Artisan

```
// Added in 5.1.11:http://laravel.com/docs/5.1/authorization#creating-policies
php artisan make policy PostPolicy
// Displays help for a given command
php artisan   help OR  h
// Do not output any message
php artisan   quiet OR  q
// Display this application version
php artisan   version OR  V
// Do not ask any interactive question
php artisan   no interaction OR  n
// Force ANSI output
php artisan   ansi
// Disable ANSI output
php artisan   no ansi
// The environment the command should run under
php artisan   env
// -v|vv|vvv Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
php artisan   verbose
// Remove the compiled class file
php artisan clear compiled
// Display the current framework environment
php artisan env
// Displays help for a command
php artisan help
// Lists commands
php artisan list
// Interact with your application
php artisan tinker
// Put the application into maintenance mode
php artisan down
// Bring the application out of maintenance mode
php artisan up
// Optimize the framework for better performance
// --force    Force the compiled class file to be written.
// --psr      Do not optimize Composer dump-autoload.
php artisan optimize    force     psr
```

```
// Serve the application on the PHP development server
php artisan serve
// Change the default port
php artisan serve   port 8080
// Get it to work outside localhost
php artisan serve   host 0.0 0.0
// Set the application namespace
php artisan app name namespace
// Flush expired password reset tokens
php artisan auth clear resets
// Flush the application cache
php artisan cache clear
// Create a migration for the cache database table
php artisan cache table
// Create a cache file for faster configuration loading
php artisan config cache
// Remove the configuration cache file
php artisan config clear
// In program
$exitCode   Artisan  call 'config:cache'
// Seed the database with records
// --class      The class name of the root seeder (default: "DatabaseSeeder")
// --database   The database connection to seed
// --force      Force the operation to run when in production.
php artisan db seed    class  "..."      database  "..."      force

// Generate the missing events and handlers based on registration
php artisan event generate

// Create a new command handler class
// --command      The command class the handler handles.
php artisan handler command    command "..."  name
// Create a new event handler class
// --event        The event class the handler handles.
// --queued       Indicates the event handler should be queued.
php artisan handler event    event "..."     queued  name
```

```
// Set the application key
php artisan key generate

// By default, this creates a self-handling command that isn't pushed to the queue.
// Pass this the --handler flag to generate a handler, and the --queued flag to make it
queued.
php artisan make command    handler     queued  name
// Create a new Artisan command
//  --command     The terminal command that should be assigned. (default: "command:name")
make console    command  "..."   name
// Create a new resourceful controller
// --plain      Generate an empty controller class.
php artisan make controller    plain  name
php artisan make controller App\\Admin\\Http\\Controllers\\DashboardController
// Create a new event class
php artisan make event name
// Create a new middleware class
php artisan make middleware name
// Create a new migration file
// --create     The table to be created.
// --table      The table to migrate.
php artisan make migration    create  "..."      table  "..."   name
// Create a new Eloquent model class
php artisan make model name
// Create a new service provider class
php artisan make provider name
// Create a new form request class
php artisan make request name
// Database migrations
// --database   The database connection to use.
// --force      Force the operation to run when in production.
// --path       The path of migrations files to be executed.
// --pretend    Dump the SQL queries that would be run.
// --seed       Indicates if the seed task should be re-run.
php artisan migrate    database  "..."      force     path  "..."      pretend     seed
// Create the migration repository
php artisan migrate install    database  "..."
```

```
// Create a new migration file
// --seeder     The class name of the root seeder.
php artisan migrate refresh    database  "..."      force     seed     seeder  "..."
// Rollback all database migrations
// --pretend    Dump the SQL queries that would be run.
php artisan migrate reset    database  "..."      force     pretend
// Rollback the last database migration
php artisan migrate rollback    database  "..."      force     pretend
// Show a list of migrations up/down
php artisan migrate status
// Create a migration for the queue jobs database table
php artisan queue table
// Listen to a given queue
// --queue      The queue to listen on
// --delay      Amount of time to delay failed jobs (default: 0)
// --memory     The memory limit in megabytes (default: 128)
// --timeout    Seconds a job may run before timing out (default: 60)
// --sleep      Seconds to wait before checking queue for jobs (default: 3)
// --tries      Number of times to attempt a job before logging it failed (default: 0)
php artisan queue listen    queue  "..."      delay  "..."      memory  "..."
timeout  "..."      sleep  "..."      tries  "..."    connection
// List all of the failed queue jobs
php artisan queue failed
// Create a migration for the failed queue jobs database table
php artisan queue failed table
// Flush all of the failed queue jobs
php artisan queue flush
// Delete a failed queue job
php artisan queue forget
// Restart queue worker daemons after their current job
php artisan queue restart
// Retry a failed queue job(id: The ID of the failed job)
php artisan queue retry id
// Subscribe a URL to an Iron.io push queue
// queue: The name of Iron.io queue.
// url: The URL to be subscribed.
// --type       The push type for the queue.
```

```
php artisan queue subscribe    type  "..."   queue url
// Process the next job on a queue
// --queue      The queue to listen on
// --daemon     Run the worker in daemon mode
// --delay      Amount of time to delay failed jobs (default: 0)
// --force      Force the worker to run even in maintenance mode
// --memory     The memory limit in megabytes (default: 128)
// --sleep      Number of seconds to sleep when no job is available (default: 3)
// --tries      Number of times to attempt a job before logging it failed (default: 0)
php artisan queue work    queue  "..."      daemon     delay  "..."      force
memory  "..."      sleep  "..."      tries  "..."    connection

// Create a route cache file for faster route registration
php artisan route cache
// Remove the route cache file
php artisan route clear
// List all registered routes
php artisan route list

// Run the scheduled commands
php artisan schedule run

// Create a migration for the session database table
php artisan session table

// Publish any publishable assets from vendor packages
// --force        Overwrite any existing files.
// --provider     The service provider that has assets you want to publish.
// --tag          The tag that has assets you want to publish.
php artisan vendor publish    force     provider  "..."      tag  "..."
php artisan tail    path  "..."      lines  "..."    connection
```

Composer

```
composer create project laravel laravel folder_name
composer install
composer update
composer dump autoload    optimize
composer self update
```

```
composer require  options        vendor packages
```

Config

```
Config  get 'app.timezone'
//get with Default value
Config  get 'app.timezone'  'UTC'
//set Configuration
Config  set 'database.default'  'sqlite'
```

Route

```
Route  get 'foo'  function
Route  get 'foo'  'ControllerName@function'
Route  controller 'foo'  'FooController'
```

RESTful Controllers

```
Route  resource 'posts' 'PostsController'
//Specify a subset of actions to handle on the route
Route  resource 'photo'  'PhotoController'  'only'     'index'  'show'
Route  resource 'photo'  'PhotoController'  'except'     'update'  'destroy'
```

Triggering Errors

```
App  abort 404
$handler  missing      in ErrorServiceProvider  boot
throw new NotFoundHttpException
```

