Full Stack Web Development with Julia and Genie

JuliaCon 2017

Adrian Salceanu

June 23rd 2017

Web developer for 15 years

Developing real-time, data-intensive web products since 2008

CTO at OLBG, one of the biggest sportsbook affiliate in the UK (loooots of real-time, sports betting-related data!)

Creator of Genie.jl

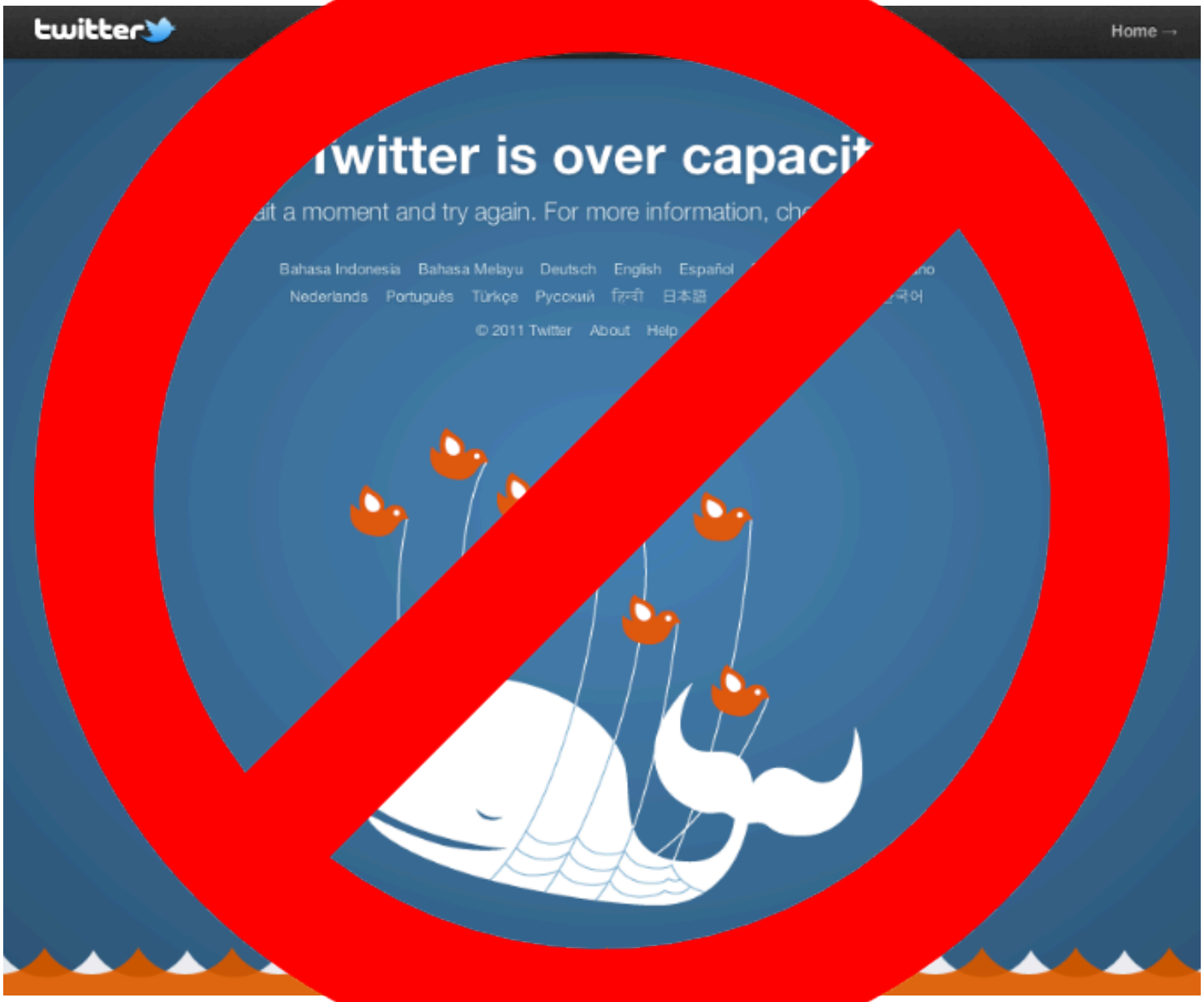
genieframework.com

Genie is a server-side web application framework, written in Julia.

