a 404 error - Not found into response	returns controller instance
<code>controller.view(name/url, [model])</code>	Sends the view	returns controller instance or string

Info

Website	https://totaljs.com
Docs	https://docs.totaljs.com
Total.js Cloud	https://platform.totaljs.com
Telegram Community	https://t.me/totaljs
Blog	https://blog.totaljs.com
Tutorials	Youtube Videos

Setup

```
$ sudo npm install -g total4
$ total4 create empty myapp
```

Create an empty project from template.

\$(delegate) (cont)

To list available commands: `$ total4 help`
 To list available templates: `$ total4 templates`
 To create from template: `$ total4 create <template_name> <path/to/empty/ folder>`

Start Script

```
// index.js
var opt = {};
opt.port = 5000;
require('total4/debug')(opt);
```

[Read more options in docs](#)

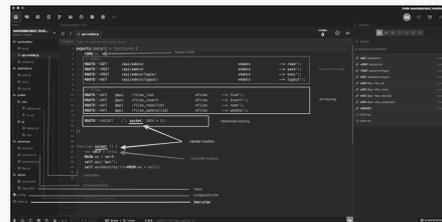
\$(delegate)

<code>\$.body</code>	Returns body data from the request	returns object
<code>\$.controller</code>	Contains current instance of Controller	returns controller instance
<code>\$.files</code>	Returns list of uploaded files	returns Array HttpFile
<code>\$.headers</code>	Returns current request headers	returns object
<code>\$.id</code>	Returns the value of the first parsed dynamic parameter	returns string
<code>\$.ip</code>	Returns IP address from the Request	returns string
<code>\$.language</code>	Returns parsed language from the Request	returns String
<code>\$.model</code>	Returns the current model according to the Schema	returns object
<code>\$.params</code>	Returns parsed dynamic parameters from URL	returns object
<code>\$.req</code>	Returns the current Request instance	returns Request

<code>\$.res</code>	Returns the current Response instance	returns Response
<code>\$.user</code>	Returns user instance from the Request	returns object
<code>\$.action(schema, [data])</code>	This method is an alias from the <code>CALL()</code> method	returns Callback
<code>\$.callback([error], [value])</code>	Performs a callback	returns Function
<code>\$.done([arg])</code>	It wraps <code>\$.success()</code> and <code>\$.invalid()</code> method into the one function	returns Function (response)
<code>\$.invalid(name, [error])</code>	This method appends an Error to <code>ErrorBuilder</code>	returns Error
<code>\$.redirect(url)</code>	Performs redirect directly from the operation	returns Function
<code>\$.success([is], [value])</code>	This method returns predefined success object	returns Object

[Read more in docs](#)

Project Structure (MVC)



Total.js project contains a simple directory structure, and all directories are optional. You can choose what you want to use and when.

[Read more in docs](#)

Schemas

<code>schema s.f ields</code>	Returns all fields names from the schema
<code>schema s.d efi ne(name, type)</code>	Defines the fields for the current entity
<code>schema s.a cti on(name, opti ons)</code>	Registers a new schema action
<code>action options</code>	Options for registering new schema action
<code>option s.name {String}</code>	(optional), a human readable action name
<code>option s.p arams {String}</code>	(optional), a parameter for the schema



By **Louis Bertson**
cheatography.com/louis-bertson/

Not published yet.
Last updated 7th April, 2023.
Page 2 of 4.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Schemas (cont)

<code>option s.query {String}</code>	(optional), query arguments for the schema
<code>option s.input {String}</code>	(optional), input data schema
<code>option s.action {Function(\$, mode 1)}</code>	(required), a function that will be executed

Schemas are one of the most used parts of the Total.js framework. Schemas offer you full control over incoming data.

C

By **Louis Bertson**
cheatography.com/louis-bertson/

Not published yet.
Last updated 7th April, 2023.
Page 3 of 4.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>