Application Layer

SimpleApplication.java

include all followings and configuration

CreateReceipt Request API

Receipt Response API

Interface Layer

Create JSON/JavaParam {merchant,amount}

ReceiptRecords -> JSON / JavaParam {merchantName, amount, created, id}

ReceiptController   **TagController**

HTTP request & response - JAX-RS provides REST API
Enable receipt & tag manipulation on top of DAO (database operation) layer

@path("/ receipts")
@POST createReceipt:
CreateReceiptRequest JSON [from Request Body]-> Dao (insert)
@GET getReceipts:
Dao (get)->ReceiptRecords -> ReceiptResponse

@GET GetReceiptsTag

@path("/ receipts")
@PUT Response for tag/untag [receiptid from Request Body] + [tag_name from PathParam]
->Dao (check receiptid existence, insert or delete {tagid,receiptid} in tag receipts table)
@GET getReceiptsFromTag [tag_name from PathParam] + Dao (get) —>ReceiptResponse

Test.java (create receipt, get receipt by tag, tag/untag, etc)

@Test assertThat
gradle test

We don't have this layer in this simple application

Service Layer

Business logic (i.e., Receipts Managers, CRM, etc)

Java codes that cannot be hosted in controllers (like those that need to be applied cross- controllers)

Dao Layer

Receipt Dao   **Tag Dao**

Defined Insert/ Delete/ GetAll/ GetByID/ CheckExist functions enabled by jOOQ (all the database operations) for controllers to use.

Database Layer

H2 DB

TagRecords

ReceiptRecords

Schema.sql

- the objects in the formate of the return of queries

generated by jOOQ
gradle generate

Jooq - an internal domain specific language API (dsl) in Java will enable SQL-like java code to do database

```
Tags
PK id: int unassigned auto_increment
   tag: varchar(255)
   uploaded: time default current_time()
```

```
Tagreceipts
PK (receiptstid,tagid)
   tagid: int unassigned auto_increment
   receiptsid: int unassigned auto_increm
```

```
Receipts
PK id: int unassigned auto_increment
   merchant: varchar(255)
   amount: bigdecimal(12,2)
   uploaded: time default current_time()
```
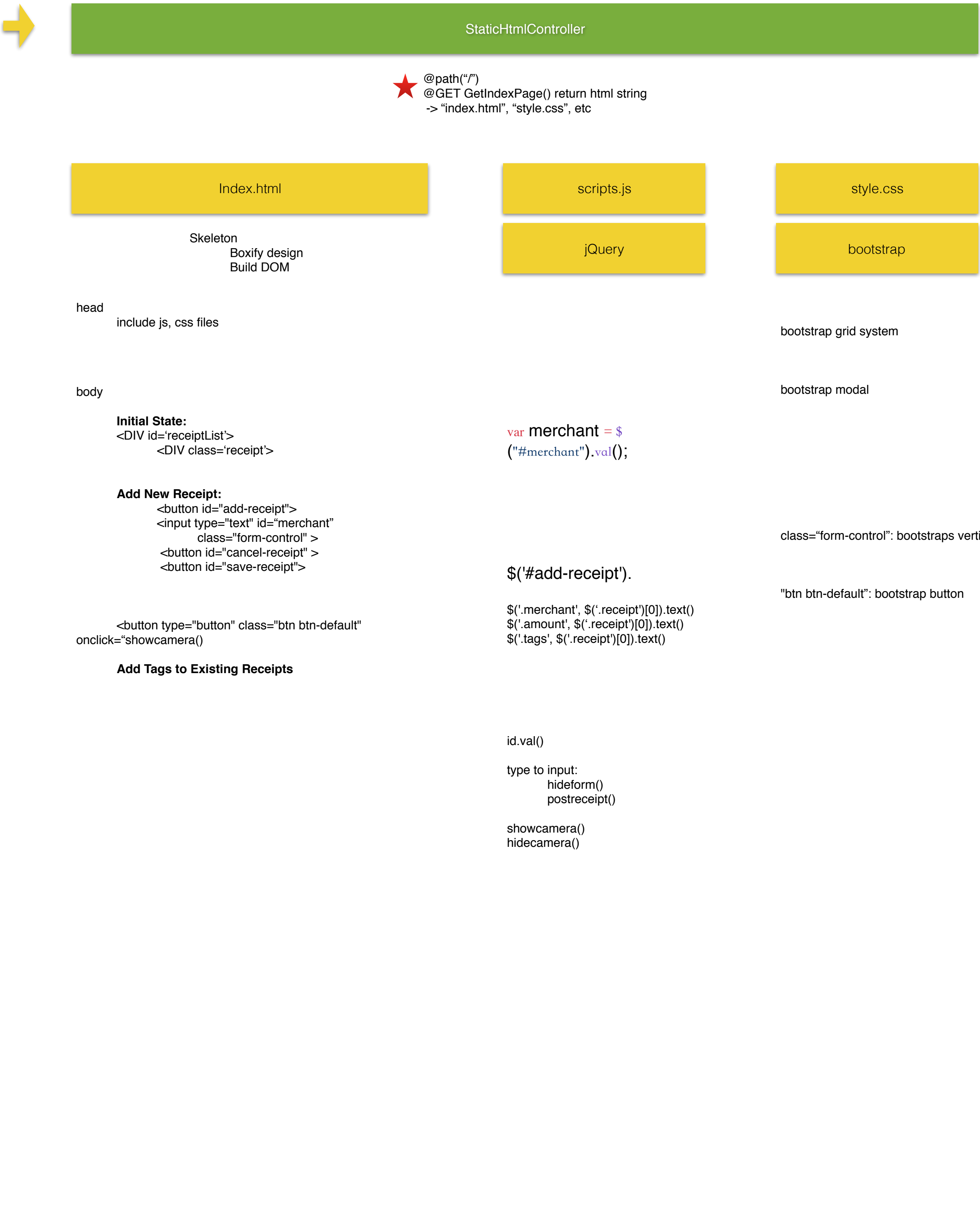
Notes:
We didn''t deal with session management and access control---any user can read / write any receipt.
We didn't set up an external database - we used H2 (which supports the MySQL SQL Dialect) locally.

