# Diagram of the request processing

include all libraries

init data connection

forward

page draw

run views to build GUI

Business process

init $CONFIG variable

# engine/start.php

## Attributes of $CONFIG

* root: <string> Base URL of site (example: *http://yoursite.com/*). Get from the request URL.
* title: <string> Site title.
* user: BKITUser object of logged-in user

# engine/event.php

## Object definition:

* Class name: **BKITEvent**
* Inherits: None
* Interfaces: None

## Attributes:

* **eid**: <int>
* **name**: <string>
* **title**: <string>
* **headline**: <string> HTML of headline
* **content**: <string> HTML of full content
* **image**: <string> Full URL of image (this attributes is not inserted to database)
* **image\_name**: <string> filename of image. Image upload to ./upload/images/ folder
* **time\_created**: <int> Unix time
* **time\_updated**: <int> Unix time

## Methods:

* **get(name)**: Get value of attribute by name
  + Param:
    - **name**: <string> Name of attribute to get
  + Return: <mixed> Value of attribute name
* **set(name, value)**: Set value of attribute by name
  + Param:
    - **name**: <string> Name of attribute to set
    - **value**: <mixed> Value of attribute to set
  + Return: <boolean> Depend on success
* **save()**: Save this event to database
  + Process:
    - Check whether *eid* is exist in database.
    - Insert new row into database if *eid* did not exist in database.
    - Update into database if *eid* existed in database.
  + Return:
    - false: if error
    - int: eid of new row
* **delete()**: Delete this event in database
  + Return:
    - false: if error
    - int: eid of deleted row

## Other functions:

* **get\_event(eid)**: Get BKITEvent object by eid
  + Param:
    - **eid**: <int> eid of event to get
  + Return:
    - false: if error
    - BKITEvent: if succeed
* **get\_events(order\_by, limit, offset, min\_time\_created, max\_time\_created)**: Get list of BKITEvent object
  + Param:
    - **order\_by**: <string> Order list by. Default: “time\_created ASC”
    - **limit**: <int> Numbers of list. Default: 10
    - **offset**: <int> List start at. Default: 0
    - **min\_time\_created**: <int> Unix time, time\_created of BKITEvent must be greater than. Default: 0
    - **max\_time\_created**: <int> Unix time, time\_created of BKITEvent must be less than. Default: 0 (unlimited)
  + Return:
    - false: if error
    - Array of BKITEvent objects if success

# engine/user.php

## Object definition:

* Class name: **BKITUser**
* Inherits: None
* Interfaces: None

## Attributes:

* **uid**: <int>
* **lastname**: <string>
* **firstname**: <string>
* **birthday**: <int> Unix time
* **sex**: <string> “male” or “female”
* **courseyear**: <int>
* **address**: <string>
* **phone**: <string>
* **yahooid**: <string>
* **email**: <string>
* **time\_created**: <int> Unix time
* **time\_updated**: <int> Unix time
* **expectation**: <string> convert all HTML from user
* **hobby**: <string>
* **skills**: <array>, each item have a pairs of key and value:
  + Key: **sid** of skill
  + Value: Array of 3 values: **skillname**, **level** and **skillcategory**

## Methods:

* **get(name)**: Get value of attribute by name
  + Param:
    - **name**: <string> Name of attribute to get
  + Return: <mixed> Value of attribute name
* **set(name, value)**: Set value of attribute by name
  + Param:
    - **name**: <string> Name of attribute to set
    - **value**: <mixed> Value of attribute to set
  + Return: <boolean> Depend on success
* **approve()**: Approve if status is waiting
  + Process:
    - Check whether *status* is *waiting* or *deny*
    - Change *status* to *approve*
    - Send email
  + Return: <boolean> Depend on success
* **deny()**: Deny if status is waiting
  + Process:
    - Check whether *status* is *waiting* or *deny*
    - Change *status* to *approve*
    - Send email
  + Return: <boolean> Depend on success
* **save()**: Save this event to database
  + Process:
    - Check whether *uid* is exist in database.
    - Insert new row into database if *uid* did not exist in database.
    - Update into database if *uid* existed in database.
  + Return:
    - false: if error
    - int: uid of new row
* **delete()**: Delete this event in database
  + Return:
    - false: if error
    - int: uid of deleted row