Route Parameters

```
Route  get 'foo/{bar}'  function $bar
Route  get 'foo/{bar?}'  function $bar   'bar'
```

HTTP Verbs

```
Route  any 'foo'  function
Route  post 'foo'  function
Route  put 'foo'  function
Route  patch 'foo'  function
Route  delete 'foo'  function
// RESTful actions
Route  resource 'foo'  'FooController'
// Registering A Route For Multiple Verbs
Route  match  'get'  'post'   '/'  function
```

Secure Routes(TBD)

```
Route  get 'foo'  array 'https'  function
```

Route Constraints

```
Route  get 'foo/{bar}'  function $bar
  where 'bar'  '[0-9]+'
Route  get 'foo/{bar}/{baz}'  function $bar  $baz
  where array 'bar'    '[0-9]+'  'baz'    '[A-Za-z]'

// Set a pattern to be used across routes
Route  pattern 'bar'  '[0-9]+'
```

HTTP Middleware

```
// Assigning Middleware To Routes
Route  get 'admin/profile'   'middleware'    'auth'  function
Route  get 'admin/profile'  function       middleware 'auth'
```

Named Routes

```
Route  currentRouteName
Route  get 'foo/bar'  array 'as'    'foobar'  function
Route  get 'user/profile'
  'as'    'profile'  'uses'    'UserController@showProfile'

Route  get 'user/profile'  'UserController@showProfile'   name 'profile'
$url   route 'profile'
$redirect   redirect    route 'profile'
```

Route Prefixing

```
Route  group  'prefix'    'admin'   function

  Route  get 'users'  function
      return 'Matches The "/admin/users" URL'
```

Route Namespacing

```
// This route group will carry the namespace 'Foo\Bar'
Route  group array 'namespace'    'Foo\Bar'   function
```

Sub-Domain Routing

```
// {sub} will be passed to the closure
```

```
Route  group array 'domain'    '{sub}.example.com'   function
```

Environment

```
$environment   app    environment
$environment   App  environment
$environment   $app  environment
// The environment is local
if  $app  environment 'local'
// The environment is either local OR staging...
if  $app  environment 'local'  'staging'
```

Log

```
// The logger provides the seven logging levels defined in RFC 5424:
// debug, info, notice, warning, error, critical, and alert.
Log  info 'info'
Log  info 'info' array 'context'  'additional info'
Log  error 'error'
Log  warning 'warning'
// get monolog instance
Log  getMonolog
// add listener
Log  listen function $level  $message  $context
```

Query Logging

```
// enable the log
DB  connection    enableQueryLog
// get an array of the executed queries
DB  getQueryLog
```

URL

```
URL  full
URL  current
URL  previous
URL  to 'foo/bar'  $parameters  $secure
URL  action 'NewsController@item'   'id'  123
// need be in appropriate namespace
URL  action 'Auth\AuthController@logout'
URL  action 'FooController@method'  $parameters  $absolute
```

```
URL  route 'foo'  $parameters  $absolute
URL  secure 'foo/bar'  $parameters
URL  asset 'css/foo.css'  $secure
URL  secureAsset 'css/foo.css'
URL  isValidUrl 'http://example.com'
URL  getRequest
URL  setRequest $request
```

Event

```
Event  fire 'foo.bar'  array $bar
// Register an event listener with the dispatcher.
// void listen(string|array $events, mixed $listener, int $priority)
Event  listen 'App\Events\UserSignup'  function $bar
Event  listen 'foo.*'  function $bar
Event  listen 'foo.bar'  'FooHandler'  10
Event  listen 'foo.bar'  'BarHandler'  5
// Stopping The Propagation Of An Event
// You may do so by returning false from your handler.
Event  listen 'foor.bar'  function $event   return false
Event  subscribe 'UserEventHandler'
```

DB

Basic Database Usage

```
DB  connection 'connection_name'
// Running A Select Query
$results   DB  select 'select * from users where id = ?'   1
$results   DB  select 'select * from users where id = :id'   'id'    1
// Running A General Statement
DB  statement 'drop table users'
// Listening For Query Events
DB  listen function $sql  $bindings  $time   code_here
// Database Transactions
DB  transaction function

  DB  table 'users'   update  'votes'    1
  DB  table 'posts'   delete

DB  beginTransaction
```

```
DB  rollBack
DB  commit
```

Query Builder

```
// Retrieving All Rows From A Table
DB  table 'name'   get
// Chunking Results From A Table
DB  table 'users'   chunk 100  function $users

  foreach  $users as $user

      //


// Retrieving A Single Row From A Table
$user   DB  table 'users'   where 'name'  'John'   first
DB  table 'name'   first
// Retrieving A Single Column From A Row
$name   DB  table 'users'   where 'name'  'John'   pluck 'name'
DB  table 'name'   pluck 'column'
// Retrieving A List Of Column Values
$roles   DB  table 'roles'   lists 'title'
$roles   DB  table 'roles'   lists 'title'  'name'
// Specifying A Select Clause
$users   DB  table 'users'   select 'name'  'email'   get
$users   DB  table 'users'   distinct    get
$users   DB  table 'users'   select 'name as user_name'   get
// Adding A Select Clause To An Existing Query
$query   DB  table 'users'   select 'name'
$users   $query  addSelect 'age'   get
// Using Where Operators
$users   DB  table 'users'   where 'votes'  '>'  100   get
$users   DB  table 'users'
                where 'votes'  '>'  100
                orWhere 'name'  'John'
                get
$users   DB  table 'users'
                whereBetween 'votes'   1  100    get
```

```
$users   DB  table 'users'
               whereNotBetween 'votes'   1  100    get
$users   DB  table 'users'
               whereIn 'id'   1  2  3    get
$users   DB  table 'users'
               whereNotIn 'id'   1  2  3    get
$users   DB  table 'users'
               whereNull 'updated_at'   get
DB  table 'name'   whereNotNull 'column'   get
// Dynamic Where Clauses
$admin   DB  table 'users'   whereId 1   first
$john   DB  table 'users'
               whereIdAndEmail 2  'john@doe.com'
               first
$jane   DB  table 'users'
               whereNameOrAge 'Jane'  22
               first
// Order By, Group By, And Having
$users   DB  table 'users'
               orderBy 'name'  'desc'
               groupBy 'count'
               having 'count'  '>'  100
               get
DB  table 'name'   orderBy 'column'   get
DB  table 'name'   orderBy 'column' 'desc'   get
DB  table 'name'   having 'count'  '>'  100   get
// Offset & Limit
$users   DB  table 'users'   skip 10   take 5   get
```

Joins

```
// Basic Join Statement
DB  table 'users'
            join 'contacts'  'users.id'  '='  'contacts.user_id'
            join 'orders'  'users.id'  '='  'orders.user_id'
            select 'users.id'  'contacts.phone'  'orders.price'
            get
// Left Join Statement
DB  table 'users'
```

```
        leftJoin 'posts'  'users.id'  '='  'posts.user_id'
        get
// select * from users where name = 'John' or (votes > 100 and title <> 'Admin')
DB  table 'users'
            where 'name'  '='  'John'
            orWhere function $query

              $query  where 'votes'  '>'  100
                      where 'title'  '<>'  'Admin'

            get
```