gradle distTar will generate a compressed file

docker build will use dockerfile (docker command inside) to generate image

docker push to docker repo or AWS repo

AWS ECS (user/ usergroup/ IAM role, etc)

Registry
AWS configure: login ECS user and get login path url
Docker registry: docker log-in using AWS login path

Create ECS repo
Docker push images to ECS repo

Create ECS Cluster, Create ECS TaskDefinition (ECS container with the docker image pushed), Create service

Circle CI as testing tools

JSON

Java Param

Dao

jOOQ

StaticHtmlController

⭐ @path("/")
@GET GetIndexPage() return html string
-> "index.html", "style.css", etc

Index.html   **scripts.js**   **style.css**

        jQuery   **bootstrap**

Skeleton
    Boxify design
    Build DOM

head

    include js, css files   bootstrap grid system

body   bootstrap modal

    Initial State:
    <DIV id='receiptList'>   var merchant = $
        <DIV class='receipt'>   ("#merchant").val();

    Add New Receipt:
        <button id="add-receipt">
        <input type="text" id="merchant"   class="form-control": bootstraps vertical form
            class="form-control" >
        <button id="cancel-receipt">   "btn btn-default": bootstrap button
        <button id="save-receipt">

                        $('#add-receipt').

                        $('.merchant', $('.receipt')[0]).text()
    <button type="button" class="btn btn-default" $('.amount', $('.receipt')[0]).text()
onclick="showcamera()" $('.tags', $('.receipt')[0]).text()

    Add Tags to Existing Receipts

                        id.val()

                        type to input:
                            hideform()
                            postreceipt()

                        showcamera()
                        hidecamera()