Model-View-Controller architecture

MIT Licensed

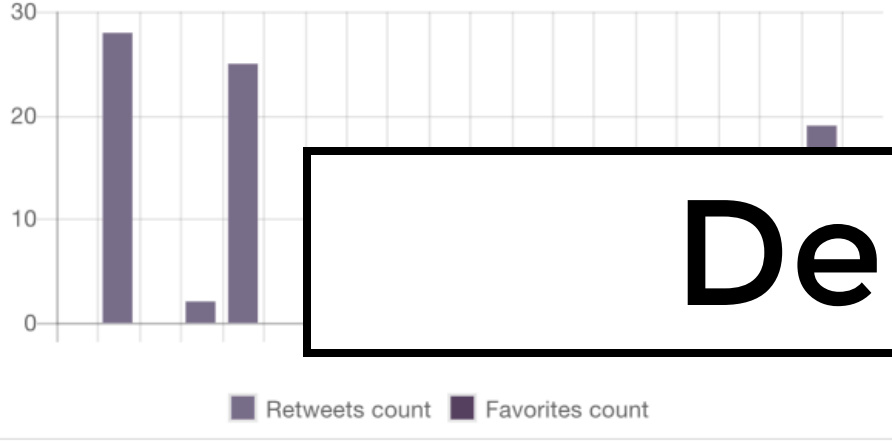
"Like Django and Rails, but without the fails..."

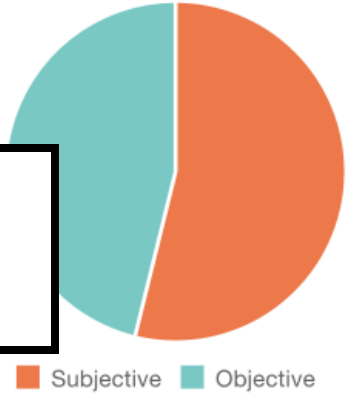

twitter
Home →
Twitter is over capacit
ait a moment and try again. For more information, ch
Bahasa Indonesia Bahasa Melayu Deutsch English Español
Nederlands Português Türkçe Русский हिन्दी 日本語
© 2011 Twitter About Help

Tweet Stats

python

Or try a recent search: C# C++ JULIALANG RUBY SWIFT AI PYTHON JAVASCRIPT





Demo

ʊ_ʊ @emptyHighways #TIL sometimes the ugliest code runs faster the a beautiful & readable code #pandas #python #numpy Positive Subjective

0 0

Moustafa Banbouk @KnowledgeRep RT @kdnuggets: #ICYMI 7 Steps to Mastering Data Preparation with #Python https://t.co/uaYzzpeWO1 https://t.co/u3FNZePaqv Neutral Objective 28 0

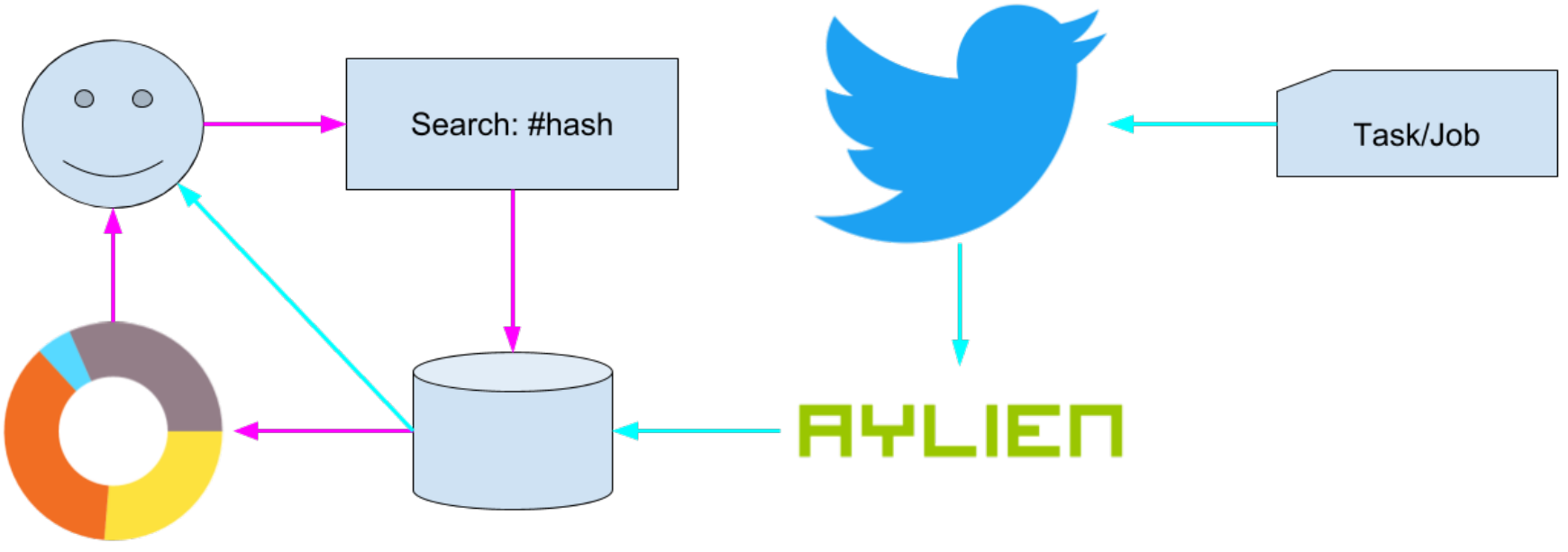
Software4iot @software4iot New Digital Home Solution offered by Comcast.. #javascript #Python https://t.co/fIVqLjSCFA Neutral Objective