## Other functions:

* **get\_user(uid)**: Get BKITUser by uid
  + param:
    - uid: <int>
  + Return:
    - false: if error
    - BKITUser: if succeed
* **get\_users(status, order\_by, limit, offset, min\_time\_created, max\_time\_created)**: Get list of BKITUser object
  + Param:
    - **status**: <string> Status of list user. Default: “all” (all status)
    - **order\_by**: <string> Order list by. Default: “time\_created ASC”
    - **limit**: <int> Numbers of list. Default: 10
    - **offset**: <int> List start at. Default: 0
    - **min\_time\_created**: <int> Unix time, time\_created of BKITUser must be greater than. Default: 0
    - **max\_time\_created**: <int> Unix time, time\_created of BKITUser must be less than. Default: 0 (unlimited)
  + Return:
    - false: if error
    - Array of BKITUser objects if success
* **get\_user\_login()**: Get BKITUser object that currently login
  + Return:
    - false: if user do not login
    - BKITUser of current user if succeed
* **is\_user\_login()**: Check whether user is login (admin or normal user are returned the same)
  + Return: <boolean> Depend on success
* **is\_admin\_login()**: Check whether user is admin and login
  + Return: <boolean> Depend on success
* **login(email, password)**: Login with email and password
  + Param:
    - **email**: <string>
    - **password**: <string> Password was hashed with MD5
  + Process:
    - Check syntax of email and password
    - Check if user can login
    - Set cookie with email and password
    - Set current user into **$CONFIG**
  + Return:
    - false: if error or wrong
    - BKITUser of logged-in user if succeed
* **logout()**: Logout
  + Process:
    - Check if user is login
    - Clear cookie for email and password
    - Remove current user from **$CONFIG**

# engine/db.php

## Functions:

* **get\_data(query, datatype)**: Run query and fetch data into object of datatype
  + Param:
    - **query**: <string> Query string (MySQL)
    - **datatype**: <string> Type of returned data, it can be “user”, “event”.
  + Process:
    - Call *db\_query* to run *query* and get the returned resource
    - Fetch resource, for each item: Create new object (by using *get\_dataclass*), and set value.
  + Return:
    - false: if error
    - array of mixed value (means BKITUser or BKITEvent, depend on datatype)
* **db\_query(query)**: Run query like mysql\_query()
  + Param:
    - **query**: <string> Query string (MySQL)
  + Process:
    - Clear *query* to reject all injection
    - Call mysql\_query()
  + Return:
    - false: if error
    - resource of MySQL if succeed
* **get\_dataclass(datatype)**: Create object of datatype
  + Param:
    - **datatype**: <string> Can be “user”, “event”
  + Process:
    - Create new object by
      * “user”: BKITUser
      * “event”: BKITEvent
  + Return:
    - false: if error
    - mixed: object if succeed
* **start\_connection()**: Start connection to MySQL server
  + Process:
    - Call *get\_connection\_config()*.
    - Open connection to MySQL server (by using *mysql\_connection*)
    - Save connection link to **$DB**
  + Return: <boolean> Depend on success
* **get\_connection\_config()**: Get site config to connect MySQL server.
  + Process:
    - Read config from engine/config.php
    - Insert into array to return.
  + Return:
    - <Array>: pair of key and value. Keys are: “host”, “username”, “password”, “database\_name”, “table\_prefix”,...
* **close\_connection()**: Close connection
  + Process:
    - Get connection link from **$DB**, and close it