Aggregates

```
$users   DB  table 'users'   count
$price   DB  table 'orders'   max 'price'
$price   DB  table 'orders'   min 'price'
$price   DB  table 'orders'   avg 'price'
$total   DB  table 'users'   sum 'votes'
```

Raw Expressions

```
$users   DB  table 'users'
                    select DB  raw 'count(*) as user_count, status'
                    where 'status'  '<>'  1
                    groupBy 'status'
                    get
// return rows
DB  select 'select * from users where id = ?'  array 'value'
// return nr affected rows
DB  insert 'insert into foo set bar=2'
DB  update 'update foo set bar=2'
DB  delete 'delete from bar'
// returns void
DB  statement 'update foo set bar=2'
// raw expression inside a statement
DB  table 'name'   select DB  raw 'count(*) as count, column2'    get
```

Inserts / Updates / Deletes / Unions / Pessimistic Locking

```
// Inserts
DB  table 'users'   insert
```

```
    'email'    'john@example.com'  'votes'    0

$id   DB  table 'users'   insertGetId
    'email'    'john@example.com'  'votes'    0

DB  table 'users'   insert
    'email'    'taylor@example.com'  'votes'    0
    'email'    'dayle@example.com'  'votes'    0

// Updates
DB  table 'users'
            where 'id'  1
            update  'votes'    1
DB  table 'users'   increment 'votes'
DB  table 'users'   increment 'votes'  5
DB  table 'users'   decrement 'votes'
DB  table 'users'   decrement 'votes'  5
DB  table 'users'   increment 'votes'  1   'name'    'John'
// Deletes
DB  table 'users'   where 'votes'  '<'  100   delete
DB  table 'users'   delete
DB  table 'users'   truncate
// Unions
// The unionAll() method is also available, and has the same method signature as union.
$first   DB  table 'users'   whereNull 'first_name'
$users   DB  table 'users'   whereNull 'last_name'   union $first   get
// Pessimistic Locking
DB  table 'users'   where 'votes'  '>'  100   sharedLock    get
DB  table 'users'   where 'votes'  '>'  100   lockForUpdate    get
```

Model

Basic Usage

```
// Defining An Eloquent Model
class User extends Model
// generate Eloquent models
php artisan make model User
// specify a custom table name
class User extends Model
```

```
  protected $table   'my_users'
```

More

```
Model  create array 'key'    'value'
// Find first matching record by attributes or create
Model  firstOrCreate array 'key'    'value'
// Find first record by attributes or instantiate
Model  firstOrNew array 'key'    'value'
// Create or update a record matching attibutes, and fill with values
Model  updateOrCreate array 'search_key'    'search_value'   array 'key'    'value'
// Fill a model with an array of attributes, beware of mass assignment!
Model  fill $attributes
Model  destroy 1
Model  all
Model  find 1
// Find using dual primary key
Model  find array 'first'  'last'
// Throw an exception if the lookup fails
Model  findOrFail 1
// Find using dual primary key and throw exception if the lookup fails
Model  findOrFail array 'first'  'last'
Model  where 'foo'  '='  'bar'   get
Model  where 'foo'  '='  'bar'   first
Model  where 'foo'  '='  'bar'   exists
// dynamic
Model  whereFoo 'bar'   first
// Throw an exception if the lookup fails
Model  where 'foo'  '='  'bar'   firstOrFail
Model  where 'foo'  '='  'bar'   count
Model  where 'foo'  '='  'bar'   delete
//Output raw query
Model  where 'foo'  '='  'bar'   toSql
Model  whereRaw 'foo = bar and cars = 2'  array 20    get
Model  on 'connection-name'   find 1
Model  with 'relation'   get
Model  all    take 10
Model  all    skip 10
```

```
// Default Eloquent sort is ascendant
Model  all    orderBy 'column'
Model  all    orderBy 'column' 'desc'
```

Soft Delete

```
Model  withTrashed    where 'cars'  2   get
// Include the soft deleted models in the results
Model  withTrashed    where 'cars'  2   restore
Model  where 'cars'  2   forceDelete
// Force the result set to only included soft deletes
Model  onlyTrashed    where 'cars'  2   get
```

Relationships

```
// One To One - User::phone()
  return $this  hasOne 'App\Phone'  'foreign_key'  'local_key'
// One To One - Phone::user(), The Inverse Of The Relation
return $this  belongsTo 'App\User'  'foreign_key'  'other_key'

// One To Many - Post::comments()
return $this  hasMany 'App\Comment'  'foreign_key'  'local_key'
//  One To Many - Comment::post()
return $this  belongsTo 'App\Post'  'foreign_key'  'other_key'

// Many To Many - User::roles();
return $this  belongsToMany 'App\Role'  'user_roles'  'user_id'  'role_id'
// Many To Many - Role::users();
return $this  belongsToMany 'App\User'
// Many To Many - Retrieving Intermediate Table Columns
$role  pivot  created_at
// Many To Many - Pivot table with extra attributes
return $this  belongsToMany 'App\Role'   withPivot 'column1'  'column2'
// Many To Many - Automatically maintained created_at and updated_at timestamps
return $this  belongsToMany 'App\Role'   withTimestamps

// Has Many Through - Country::posts(), A Country model have
// many Post models through an intermediate User model (User::country_id)
return $this  hasManyThrough 'App\Post'  'App\User'  'country_id'  'user_id'
```

```
// Polymorphic Relations - Photo::imageable()
return $this  morphTo
// Polymorphic Relations - Staff::photos()
return $this  morphMany 'App\Photo'  'imageable'
// Polymorphic Relations - Product::photos()
return $this  morphMany 'App\Photo'  'imageable'
// Polymorphic Relations - Register the morphMap in your AppServiceProvider
Relation  morphMap
    'Post'    App\Post  class
    'Comment'    App\Comment  class


// Many To Many Polymorphic Relations - Tables: posts,videos,tags,taggables
// Post::tags()
return $this  morphToMany 'App\Tag'  'taggable'
// Video::tags()
return $this  morphToMany 'App\Tag'  'taggable'
// Tag::posts()
return $this  morphedByMany 'App\Post'  'taggable'
// Tag::videos()
return $this  morphedByMany 'App\Video'  'taggable'

// Querying Relations
$user  posts    where 'active'  1   get
// Retrieve all posts that have at least one comment...
$posts   App\Post  has 'comments'   get
// Retrieve all posts that have three or more comments...
$posts   Post  has 'comments'  '>='  3   get
// Retrieve all posts that have at least one comment with votes...
$posts   Post  has 'comments.votes'   get
// Retrieve all posts with at least one comment containing words like foo%
$posts   Post  whereHas 'comments'  function  $query
    $query  where 'content'  'like'  'foo%'
    get

// Eager Loading
$books   App\Book  with 'author'   get
```

```
$books = App\Book with 'author' 'publisher' get
$books = App\Book with 'author.contacts' get

// Lazy Eager Loading
$books load 'author' 'publisher'

// Inserting Related Models
$comment = new App\Comment 'message' 'A new comment.'
$post comments save $comment
// save multiple related models
$post comments saveMany
    new App\Comment 'message' 'A new comment.'
    new App\Comment 'message' 'Another comment.'

$post comments create 'message' 'A new comment.'

// Updating a belongsTo relationship
$user account associate $account
$user save
$user account dissociate
$user save

// Inserting Related Models - Many To Many Relationships
$user roles attach $roleId
$user roles attach $roleId 'expires' $expires
// Detach a single role from the user...
$user roles detach $roleId
// Detach all roles from the user...
$user roles detach
$user roles detach 1 2 3
$user roles attach 1 'expires' $expires 2 3

// Any IDs that are not in the given array will be removed from the intermediate table.
$user roles sync 1 2 3
// You may also pass additional intermediate table values with the IDs:
$user roles sync 1 'expires' true 2 3
```