Demo requirements

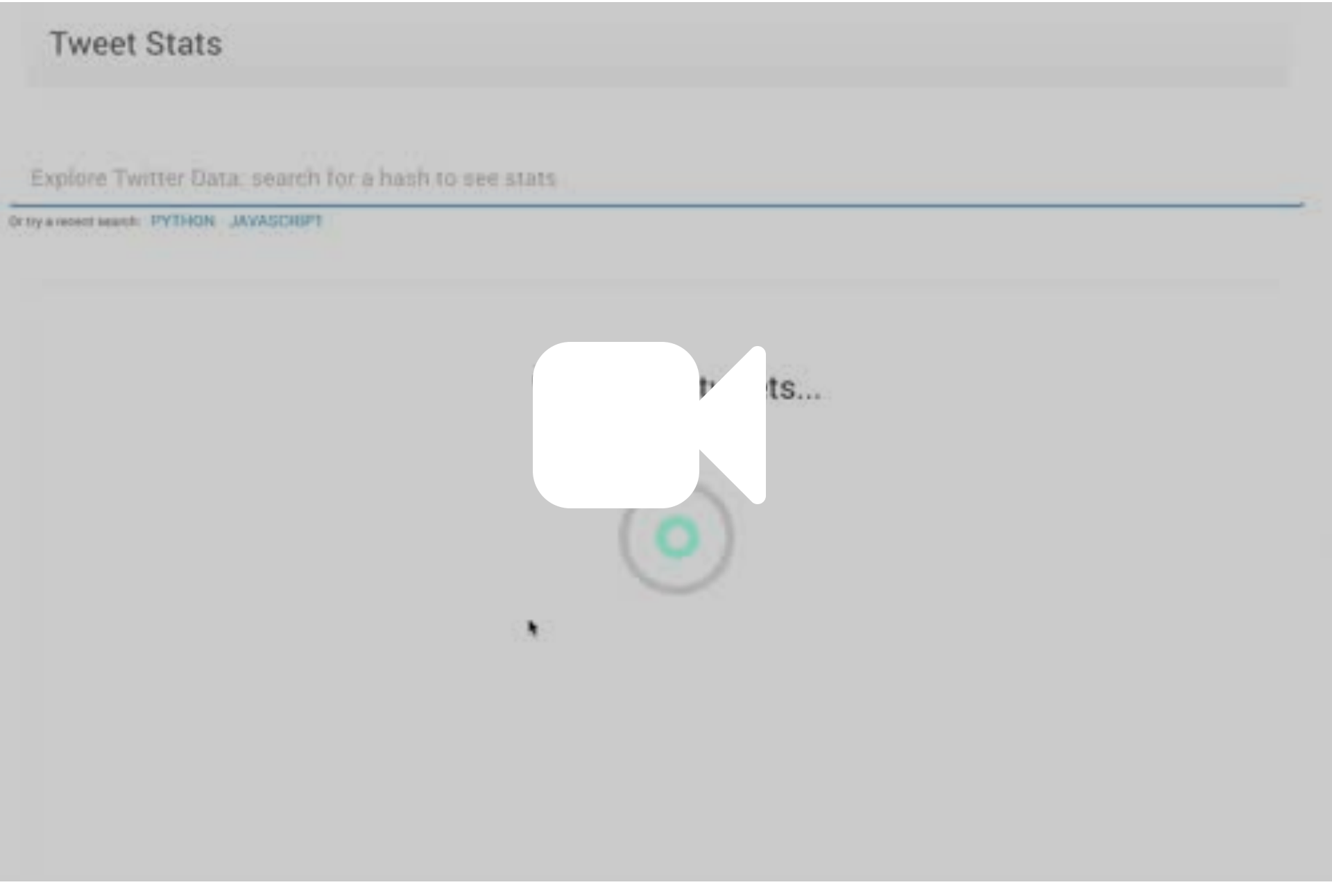
1. user searches for a Twitter #hash
2. app performs a #hash search against the Twitter API
3. app performs sentiment analysis on the tweets using the Aylien API (polarity, subjectivity)

Demo requirements

4. app computes some stats and displays some charts (retweets, favorites, polarity, subjectivity)
5. keeps updating in real time, as new tweets are created.



