Syntactically Awesome StyleSheets

Sass can't be directly interpreted by your browser, so it must first be converted, or compiled, to CSS before the browser can directly understand it.

@import

@mixin

@extend

Operator: +, -, \*,

Tree class

NSM shows good performance when tree is updated rarely. It is tuned to be fast for getting related nodes. It'is ideally suited for building multi-depth menu or categories for shop.

Suppose that we have a model Category; a $node variable is an instance of that model and the node that we are manipulating. It can be a fresh model or one from database.

**Relationships**

Node has following relationships that are fully functional and can be eagerly loaded:

* Node belongs to parent
* Node has many children
* Node has many descendants

### Inserting nodes

Moving and inserting nodes includes several database queries, so **transaction is automatically started** when node is saved. It is safe to use global transaction if you work with several models.

Another important note is that **structural manipulations are deferred** until you hit save on model (some methods implicitly call save and return boolean result of the operation).

If model is successfully saved it doesn't mean that node was moved. If your application depends on whether the node has actually changed its position, use hasMoved method:

if ($node->save()) {

$moved = $node->hasMoved();

}

#### **Creating nodes**

When you simply creating a node, it will be appended to the end of the tree:

Category::create($attributes); // Saved as root

$node = new Category($attributes);

$node->save(); // Saved as root

In this case the node is considered a root which means that it doesn't have a parent.

#### **Making a root from existing node**

// #1 Implicit save

$node->saveAsRoot();

// #2 Explicit save

$node->makeRoot()->save();

The node will be appended to the end of the tree.

#### **Appending and prepending to the specified parent**

If you want to make node a child of other node, you can make it last or first child.

In following examples, *$parent* is some existing node.

There are few ways to append a node:

// #1 Using deferred insert

$node->appendToNode($parent)->save();

// #2 Using parent node

$parent->appendNode($node);

// #3 Using parent's children relationship

$parent->children()->create($attributes);

// #5 Using node's parent relationship

$node->parent()->associate($parent)->save();

// #6 Using the parent attribute

$node->parent\_id = $parent->id;

$node->save();

// #7 Using static method

Category::create($attributes, $parent);

And only a couple ways to prepend:

// #1

$node->prependToNode($parent)->save();

// #2

$parent->prependNode($node);

#### **Inserting before or after specified node**

You can make $node to be a neighbor of the $neighbor node using following methods:

*$neighbor* must exists, target node can be fresh. If target node exists, it will be moved to the new position and parent will be changed if it's required.

# Explicit save

$node->afterNode($neighbor)->save();

$node->beforeNode($neighbor)->save();

# Implicit save

$node->insertAfterNode($neighbor);

$node->insertBeforeNode($neighbor);

PHP strom

Con Emu

Collective\Html\HtmlServiceProvider::*class*,  
Illuminate\Html\MenuServiceProvider::*class*,  
Kalnoy\Cruddy\CKEditor\CKEditorServiceProvider::*class*,  
Kalnoy\Cruddy\CruddyServiceProvider::*class*,  
Cartalyst\Sentinel\Laravel\SentinelServiceProvider::*class*,  
Barryvdh\Debugbar\ServiceProvider::*class*,  
Morilog\Jalali\JalaliServiceProvider::*class*,  
GrahamCampbell\Exceptions\ExceptionsServiceProvider::*class*,  
Vinkla\Hashids\HashidsServiceProvider::*class*,