Events

```
Model  creating function $model
Model  created function $model
Model  updating function $model
Model  updated function $model
Model  saving function $model
Model  saved function $model
Model  deleting function $model
Model  deleted function $model
Model  observe new FooObserver
```

Eloquent Configuration

```
// Disables mass assignment exceptions from being thrown from model inserts and updates
Eloquent  unguard
// Renables any ability to throw mass assignment exceptions
Eloquent  reguard
```

Pagination

```
// Auto-Magic Pagination
Model  paginate 15
Model  where 'cars'  2   paginate 15
// "Next" and "Previous" only
Model  where 'cars'  2   simplePaginate 15
// Manual Paginator
Paginator  make $items  $totalItems  $perPage
// Print page navigators in view
$variable  links
```

Lang

```
App  setLocale 'en'
Lang  get 'messages.welcome'
Lang  get 'messages.welcome'  array 'foo'    'Bar'
Lang  has 'messages.welcome'
Lang  choice 'messages.apples'  10
// Lang::get alias
trans 'messages.welcome'
```

File

```
File  exists 'path'
File  get 'path'
File  getRemote 'path'
// Get a file's contents by requiring it
File  getRequire 'path'
// Require the given file once
File  requireOnce 'path'
// Write the contents of a file
File  put 'path'  'contents'
// Append to a file
File  append 'path'  'data'
// Delete the file at a given path
File  delete 'path'
// Move a file to a new location
File  move 'path'  'target'
// Copy a file to a new location
File  copy 'path'  'target'
// Extract the file extension from a file path
File  extension 'path'
// Get the file type of a given file
File  type 'path'
// Get the file size of a given file
File  size 'path'
// Get the file's last modification time
File  lastModified 'path'
// Determine if the given path is a directory
File  isDirectory 'directory'
// Determine if the given path is writable
File  isWritable 'path'
// Determine if the given path is a file
File  isFile 'file'
// Find path names matching a given pattern.
File  glob $patterns  $flag
// Get an array of all files in a directory.
File  files 'directory'
// Get all of the files from the given directory (recursive).
File  allFiles 'directory'
```

```
// Get all of the directories within a given directory.
File  directories 'directory'
// Create a directory
File  makeDirectory 'path'   $mode   0777  $recursive   false
// Copy a directory from one location to another
File  copyDirectory 'directory'  'destination'  $options   null
// Recursively delete a directory
File  deleteDirectory 'directory'  $preserve   false
// Empty the specified directory of all files and folders
File  cleanDirectory 'directory'
```

UnitTest

Install and run

```
// add to composer and update:
"phpunit/phpunit"  "4.0.*"
// run tests (from project root)
  vendor bin phpunit
```

Asserts

```
$this  assertTrue true
$this  assertEquals 'foo'  $bar
$this  assertCount 1 $times
$this  assertResponseOk
$this  assertResponseStatus 403
$this  assertRedirectedTo 'foo'
$this  assertRedirectedToRoute 'route.name'
$this  assertRedirectedToAction 'Controller@method'
$this  assertViewHas 'name'
$this  assertViewHas 'age'  $value
$this  assertSessionHasErrors
// Asserting the session has errors for a given key...
$this  assertSessionHasErrors 'name'
// Asserting the session has errors for several keys...
$this  assertSessionHasErrors array 'name'  'age'
$this  assertHasOldInput
```

Calling routes

```
$response   $this  call $method  $uri  $parameters  $files  $server  $content
$response   $this  callSecure 'GET'  'foo/bar'
```

```
$this  session  'foo'    'bar'
$this  flushSession
$this  seed
$this  seed $connection
```

SSH

Executing Commands

```
SSH  run array $commands
SSH  into $remote   run array $commands   // specify remote, otherwise assumes default
SSH  run array $commands  function $line

  echo $line PHP_EOL
```

Tasks

```
// define
SSH  define $taskName  array $commands
// execute
SSH  task $taskName  function $line

  echo $line PHP_EOL
```

SFTP Uploads

```
SSH  put $localFile  $remotePath
SSH  putString $string  $remotePath
```

Schema

```
// Indicate that the table needs to be created
Schema  create 'table'  function $table

  $table  increments 'id'

// Specify a Connection
Schema  connection 'foo'   create 'table'  function $table
// Rename the table to a given name
Schema  rename $from  $to
// Indicate that the table should be dropped
Schema  drop 'table'
```

```
// Indicate that the table should be dropped if it exists
Schema  dropIfExists 'table'
// Determine if the given table exists
Schema  hasTable 'table'
// Determine if the given table has a given column
Schema  hasColumn 'table'  'column'
// Update an existing table
Schema  table 'table'  function $table
// Indicate that the given columns should be renamed
$table  renameColumn 'from'  'to'
// Indicate that the given columns should be dropped
$table  dropColumn string array
// The storage engine that should be used for the table
$table  engine   'InnoDB'
// Only work on MySQL
$table  string 'name'   after 'email'
```

Indexes

```
$table  string 'column'   unique
$table  primary 'column'
// Creates a dual primary key
$table  primary array 'first'  'last'
$table  unique 'column'
$table  unique 'column'  'key_name'
// Creates a dual unique index
$table  unique array 'first'  'last'
$table  unique array 'first'  'last'   'key_name'
$table  index 'column'
$table  index 'column'  'key_name'
// Creates a dual index
$table  index array 'first'  'last'
$table  index array 'first'  'last'   'key_name'
$table  dropPrimary array 'column'
$table  dropPrimary 'table_column_primary'
$table  dropUnique array 'column'
$table  dropUnique 'table_column_unique'
$table  dropIndex array 'column'
$table  dropIndex 'table_column_index'
```

Foreign Keys

```
$table  foreign 'user_id'   references 'id'   on 'users'
$table  foreign 'user_id'   references 'id'   on 'users'
 onDelete 'cascade' 'restrict' 'set null' 'no action'
$table  foreign 'user_id'   references 'id'   on 'users'
 onUpdate 'cascade' 'restrict' 'set null' 'no action'
$table  dropForeign array 'user_id'
$table  dropForeign 'posts_user_id_foreign'
```