1. Synchronous request/response cycle
2. WebSockets request/response cycle

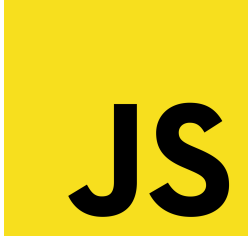

Tweet Stats
Explore Twitter Data: search for a hash to see stats
Or try a recent search: PYTHON JAVASCRIPT
ts...

100% 

backend

200 lines of 

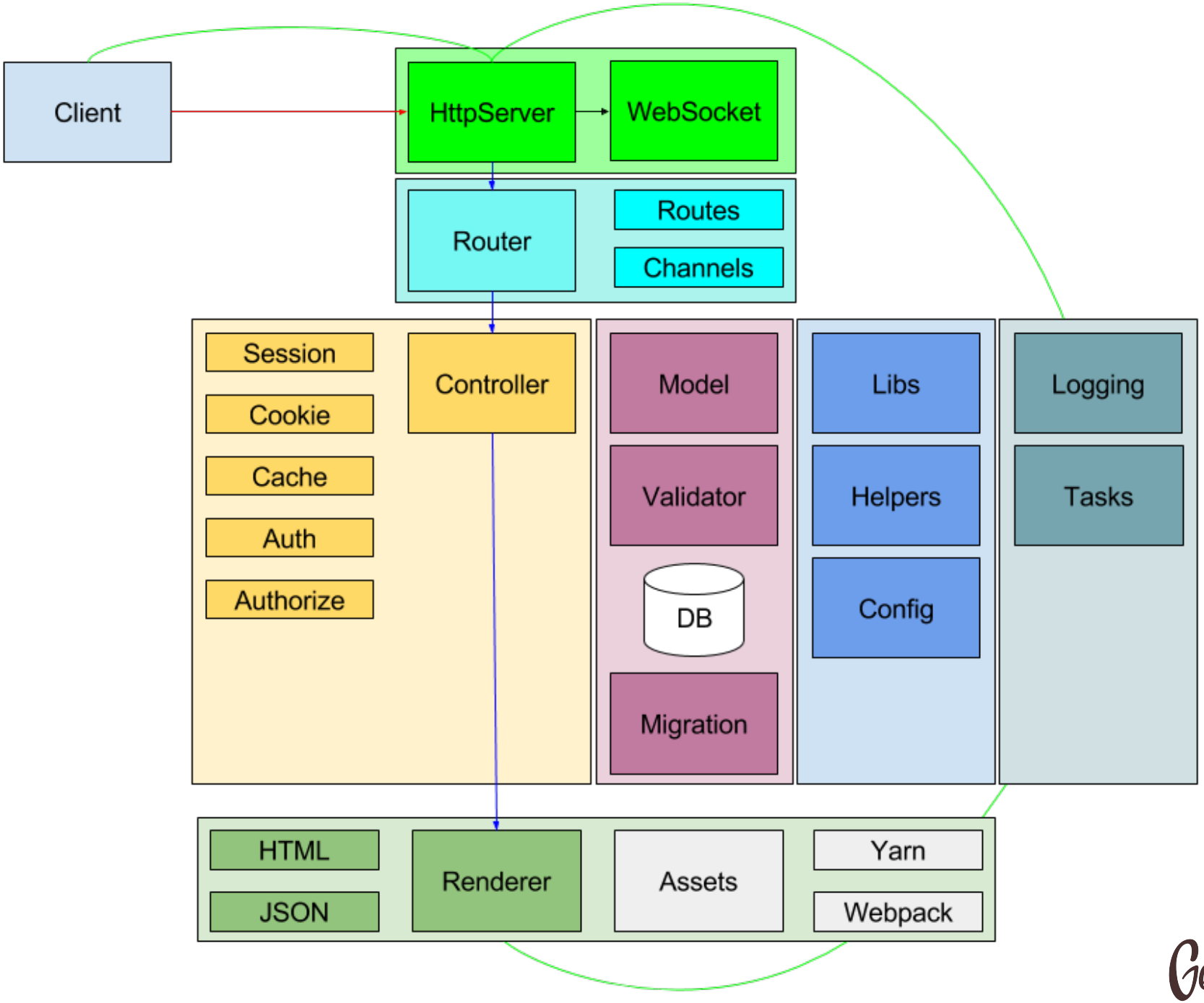
150 lines of 

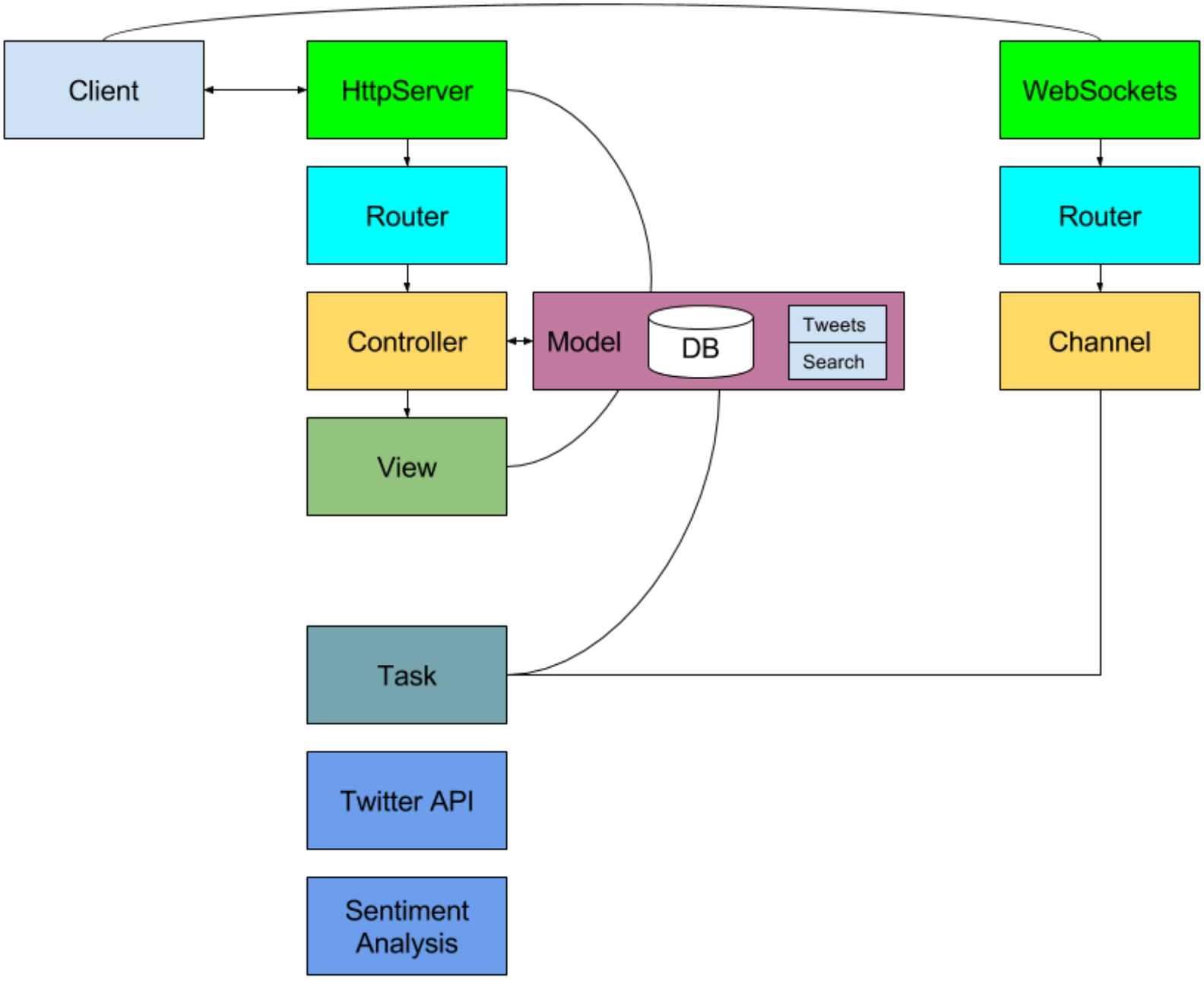
100 lines of 

Less than

2

hours

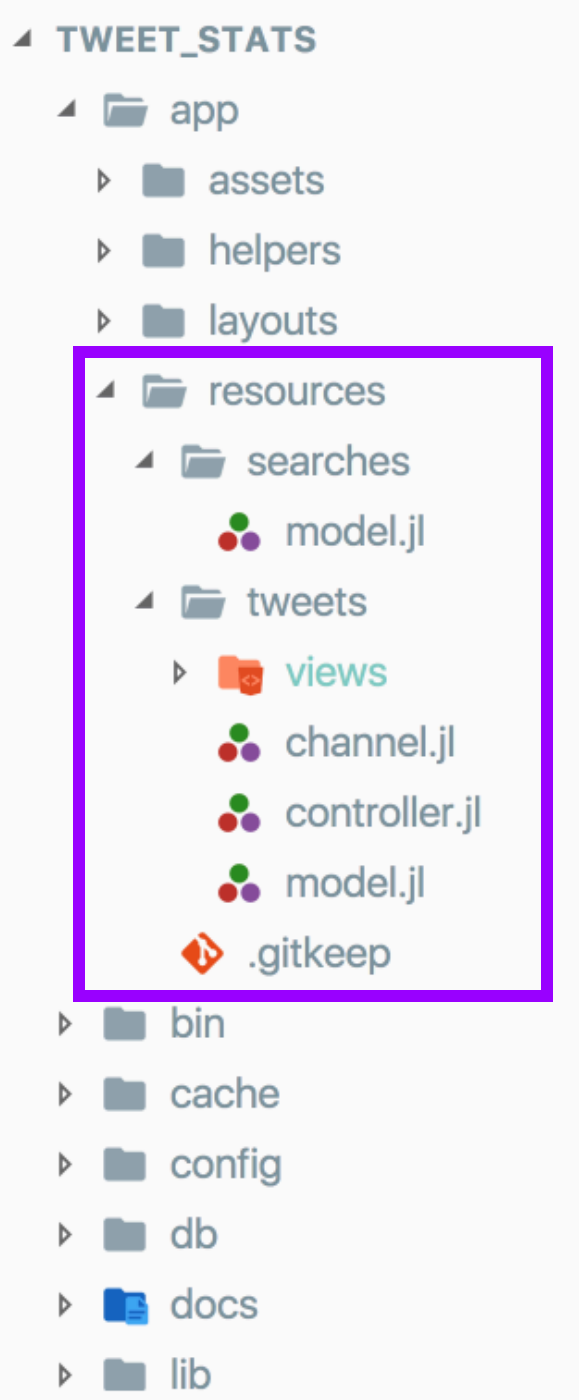

Client
HttpServer
WebSocket
Router
Routes
Channels
Session
Cookie
Cache
Auth
Authorize
Controller
Model
Validator
DB
Migration
Libs
Helpers
Config
Logging
Tasks
HTML
JSON
Renderer
Assets
Yarn
Webpack


Client
HttpServer
WebSockets
Router
Router
Controller
Model
DB
Tweets
Search
Channel
View
Task
Twitter API
Sentiment Analysis

Web Resources

Representational State Transfer (REST) approach

Genie apps provide access to textual (HTML, JSON, XML) representations of web resources



