# Installation

npm install express-handlebars --save

# Init Setup(\*không được thiếu)

const express = require("express");

const handlebars = require("express-handlebars");

const app = express();

/\* Set View Engine \*/

app.engine(

    "hbs",

    handlebars({

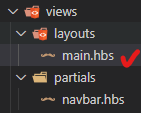
        extname: ".hbs",

        layoutsDir: \_\_dirname + "/views/layouts",

    })

);

app.set("view engine", "hbs");

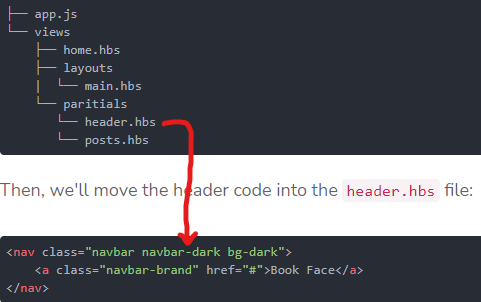
defaultLayout(string) : là ‘main.hbs’

partialsDir(string) : tên thư mục của mảnh default : ‘partials’

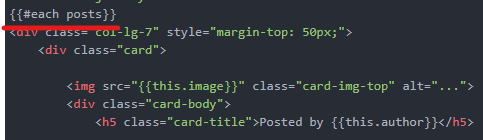
# Partial

{{> navbar}}

## Truyền content cho partial



Trong posts.hbs



Trong home.hbs(layout)

{{>header}}

{{>posts posts=posts}}

Hoặc khi đi kèm với #each

{{#each students}}

    {{> student-list student=.}}

{{/each}}

Trong partial

{{#with student}}

<tr>

    <td>{{id}}</td>

    <td>{{name}}</td>

    <td>{{age}}</td>

</tr>

{{/with}}

# Truyền params sang view

res.render("home", {

        content: "Đây là param truyền sang view",

});

ở view

<h1>{{content}}</h1>

# Xét điều kiện

Handlebar không cho ta cách để xét điều kiện trực tiếp trên view mà phải xét & filter từ server

res.render('shop', {

    prods: products,

    pageTitle: 'Shopping Page',

    path: '/',

    hasProducts: products.length > 0

  })

## Rendering

Don’t : 

Do : 

# Set Attribute

res.render('home', {

    post: {

        image: 'https://picsum.photos/500/500',

    }

});

Render :

<img src="{{post.image}}">

Block

Bắt đầu bằng {{# }} & kết thúc bằng {{/}}

# Path

var context = {

"occupation" : "developer",

"website" : {

"name" : "sitepoint"

}

"names" : [

{"firstName" : "Ritesh", "lastName" : "Kumar"},

{"firstName" : "John" , "lastName" : "Doe"}

]

}

Render

This article is available on {{website.name}}.<br/>

{{#each names}}

I am a {{../occupation}}. My name is {{firstName}} {{lastName}}.<br/>

{{/each}}

# Built-in Helpers

if, each, unless, and with

render

{{#helperName parameter1 parameter2 ...}}

Content here

{{/helperName}}

# Each

var context = {

"countries":["Russia","India","USA"],

"names" : [

{"firstName":"Ritesh","lastName":"Kumar"},

{"firstName":"John","lastName":"Doe"}

]

}

Có thể sử dụng {{this}} hoặc khỏi cần cũng được

{{#each countries}}

{{@index}} : {{this}}<br>

{{/each}}

{{#each names}}

Name : {{firstName}} {{lastName}}<br>

{{/each}}

## Nested loop

posts: [

    {

        author: 'Janith Kasun',

        image: 'https://picsum.photos/500/500',

        comments: [

            'This is the first comment',

            'This is the second comment',

            'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum nec fermentum ligula. Sed vitae erat lectus.'

        ]

    },

]

{{#each posts}}

    {{#if this.comments}}

    <ul class="list-group">

        {{#each this.comments}} {{!-- Nested --}}

        <li class="list-group-item">{{this}}</li>

        {{/each}}

    </ul>

    {{else}}

    <ul class="list-group">

        <li class="list-group-item">Be first to comment on this post</li>

    </ul>

    {{/if}}

{{/each}}

# If condition

{{#if countries}}

The countries are present.

{{else}}

The countries are not present.

{{/if}}

If false, undefined, null, "", 0, or [] are returned, the block won't be rendered.

nên trong 1 số trường hợp có thể sử dụng như này để check not render when empty

post: {

    comments: []

}

Render

{{#if post.comments}}

...