Column Types

```
// Increments
$table  increments 'id'
$table  bigIncrements 'id'

// Numbers
$table  integer 'votes'
$table  tinyInteger 'votes'
$table  smallInteger 'votes'
$table  mediumInteger 'votes'
$table  bigInteger 'votes'
$table  float 'amount'
$table  double 'column'  15  8
$table  decimal 'amount'  5  2

//String and Text
$table  char 'name'  4
$table  string 'email'
$table  string 'name'  100
$table  text 'description'
$table  mediumText 'description'
$table  longText 'description'

//Date and Time
$table  date 'created_at'
$table  dateTime 'created_at'
$table  time 'sunrise'
$table  timestamp 'added_on'
// Adds created_at and updated_at columns
```

```
$table  timestamps
$table  nullableTimestamps

// Others
$table  binary 'data'
$table  boolean 'confirmed'
// Adds deleted_at column for soft deletes
$table  softDeletes
$table  enum 'choices'  array 'foo'  'bar'
// Adds remember_token as VARCHAR(100) NULL
$table  rememberToken
// Adds INTEGER parent_id and STRING parent_type
$table  morphs 'parent'
  nullable
  default $value
  unsigned
```

Input

```
Input  get 'key'
// Default if the key is missing
Input  get 'key'  'default'
Input  has 'key'
Input  all
// Only retrieve 'foo' and 'bar' when getting input
Input  only 'foo'  'bar'
// Disregard 'foo' when getting input
Input  except 'foo'
Input  flush
```

Session Input (flash)

```
// Flash input to the session
Input  flash
// Flash only some of the input to the session
Input  flashOnly 'foo'  'bar'
// Flash only some of the input to the session
Input  flashExcept 'foo'  'baz'
// Retrieve an old input item
Input  old 'key' 'default_value'
```

Files

```
// Use a file that's been uploaded
Input  file 'filename'
// Determine if a file was uploaded
Input  hasFile 'filename'
// Access file properties
Input  file 'name'   getRealPath
Input  file 'name'   getClientOriginalName
Input  file 'name'   getClientOriginalExtension
Input  file 'name'   getSize
Input  file 'name'   getMimeType
// Move an uploaded file
Input  file 'name'   move $destinationPath
// Move an uploaded file
Input  file 'name'   move $destinationPath  $fileName
```

Cache

```
Cache  put 'key'  'value'  $minutes
Cache  add 'key'  'value'  $minutes
Cache  forever 'key'  'value'
Cache  remember 'key'  $minutes  function    return 'value'
Cache  rememberForever 'key'  function    return 'value'
Cache  forget 'key'
Cache  has 'key'
Cache  get 'key'
Cache  get 'key'  'default'
Cache  get 'key'  function    return 'default'
Cache  tags 'my-tag'   put 'key' 'value'  $minutes
Cache  tags 'my-tag'   has 'key'
Cache  tags 'my-tag'   get 'key'
Cache  tags 'my-tag'   forget 'key'
Cache  tags 'my-tag'   flush
Cache  increment 'key'
Cache  increment 'key'  $amount
Cache  decrement 'key'
Cache  decrement 'key'  $amount
Cache  section 'group'   put 'key'  $value
```

```
Cache  section 'group'   get 'key'
Cache  section 'group'   flush
```

Cookie

```
Cookie  get 'key'
Cookie  get 'key'  'default'
// Create a cookie that lasts for ever
Cookie  forever 'key'  'value'
// Create a cookie that lasts N minutes
Cookie  make 'key'  'value'  'minutes'
// Set a cookie before a response has been created
Cookie  queue 'key'  'value'  'minutes'
// Forget cookie
Cookie  forget 'key'
// Send a cookie with a response
$response   Response  make 'Hello World'
// Add a cookie to the response
$response  withCookie Cookie  make 'name'  'value'  $minutes
```

Session

```
Session  get 'key'
// Returns an item from the session
Session  get 'key'  'default'
Session  get 'key'  function    return 'default'
// Get the session ID
Session  getId
// Put a key / value pair in the session
Session  put 'key'  'value'
// Push a value into an array in the session
Session  push 'foo.bar' 'value'
// Returns all items from the session
Session  all
// Checks if an item is defined
Session  has 'key'
// Remove an item from the session
Session  forget 'key'
// Remove all of the items from the session
Session  flush
```

```
// Generate a new session identifier
Session  regenerate
// Flash a key / value pair to the session
Session  flash 'key'  'value'
// Reflash all of the session flash data
Session  reflash
// Reflash a subset of the current flash data
Session  keep array 'key1'  'key2'
```

Request

```
// url: http://xx.com/aa/bb
Request  url
// path: /aa/bb
Request  path
// getRequestUri: /aa/bb/?c=d
Request  getRequestUri
// Returns user's IP
Request  ip
// getUri: http://xx.com/aa/bb/?c=d
Request  getUri
// getQueryString: c=d
Request  getQueryString
// Get the port scheme of the request (e.g., 80, 443, etc.)
Request  getPort
// Determine if the current request URI matches a pattern
Request  is 'foo/*'
// Get a segment from the URI (1 based index)
Request  segment 1
// Retrieve a header from the request
Request  header 'Content-Type'
// Retrieve a server variable from the request
Request  server 'PATH_INFO'
// Determine if the request is the result of an AJAX call
Request  ajax
// Determine if the request is over HTTPS
Request  secure
// Get the request method
Request  method
```

```
// Checks if the request method is of specified type
Request  isMethod 'post'
// Get raw POST data
Request  instance    getContent
// Get requested response format
Request  format
// true if HTTP Content-Type header contains */json
Request  isJson
// true if HTTP Accept header is application/json
Request  wantsJson
```

Response

```
return Response  make $contents
return Response  make $contents  200
return Response  json array 'key'    'value'
return Response  json array 'key'    'value'
  setCallback Input  get 'callback'
return Response  download $filepath
return Response  download $filepath  $filename  $headers
// Create a response and modify a header value
$response   Response  make $contents  200
$response  header 'Content-Type'  'application/json'
return $response
// Attach a cookie to a response
return Response  make $content
  withCookie Cookie  make 'key'  'value'
```

Redirect

```
return Redirect  to 'foo/bar'
return Redirect  to 'foo/bar'   with 'key'  'value'
return Redirect  to 'foo/bar'   withInput Input  get
return Redirect  to 'foo/bar'   withInput Input  except 'password'
return Redirect  to 'foo/bar'   withErrors $validator
// Create a new redirect response to the previous location
return Redirect  back
// Create a new redirect response to a named route
return Redirect  route 'foobar'
```

```
return Redirect  route 'foobar'  array 'value'
return Redirect  route 'foobar'  array 'key'    'value'
// Create a new redirect response to a controller action
return Redirect  action 'FooController@index'
return Redirect  action 'FooController@baz'  array 'value'
return Redirect  action 'FooController@baz'  array 'key'    'value'
// If intended redirect is not defined, defaults to foo/bar.
return Redirect  intended 'foo/bar'
```