File structure of "searches" and "tweets" resources

Controller

app/resources/tweets/controller.jl

```
module TweetsController

using App, SearchLight
@dependencies

function index()
  search_term = haskey(@params, :q) ? lowercase(@params(:q)) : ""

  recent_searches = SearchLight.find(Search,
                                     SQLQuery(order = "id DESC",
                                              limit = 10))

  # ... more code here ...

  respond_with_html(:tweets, :index,
                    search_term = search_term,
                    recent_searches = recent_searches)
end

end
```

Searchlight

Model layer

ORM design pattern

DB query interface

Convention over configuration

app/resources/searches/model.jl

```
export Search, Searches

type Search <: AbstractModel
  ### internals
  _table_name::String
  _id::String

  ### fields
  id::Nullable{SearchLight.DbId}
  search::String

  ### constructor
  Search(;
    id = Nullable{SearchLight.DbId}(),
    search = ""
  ) = new("searches", "id", id, search)
end

module Searches
end
```

app/resources/tweets/controller.jl

```
module TweetsController

using App, SearchLight
@dependencies

function index()
  search_term = haskey(@params, :q) ? lowercase(@params(:q)) : ""

  recent_searches = SearchLight.find(Search,
                                     SQLQuery(order = "id DESC", limit = 10))

  # ... more code ...

  tweets = SearchLight.find(Tweet,
                            SQLQuery(where = ["polarity != -100",
                                              SQLWhereExpression("text ILIKE ?",
                                                                 "%#$(search_term)%")],
                                     limit = 20,
                                     order = "id DESC"))

  search = Search(search = search_term)

  SearchLight.create_or_update_by!!(search, :search)

  # ... more code ...

end

end
```