{{/if}}

# Build helper

## Function helper

Setup (for express-handlebar)

app.engine('hbs', exphbs({

    helpers: {

        getShortComment(comment) {

            if (comment.length < 64) {

                return comment;

            }

            return comment.substring(0, 61) + '...';

        }

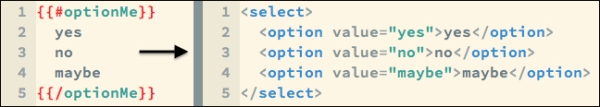
    }

}));

Render

{{getShortComment this}}

## Block Helper



Handlebars.registerHelper("optionMe", function(options){

var list = options.fn();

list = list.trim().split("\n");

var out = "<select>";

for (var i in list) {

var item = list[i].trim();

out += "<option value='" + item + "'>"

+ item

+ "</option>";

}

out += "</select>";

return out;

});

options.fn is already being escaped automatically

and if it escapes the outer helper as well, your content will be double-escaped.

Àaaaa ! Bố mày hiểu thằng options.fn làm gì rồi nhá

Để cho dễ hiểu hơn thì xem cái đoạn này

Handlebars.registerHelper("each", function(context, options) {

var ret = "";

for (var i = 0, j = context.length; i < j; i++) {

ret = ret + options.fn(context[i]);

}

return ret;

});

Tại sao phải đưa context vào trong options.fn ?

Thứ 1, context là data param mà ta truyền vào helper

Ví dụ : {{#each comments}}

Thì thằng options.fn thực hiện chức năng parsing data sang template

Đây là template mình có

{{#each comments}

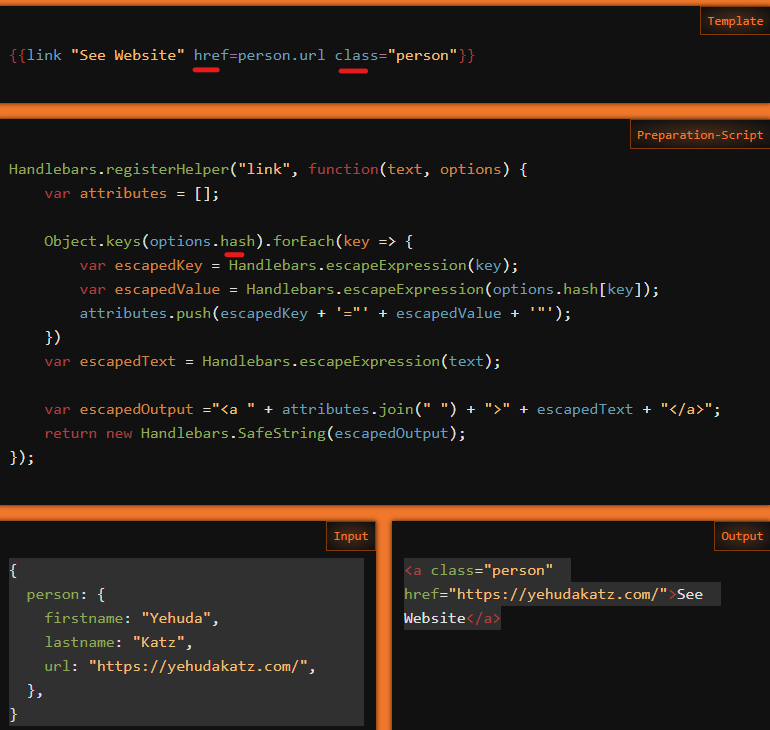
<h2>{{subject}}</h2>

{{/each}}

Options.fn sẽ parsing data sang {{subject}} rồi sau đó trả về toàn bộ template : <h2>{{subject}}</h2> đã escaped và render

Nhưng quan trọng là ~ gì được render ra DOM phải được return từ helper ở cuối function

## Helpers with Hash arguments



# Handlebar Helper Library

Install :

$ npm install --save handlebars-helpers

Import

var helpers = require('handlebars-helpers');

var math = helpers.math();

//=> only the `math` helpers

var array = helpers.array();

//=> only the `collections` helpers

var helpers = require('handlebars-helpers')(['math', 'string']);

//=> only the `math` and `string` helpers

Then config

var handlebars = require('handlebars');

var helpers = require('handlebars-helpers')({

  handlebars: handlebars

});

// or for a specific collection

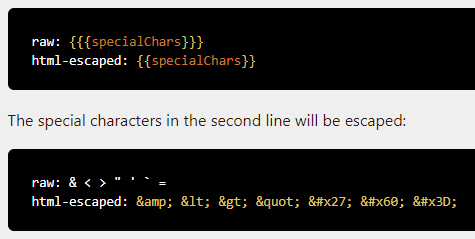
var math = helpers.math({

  handlebars: handlebars

});

# Advanced

## HTML Escaping



## Tránh nhầm lẫn data & helper render



## White-space control

Xét xem có render khoảng trắng không

This 

Or this 