Container

```
App  bind 'foo'  function $app   return new Foo
App  make 'foo'
// If this class exists, it's returned
App  make 'FooBar'
// Register a shared binding in the container
App  singleton 'foo'  function    return new Foo
// Register an existing instance as shared in the container
App  instance 'foo'  new Foo
// Register a binding with the container
App  bind 'FooRepositoryInterface'  'BarRepository'
// Register a service provider with the application
App  register 'FooServiceProvider'
// Listen for object resolution
App  resolving function $object
```

Security

Hashing

```
Hash  make 'secretpassword'
Hash  check 'secretpassword'  $hashedPassword
Hash  needsRehash $hashedPassword
```

Encryption

```
Crypt  encrypt 'secretstring'
Crypt  decrypt $encryptedString
Crypt  setMode 'ctr'
Crypt  setCipher $cipher
```

Auth

Authentication

```
// Determine if the current user is authenticated
```

```
Auth  check
// Get the currently authenticated user
Auth  user
// Get the ID of the currently authenticated user
Auth  id
// Attempt to authenticate a user using the given credentials
Auth  attempt array 'email'    $email  'password'    $password
// 'Remember me' by passing true to Auth::attempt()
Auth  attempt $credentials  true
// Log in for a single request
Auth  once $credentials
// Log a user into the application
Auth  login User  find 1
// Log the given user ID into the application
Auth  loginUsingId 1
// Log the user out of the application
Auth  logout
// Validate a user's credentials
Auth  validate $credentials
// Attempt to authenticate using HTTP Basic Auth
Auth  basic 'username'
// Perform a stateless HTTP Basic login attempt
Auth  onceBasic
// Send a password reminder to a user
Password  remind $credentials  function $message  $user
```

Authorization

```
// Define abilities
Gate  define 'update-post'  'Class@method'
Gate  define 'update-post'  function  $user  $post
// Passing multiple argument
Gate  define 'delete-comment'  function  $user  $post  $comment

// Check abilities
Gate  denies 'update-post'  $post
Gate  allows 'update-post'  $post
Gate  check 'update-post'  $post
// Specified a user for checking
```

```
Gate  forUser $user   allows 'update-post'  $post
// Through User model, using Authorizable trait
User  find 1   can 'update-post'  $post
User  find 1   cannot 'update-post'  $post

// Intercepting Authorization Checks
Gate  before function  $user  $ability
Gate  after function  $user  $ability

// Chekcing in Blade template
@can 'update-post'  $post
@endcan
// with else
@can 'update-post'  $post
@else
@endcan

// Generate a Policy
php artisan make policy PostPolicy
// `policy` helper function
policy $post   update $user  $post

// Controller Authorization
$this  authorize 'update'  $post
// for $user
$this  authorizeForUser $user  'update'  $post
```

Mail

```
Mail  send 'email.view'  $data  function $message
Mail  send array 'html.view'  'text.view'   $data  $callback
Mail  queue 'email.view'  $data  function $message
Mail  queueOn 'queue-name'  'email.view'  $data  $callback
Mail  later 5  'email.view'  $data  function $message
// Write all email to logs instead of sending
Mail  pretend
```

Messages

```
// These can be used on the $message instance passed into Mail::send() or Mail::queue()
$message  from 'email@example.com'  'Mr. Example'
$message  sender 'email@example.com'  'Mr. Example'
$message  returnPath 'email@example.com'
$message  to 'email@example.com'  'Mr. Example'
$message  cc 'email@example.com'  'Mr. Example'
$message  bcc 'email@example.com'  'Mr. Example'
$message  replyTo 'email@example.com'  'Mr. Example'
$message  subject 'Welcome to the Jungle'
$message  priority 2
$message  attach 'foo\bar.txt'  $options
// This uses in-memory data as attachments
$message  attachData 'bar'  'Data Name'  $options
// Embed a file in the message and get the CID
$message  embed 'foo\bar.txt'
$message  embedData 'foo'  'Data Name'  $options
// Get the underlying Swift Message instance
$message  getSwiftMessage
```

Queue

```
Queue  push 'SendMail'  array 'message'    $message
Queue  push 'SendEmail@send'  array 'message'    $message
Queue  push function $job  use $id
// Same payload to multiple workers
Queue  bulk array 'SendEmail'  'NotifyUser'   $payload
// Starting the queue listener
php artisan queue listen
php artisan queue listen connection
php artisan queue listen   timeout 60
// Process only the first job on the queue
php artisan queue work
// Start a queue worker in daemon mode
php artisan queue work   daemon
// Create migration file for failed jobs
php artisan queue failed table
// Listing failed jobs
php artisan queue failed
// Delete failed job by id
```

```
php artisan queue forget 5
// Delete all failed jobs
php artisan queue flush
```

Validation

```
Validator  make
array 'key'    'Foo'
array 'key'    'required|in:Foo'

Validator  extend 'foo'  function $attribute  $value  $params
Validator  extend 'foo'  'FooValidator@validate'
Validator  resolver function $translator  $data  $rules  $msgs

return new FooValidator $translator  $data  $rules  $msgs
```

Rules

```
accepted
active_url
after YYYY MM DD
before YYYY MM DD
alpha
alpha_dash
alpha_num
array
between 1 10
confirmed
date
date_format YYYY MM DD
different fieldname
digits value
digits_between min max
boolean
email
exists table column
image
in foo bar
not_in foo bar
```

```
integer
numeric
ip
max value
min value
mimes jpeg png
regex  0 9
required
required_if field value
required_with foo bar
required_with_all foo bar
required_without foo bar
required_without_all foo bar
same field
size value
timezone
unique table column except idColumn
url
```

View

```
View  make 'path/to/view'
View  make 'foo/bar'   with 'key'  'value'
View  make 'foo/bar'   withKey 'value'
View  make 'foo/bar'  array 'key'    'value'
View  exists 'foo/bar'
// Share a value across all views
View  share 'key'  'value'
// Nesting views
View  make 'foo/bar'   nest 'name'  'foo/baz'  $data
// Register a view composer
View  composer 'viewname'  function $view
//Register multiple views to a composer
View  composer array 'view1'  'view2'   function $view
// Register a composer class
View  composer 'viewname'  'FooComposer'
View  creator 'viewname'  function $view
```

Blade

```
// Show a section in a template
@yield 'name'
@extends 'layout.name'
// Begin a section
@section 'name'
// End a section
@stop
// End a section and yield
@section 'sidebar'
@show
@parent

@include 'view.name'
@include 'view.name'  array 'key'    'value'
@lang 'messages.name'
@choice 'messages.name'  1

@if
@else
@elseif
@endif

@unless
@endunless

@for
@endfor

@foreach
@endforeach

@while
@endwhile

//forelse 4.2 feature
@forelse $users as $user
@empty
```

```
@endforelse

// Echo content
   $var
// Echo escaped content
    $var
// Echo unescaped content; 5.0 feature
    $var
     Blade Comment
// Echoing Data After Checking For Existence
    $name or 'Default'
// Displaying Raw Text With Curly Braces
    This will not be processed by Blade
```