app/resources/tweets/model.jl

```
sql = "
    SELECT
      COUNT(*) AS results_count
    FROM
      (
      SELECT
        to_tsvector('english', readme) AS repo_info
      FROM
        repos
      ) repos_search,
      to_tsquery('$search_term') query
    WHERE
      repos_search.repo_info @@ query
  "
  SearchLight.query(sql)[1, :results_count]


SearchLight.query_raw("
    SELECT
      SUM((polarity = -1)::int) AS negative,
      SUM((polarity = 0)::int) AS neutral,
      SUM((polarity = 1)::int) AS positive,
      SUM((subjective = TRUE)::int) AS subjective,
      SUM((subjective = FALSE)::int) AS objective
    FROM tweets
    WHERE $(filters[1]) $(filters[2] |> string )") |> first
```

```
export DbId, SQLType, AbstractModel, ModelValidator

export SQLInput, SQLColumn, SQLColumns, SQLLogicOperator

export SQLWhere, SQLWhereExpression, SQLWhereEntity

export SQLLimit, SQLOrder, SQLQuery, SQLRaw

export SQLRelation, SQLRelationData

export SQLJoin, SQLOn, SQLJoinType, SQLHaving, SQLScope
```

SearchLight/src/SearchLight.jl

```
# Data retrieval
find(), find_df(), find_by()
find_one_by(), find_one_by!!(), find_one(), find_one!!()
rand(), rand_one(), rand_one!!(), all()

# Data persistance
save(), save!(), save!!()
update_with!(), update_with!!()
create_with()
update_by_or_create!!()
find_one_by_or_create()

# Data removal
delete(), delete_all()

# Low-level query
query(), query_raw()

# Data aggregation
count()
```

There's more

Relations

Validators

Callbacks

Scoping

app/resources/tweets/model.jl

```
using Channels

type Tweet <: AbstractModel
  ### internals
  _table_name::String
  _id::String

  ### fields
  id::Nullable{SearchLight.DbId}
  tweet_id::String
  text::String

  search_hash::String

  ### callbacks
  after_save::Function

  ### constructor
  Tweet(;
    id = Nullable{SearchLight.DbId}(),
    tweet_id = "",
    text = "",
    search_hash = "",
    after_save = (m::Tweet) -> Channels.broadcast(["tweets-" * m.search_hash],
                                                  "tweet",
                                                  SearchLight.to_dict(m))
  ) = new("tweets", "id", id, tweet_id, text, search_hash, after_save)
end
```

Database backends

Genie

Flax

HTML-based view templates

Uses embedded Julia

HTML to Julia transpiler

Compiled views

Fast!

app/resources/tweets/views/index.flax.html

```
<!-- list of tweets -->
<div class="row">
  <ul class="list-group" id="tweets_list" style="height: 600px; overflow: scroll;">

    <%
      foreachvar(:tweet, @vars(:tweets)) do (_)
        include_template("app/resources/tweets/views/partials/tweet.flax.html")
      end
    %>

    <li>
      <h4>
        <center>
          Waiting for tweets...
        </center>
      </h4>
    </li>
  </ul>
</div>
<!-- end list of tweets -->
```

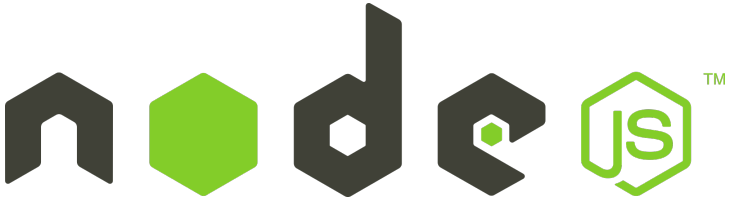
```
<li class="list-group-item" id="tweet_$( @vars(:tweet).id |> Base.get )">

  <div class="row">

    <div class="col-sm-1 col-xs-2">
      <img src="<$ @vars(:tweet).avatar_url $>" alt="user" class="img-circle" />
    </div>

    <div>
      <% @vars(:tweet).user_name %> @<% @vars(:tweet).screen_name %>
      <% @vars(:tweet).text %>
      <% sentiment_labels(@vars(:tweet).polarity) %>
      <% subjectivity_labels(@vars(:tweet).subjective) %>

      <span class="glyphicon glyphicon-retweet label label-default retweet-count">
        <span class="item-count">
          <% @vars(:tweet).retweet_count %>
        </span>
      </span>

      <span class="glyphicon glyphicon-heart label label-default favorite-count">
        <span class="item-count">
          <% @vars(:tweet).favorite_count %>
        </span>
      </span>

    </div>
  </div>
</li>
```

```
using Flax                                          app/resources/tweets/views/partials/tweet.flax.html
function func_2e03d893974211738ea4eebc1f4e63dba37d607a()
  Flax.skip_element()  do;[
    Flax.li(Symbol("class") => "list-group-item",
            Symbol("id") => "tweet_$( @vars(:tweet).id |> Base.get )" )  do;[
      Flax.div(Symbol("class") => "row" )  do;[
        Flax.div(Symbol("class") => "col-sm-1 col-xs-2" )  do;[
          Flax.img(Symbol("alt") => "user",
                   Symbol("class") => "img-circle",
                   Symbol("src") => "$(@vars(:tweet).avatar_url)" )
        ]end
        Flax.div()  do;[
          @vars(:tweet).user_name
          " @"
          @vars(:tweet).screen_name
          @vars(:tweet).text
          sentiment_labels(@vars(:tweet).polarity)
          subjectivity_labels(@vars(:tweet).subjective)
          Flax.span(Symbol("aria-hidden") => "true",
                    Symbol("class") => "glyphicon glyphicon-retweet label label-default retweet-c
          Flax.span(Symbol("class") => "item-count" )  do;[
            @vars(:tweet).retweet_count
          ]end
          ]end
          Flax.span(Symbol("aria-hidden") => "true",
                    Symbol("class") => "glyphicon glyphicon-heart label label-default favorite-co
            Flax.span(Symbol("class") => "item-count" )  do;[
              @vars(:tweet).favorite_count
            ]end
          ]end
        ]end
      ]end
    ]end
  ]end
```