# engine/lib.php

## Functions:

* **page\_draw(title, content)**: Draw completed page, output to client side
  + Param:
    - **title**: <string> Title of page
    - **content**: <string> HTML string of content
  + Process:
    - Load header, sidebar, footer, and all other component from theme (using *view\_theme()* function)
    - Arrange the components.
    - Echo HTML string to client.
    - Call shutdown().
  + Return: null
* **shutdown()**: Save **$CONFIG** and shutdown request
  + Process:
    - Save all **$CONFIG** values (if needed)
    - Disconnect to database (by using *close\_connection()* function)
  + Return: null
* **view\_theme(viewname, vars)**: View component of theme
  + Param:
    - **viewname**: <string> View name of this component, is name of a file in folder *theme/default/*. It can be ‘header’, ‘sidebar’, ‘content’, ‘footer’
    - **vars**: <string> String to insert.
  + Return: <string> HTML of this component.
* **view(viewname, vars)**: View something
  + Param:
    - **viewname**: <string> View name, is name of a file in folder *views/*
    - **vars**: <array> All variables needed to process. Each item is pair of key and value
      * “object”: <mixed> Object to view (BKITUser, BKITEvent,...)
      * “viewtype”: <string> Type to view. Can be: “full”, “short”, “one-line”
      * ...
  + Return:
    - String HTML.
* **forward(url)**: Forward to url
  + Param:
    - **url**: <string> URL to forward. Default: null (means hompage)
  + Process:
    - If *url* has “http://” or “https://”, forward to url correctly
    - If not, forward to “http://yoursite.com/<url>”.
  + Return: null

# engine/theme.php

## Functions:

* **get\_theme ()**: Get name of current theme to view
  + Process:
    - Check whether user set his (her) theme or not
    - If not, check whether admin set default theme or not
    - If not, return “default”
  + Return: <string> name of theme
* **set\_theme (name, uid)**: Set name of theme for user
  + Param:
    - **name**: <string> name of theme. Default: “default”
    - **uid**: <int> uid of user to set his (her) theme. Default: 0 (means default theme of site, only set by admin)
  + Return: <boolean> Depend on success
* **choose\_theme(name)**: Choose theme to view
  + Param:
    - **name** : <string> name of theme. Default: “default”
  + Process:
    - Set to **$CONFIG[theme]**
  + Return: <boolean> Depend on success
* **view\_theme(position, vars)**: View HTML of position
  + Param:
    - **position**: <string> Name of position to view. Values are: “header”, “footer”, “sidebar”, “content”,...
    - **vars**: <array> All variables needed to process (like *view()* function)
  + Process:
    - include file from **“$CONFIG[theme]**” folder, and the file is “**position**”.php
  + Return: <string> HTML to view this position

# engine/mail.php

## Object definition:

* Class name: **BKITMail**
* Inherits: None
* Interfaces: None

## Attributes:

* from\_address: <string> Email of sender
* from\_name: <string> Name of sender
* to\_address: <string> Email of receiver
* to\_name: <string> Name of receiver
* mid: <int> mid of template mail in database
* template: <string> Template of email, has some replacer to insert some value
* replacement: <array> pair of key and value:
  + key: replacer, such as *%User\_name%*
  + value: replacement string, such as *Kim Kha*

## Methods:

* send(): Send email
  + Return: <boolean> Depend on success
* load\_template(mid): Load template from database
  + Param:
    - mid: <int> mid of mail template
  + Return: <boolean> Depend on success
* save\_template(): Save template to database
  + Return: <boolean> Depend on success
* load\_template\_from\_file(template\_file): Load template from text file
  + Param:
    - template\_file: <string> Path to template file
  + Return: <boolean> Depend on success
* save\_template\_to\_file(template\_file): Save current template to text file
  + Param:
    - template\_file: <string> Path to template file
  + Return: < boolean> Depend on success
* get\_mailcontent(): Get content of mail (ready to send)
  + Return:
    - false: if error
    - HTML String: Content of mail after replacing all keys as all values.