Form

```
Form  open array 'url'    'foo/bar'  'method'    'PUT'
Form  open array 'route'    'foo.bar'
Form  open array 'route'    array 'foo.bar'  $parameter
Form  open array 'action'    'FooController@method'
Form  open array 'action'    array 'FooController@method'  $parameter
Form  open array 'url'    'foo/bar'  'files'    true
Form  close
Form  token
Form  model $foo  array 'route'    array 'foo.bar'  $foo  bar
```

Form Elements

```
Form  label 'id'  'Description'
Form  label 'id'  'Description'  array 'class'    'foo'
Form  text 'name'
Form  text 'name'  $value
Form  text 'name'  $value  array 'class'    'name'
Form  textarea 'name'
Form  textarea 'name'  $value
Form  textarea 'name'  $value  array 'class'    'name'
Form  hidden 'foo'  $value
Form  password 'password'
Form  password 'password'  array 'placeholder'    'Password'
```

```
Form  email 'name'  $value  array
Form  file 'name'  array 'class'    'name'
Form  checkbox 'name'  'value'
// Generating a checkbox that is checked
Form  checkbox 'name'  'value'  true  array 'class'    'name'
Form  radio 'name'  'value'
// Generating a radio input that is selected
Form  radio 'name'  'value'  true  array 'class'    'name'
Form  select 'name'  array 'key'    'value'
Form  select 'name'  array 'key'    'value'   'key'  array 'class'    'name'
Form  selectRange 'range'  1  10
Form  selectYear 'year'  2011  2015
Form  selectMonth 'month'
Form  submit 'Submit!'  array 'class'    'name'
Form  button 'name'  array 'class'    'name'
Form  macro 'fooField'  function

return '<input type="custom"/>'

Form  fooField
```

HTML

```
HTML  macro 'name'  function
// Convert an HTML string to entities
HTML  entities $value
// Convert entities to HTML characters
HTML  decode $value
// Generate a link to a JavaScript file
HTML  script $url  $attributes
// Generate a link to a CSS file
HTML  style $url  $attributes
// Generate an HTML image element
HTML  image $url  $alt  $attributes
// Generate a HTML link
HTML  link $url  'title'  $attributes  $secure
// Generate a HTTPS HTML link
HTML  secureLink $url  'title'  $attributes
// Generate a HTML link to an asset
```

```
HTML  linkAsset $url  'title'  $attributes  $secure
// Generate a HTTPS HTML link to an asset
HTML  linkSecureAsset $url  'title'  $attributes
// Generate a HTML link to a named route
HTML  linkRoute $name  'title'  $parameters  $attributes
// Generate a HTML link to a controller action
HTML  linkAction $action  'title'  $parameters  $attributes
// Generate a HTML link to an email address
HTML  mailto $email  'title'  $attributes
// Obfuscate an e-mail address to prevent spam-bots from sniffing it
HTML  email $email
// Generate an ordered list of items
HTML  ol $list  $attributes
// Generate an un-ordered list of items
HTML  ul $list  $attributes
// Create a listing HTML element
HTML  listing $type  $list  $attributes
// Create the HTML for a listing element
HTML  listingElement $key  $type  $value
// Create the HTML for a nested listing attribute
HTML  nestedListing $key  $type  $value
// Build an HTML attribute string from an array
HTML  attributes $attributes
// Build a single attribute element
HTML  attributeElement $key  $value
// Obfuscate a string to prevent spam-bots from sniffing it
HTML  obfuscate $value
```

String

```
// Transliterate a UTF-8 value to ASCII
Str  ascii $value
Str  camel $value
Str  contains $haystack  $needle
Str  endsWith $haystack  $needles
// Cap a string with a single instance of a given value.
Str  finish $value  $cap
Str  is $pattern  $value
Str  length $value
```

```
Str  limit $value  $limit   100  $end   '...'
Str  lower $value
Str  words $value  $words   100  $end   '...'
Str  plural $value  $count   2
// Generate a more truly "random" alpha-numeric string.
Str  random $length   16
// Generate a "random" alpha-numeric string.
Str  quickRandom $length   16
Str  upper $value
Str  title $value
Str  singular $value
Str  slug $title  $separator   '-'
Str  snake $value  $delimiter   '_'
Str  startsWith $haystack  $needles
// Convert a value to studly caps case.
Str  studly $value
Str  macro $name  $macro
```

Helper

Arrays

```
// adds a given key / value pair to the array if the
// given key doesn't already exist in the array
array_add $array  'key'  'value'
// collapse an array of arrays into a single array
array_collapse $array
// Divide an array into two arrays. One with keys and the other with values
array_divide $array
// Flatten a multi-dimensional associative array with dots
array_dot $array
// Get all of the given array except for a specified array of items
array_except $array  array 'key'
// Return the first element in an array passing a given truth test
array_first $array  function $key  $value     $default
// Strips keys from the array
array_flatten $array
// Remove one or many array items from a given array using "dot" notation
array_forget $array  'foo'
// Dot notation
```

```
array_forget $array  'foo.bar'
// Get an item from an array using "dot" notation
array_get $array  'foo'  'default'
array_get $array  'foo.bar'  'default'
// Checks that a given item exists in an array using "dot" notation
array_has $array  'products.desk'
// Get a subset of the items from the given array
array_only $array  array 'key'
// Return array of key => values
array_pluck $array  'key'
// Return and remove 'key' from array
array_pull $array  'key'
// Set an array item to a given value using "dot" notation
array_set $array  'key'  'value'
// Dot notation
array_set $array  'key.subkey'  'value'
// Sorts the array by the results of the given Closure
array_sort $array  function
// Recursively sorts the array using the sort function
array_sort_recursive
// Filters the array using the given Closure
array_where
// First element of an array
head $array
// Last element of an array
last $array
```

Paths

```
// Fully qualified path to the app directory
app_path
// Get the path to the public folder
base_path
// Fully qualified path to the application configuration directory
config_path
// Fully qualified path to the application's database directory
database_path
// Gets the path to the versioned Elixir file:
elixir
```

```
// Fully qualified path to the public directory
public_path
// Get the path to the storage folder
storage_path
```

Strings

```
// Convert a value to camel case
camel_case $value
// Get the class "basename" of the given object / class
class_basename $class
// Escape a string
e '<html>'
// Determine if a given string starts with a given substring
starts_with 'Foo bar.'  'Foo'
// Determine if a given string ends with a given substring
ends_with 'Foo bar.'  'bar.'
// Convert a string to snake case
snake_case 'fooBar'
// Limits the number of characters in a string
str_limit
// Determine if a given string contains a given substring
str_contains 'Hello foo bar.'  'foo'
// Result: foo/bar/
str_finish 'foo/bar'  '/'
str_is 'foo*'  'foobar'
str_plural 'car'
str_random 25
str_singular 'cars'
str_slug "Laravel 5 Framework"  "-"
// Result: FooBar
studly_case 'foo_bar'
trans 'foo.bar'
trans_choice 'foo.bar'  $count
```