app/resources/tweets/views/partials/tweet.flax.html

```
Flax.span(Symbol("class") => "item-count")  do;[
  @vars(:tweet).retweet_count
]end
```

Asset pipeline



Library installation

Dependency management

Transpiling

Concatenation

Minification

Fingerprinting (cache invalidation)


BABEL


ES6


CoffeeScript


Sass


jQuery

Router

config/routes.jl

```
using Router

route("/", resource = "tweets", controller = "TweetsController", action = "index")

route("/", "tweets#TweetsController.index")

# more examples

route("/hello") do
  "Hello World"
end

# "http://domain.com/packages/22" ==> @params(:packages_id) = "22"
route("/packages/:package_id",
      "packages#PackagesController.Website.show",
      named = :package)

# "http://pkg3.com/packages/22" ==> @params(:packages_id) = 22
route("/packages/:package_id::Int",
      "packages#PackagesController.Website.show",
      named = :package)

route(POST, "/admin/articles/create",
            "articles#AdminController.Website.article_create",
            named = :admin_article_create)
```

Tasks/Jobs

Background processes, potentially long-running, written in Julia, that execute in the context of the Genie app.

Can be started from the command line and are not exposed through the web server.

Environments

app/config/env/dev.jl

```
using Configuration
const config =
  Settings(
    output_length      = 100,
    suppress_output    = false,
    log_db             = true,
    log_queries        = true,
    log_requests       = true,
    log_responses      = true,
    log_router         = false,
    log_formatted      = true,
    log_level          = :debug,
    log_cache          = true,
    log_views          = true,
    assets_path        = "/",
    cache_duration     = 0,
    flax_compile_templates = false,
    websocket_server   = true,
    session_auto_start      = false
  )
```

app/config/env/prod.jl

```
using Configuration
const config =
  Settings(
    output_length        = 0,
    suppress_output      = false,
    log_db               = false,
    log_queries          = false,
    log_requests         = false,
    log_responses        = false,
    log_router           = false,
    log_formatted        = false,
    log_cache            = false,
    log_level            = "error",
    log_verbosity        =
        LOG_LEVEL_VERBOSITY_MINIMAL,
    assets_path          = "/",
    cache_duration       = 1_000
    flax_compile_templates = true,
    websocket_server     = true,
    session_auto_start   = false
  )
```

Caching

```
with_cache(hotel_data[:name], 600, dir = "hotels") do

  # This code is not executed if the cache is valid
  Requests.post(hotel_beds_uri,
                headers = hotel_beds_headers,
                json = hotel_beds_body) |> Requests.json

end
```

Database Migrations

```
module CreateTableSearches

using SearchLight

function up()
  SearchLight.query("CREATE SEQUENCE searches__seq_id")
  SearchLight.query("
    CREATE TABLE searches (
      id INTEGER CONSTRAINT searches__idx_id PRIMARY KEY DEFAULT NEXTVAL('searches
      search VARCHAR(50) UNIQUE
    )
  ")
  SearchLight.query("ALTER SEQUENCE searches__seq_id OWNED BY searches.id")
  SearchLight.query("CREATE INDEX searches__idx_search ON searches (search)")
end

function down()
  SearchLight.query("DROP TABLE searches")
end

end
```

Generators

App

Model

Controller

Channel

Resource

Migration

Task/Job

CLI

```
$> bin/server -p 8002

Listening on 0.0.0.0:8002...



$> bin/repl

genie>



$> bin/repl -h

# ... help output truncated ...
```

And more...

Authentication

Authorization

Sessions

Cookies

Logging

Database Seeding

Encryption

genieframework.com