URLs and Links

```
action 'FooController@method'  $parameters
// HTML Link
asset 'img/photo.jpg'  $title  $attributes
```

```
// HTTPS link
secure_asset 'img/photo.jpg'  $title  $attributes
route $route  $parameters  $absolute   true
url 'path'  $parameters   array    $secure   null
```

Miscellaneous

```
// Authenticator instance (Auth)
auth    user
// Generates a redirect response to the user's previous location
back
// Hashes the given value using Bcrypt (Hash)
bcrypt 'my-secret-password'
// Creates a collection instance from the supplied items
collect  'taylor'  'abigail'
// Gets the value of a configuration variable
config 'app.timezone'  $default
// Generates an HTML hidden input field containing the value of the CSRF token
    csrf_field
// Retrieves the value of the current CSRF token
$token   csrf_token
// Dumps the given variable and ends execution of the script
dd $value
// Gets the value of an environment variable or returns a default value
$env   env 'APP_ENV'
$env   env 'APP_ENV'  'production'
// Dispatches the given event to its listeners:
event new UserRegistered $user
// Creates a model factory builder for a given class
$user   factory App\User  class   make
// Generates an HTML hidden input field containing the spoofed value of the form's HTTP verb
    method_field 'delete'
// Retrieves an old input value flashed into the session
$value   old 'value'
$value   old 'value'  'default'
// Returns an instance of the redirector to do redirects:
return redirect '/home'
// Returns the current request instance or obtains an input item
$value   request 'key'  $default   null
```

```
// Creates a response instance or obtains an instance of the response factory
return response 'Hello World'  200  $headers
// Used to get / set a session value
$value   session 'key'
$value   session    get 'key'
session    put 'key'  $value
// Will simply return the value it is given.
value function    return 'bar'
// Retrieves a view instance
return view 'auth.login'
// Returns the value it is given
$value   with new Foo   work
```

Collection

```
// Creating Collections
collect  1  2  3
// Simply returns the underlying array represented by the collection:
$collection  all
// Returns the average of all items in the collection:
$collection  avg
// Breaks the collection into multiple, smaller collections of a given size:
$collection  chunk 4
// Collapses a collection of arrays into a flat collection:
$collection  collapse
// Determines whether the collection contains a given item:
$collection  contains 'New York'
// Returns the total number of items in the collection:
$collection  count
// Iterates over the items in the collection and passes each item to a given callback:
$collection   $collection  each function  $item  $key
   // Creates a new collection consisting of every n-th element:
$collection  every 4
// Pass offset as the second argument:
$collection  every 4  1
// Returns all items in the collection except for those with the specified keys:
$collection  except  'price'  'discount'
// Filters the collection by a given callback:
$filtered   $collection  filter function  $item
```

```
    return $item   2
   // Returns the first element in the collection that passes a given truth test:
collect  1  2  3  4    first function  $key  $value
    return $value   2
   // Flattens a multi-dimensional collection into a single dimension:
$flattened   $collection  flatten
// Swaps the collection's keys with their corresponding values:
$flipped   $collection  flip
// Removes an item from the collection by its key:
$collection  forget 'name'
// Returns a new collection containing the items:
$chunk   $collection  forPage 2  3
// Returns the item at a given key. If the key does not exist, null is returned:
$value   $collection  get 'name'
// Groups the collection's items by a given key:
$grouped   $collection  groupBy 'account_id'
// Determines if a given key exists in the collection:
$collection  has 'email'
// Joins the items in a collection:
$collection  implode 'product'  ', '
// Removes any values that are not present in the given array or collection:
$intersect   $collection  intersect  'Desk'  'Chair'  'Bookcase'
// Returns true if the collection is empty:
collect      isEmpty
// Keys the collection by the given key:
$keyed   $collection  keyBy 'product_id'
// Pass a callback, which should return the value to key the collection by:
$keyed   $collection  keyBy function  $item
    return strtoupper $item 'product_id'
   // Returns all of the collection's keys:
$keys   $collection  keys
// Returns the last element in the collection:
$collection  last
// Iterates through the collection and passes each value to the given callback:
$multiplied   $collection  map function  $item  $key
    return $item   2
   // Return the maximum value of a given key:
```

```
$max   collect   'foo'    10    'foo'    20     max 'foo'
$max   collect  1  2  3  4  5    max
// Merges the given array into the collection:
$merged   $collection  merge  'price'    100  'discount'    false
// Return the minimum value of a given key:
$min   collect   'foo'    10    'foo'    20     min 'foo'
$min   collect  1  2  3  4  5    min
// Returns the items in the collection with the specified keys:
$filtered   $collection  only  'product_id'  'name'
// Retrieves all of the collection values for a given key:
$plucked   $collection  pluck 'name'
// Removes and returns the last item from the collection:
$collection  pop
// Adds an item to the beginning of the collection:
$collection  prepend 0
// Pass a second argument to set the key of the prepended item:
$collection  prepend 0  'zero'
// Removes and returns an item from the collection by its key:
$collection  pull 'name'
// Appends an item to the end of the collection:
$collection  push 5
// Sets the given key and value in the collection:
$collection  put 'price'  100
// Returns a random item from the collection:
$collection  random
// Pass an integer to random. If that integer is more than 1, a collection of items is
returned:
$random   $collection  random 3
// Reduces the collection to a single value:
$total   $collection  reduce function  $carry  $item
    return $carry   $item
   // Filters the collection using the given callback:
$filtered   $collection  reject function  $item
    return $item   2
   // Reverses the order of the collection's items:
$reversed   $collection  reverse
// Searches the collection for the given value and returns its key if found:
```

```
$collection  search 4
// Removes and returns the first item from the collection:
$collection  shift
// Randomly shuffles the items in the collection:
$shuffled   $collection  shuffle
// Returns a slice of the collection starting at the given index:
$slice   $collection  slice 4
// Sorts the collection:
$sorted   $collection  sort
// Sorts the collection by the given key:
$sorted   $collection  sortBy 'price'
// Removes and returns a slice of items starting at the specified index:
$chunk   $collection  splice 2
// Returns the sum of all items in the collection:
collect  1  2  3  4  5    sum
// Returns a new collection with the specified number of items:
$chunk   $collection  take 3
// Converts the collection into a plain PHP array:
$collection  toArray
// Converts the collection into JSON:
$collection  toJson
// Iterates over the collection:
$collection  transform function  $item  $key
    return $item   2
   // Returns all of the unique items in the collection:
$unique   $collection  unique
// Returns a new collection with the keys reset to consecutive integers:
$values   $collection  values
// Filters the collection by a given key / value pair:
$filtered   $collection  where 'price'  100
// Merges together the values of the given array with the values of the collection:
$zipped   $collection  zip  100  200
```