Tässä dokumentissa selitetään MVC malli ja sen osat. Esimerkkinä käytetään Käyttäjä contenttia, content/account ja kirjautumistoimintoja content/login sovelmassa.

MVC malli

MVC malli eriyttää kolmeen eri osa-alueeseen html koodin luomisen. Koska MVC mallissa on perustoimintoja, tarjoaa jokainen framework aina perusluokat, joista ohjelmoija periyttää omat MVC tiedostonsa. Siis sen sijaan, että käytettäisiin framework tiedostoja SomeController, SomeModel ja SomeView suoraan, käytettään esimerkiksi luokkia:

SomeControllerDefault extends SomeController
SomeModelRegister extends SomeModel
SomeViewRegister extends SomeView

Neljäs MVC mallin toiminto on SomeView olion ominaisuus. Tämä olio käyttää html koodin luomiseksi template -tiedostoja. Template tiedostot löytyvät tmpl/ hakemistoista, hakemistoja on yksi jokaista näkymää, "view", kohden.

Käyttäjän osalta keskeinen toiminto on rekisteröityminen ja siihen liittyen 'register' näkymä. Tätä käytetään esimerkkinä tässä dokumentissa.

Nimeämiskäytännöt Pähkinänkuoressa:

Malli, model.

Malli nimeltä foo on tiedostossa
model/foo.php
jossa luokan nimi on
SomeModelFoo

Näkymä, view

Näkymä nimeltä bar on tiedostossa
/view/bar/bar.php
jossa luokan nimi on
SomeViewBar
Jonka template on default, jos muuta ei anneta, ja on tiedostossa
/view/bar/tmpl/default.php

Controllerilla ei ole nimeämiskäytäntöä, mutta ylläolevat malli ja näkymä luodaan koodilla:

```
$model = $this->getModel( 'foo' );
$view = $this->getView( 'bar' );
```

ja jos halutaan vaihtaa template default templateksi foobar:

```
$view->display('foobar');
```

jolloin template on

view/bar/tmpl/foobar.php

Nimeämiskäytännöissä on riippuvuuksia tiedoston nimen, ja luokan nimen päätteen välillä. Siis mallien SomeModel ja näkymien SomeView sidonnaisuus on register näkymän osalta seuraava:

SomeModelRegister on tiedostossa, jonka nimi on register.php.

Koska tämä on SomeModel luokka, se on kansiossa model/. Kansion model nimeä ei saa muuttaa eikä malleja saa laittaa muihin kansioihin. Siellä olevan tiedoston nimi voi olla mikä vain: default.php, login.php, logout.php ja register.php, mutta luokan nimen on oltava vastaavasti SomeModelDefault, SomeModelLogin, SomeModelLogout ja SomeModel Register. Tiedoston nimi vastaa luokan nimen päätettä SomeModel -osan jälkeen. Tämä käytäntö tarkoittaa siis, että tiedostossa model/default.php ei voi olla luokkaa SomeModelLogin (vaan luokka SomeModelDefault).

Syy tähän nimeämiskäytäntöön on se, että SomeController luokan getModel($name) yrittää etsiä

$name argumenttinsa mukaan luokan ja luoda siitä instanssin seuraavien sääntöjen mukaan:
On oltava tiedosto model/{$name}.php, jossa on oltava luokka nimelta SomeModel{$name}. Siis:
SomeController->getModel('register') olettaa, että on olemassa tiedosto:

model/register.php

jossa on luokkamäärittely:

class SomeModelRegister extends SomeModel.

Toinen nimeämiskäytäntö, joka on vastaava, mutta vaihtuu ainoastaan yhden hakemiston mukaan ottamisen myötä, on SomeView olion luonti ja siihen liittyvä nimeämiskäytäntö. SomeController luokan getView($name) yrittää etsiä näkymää kansiosta view/$name/. Tuossa kansiossa on oltava tiedosto nimeltä $name.php, siis tiedosto view/{$name}/{$name}.php. Kansion nimeä 'view' ei voi vaihtaa ja kaikki näkymät on laitettava omaan alikansioon.

Esimerkiksi SomeController- >getView('register') olettaa, että on olemassa tiedosto:

view/register/register.php

jossa on luokka SomeView{$name}, eli luokkamäärittely:

class SomeViewRegister extends SomeView

SomeController luokalla ei ole tiedoston nimen ja sen sisältävän luokan mukaan nimeämiskäytäntöjä, silla content bootstrap luo controllerin niin, että ohjelmoija kirjoittaa suoraan tiedoston nimen ja luokan nimen koodissa. Ne voi vapaasti valita.

Kertauksena: Rekisteröinti mallin ja näkymän luominen user content toteutukseen sisältää siis seuraavat tiedostot silloin, kuin oletetaan, että käytetään apumetodeja getModel(string $name) ja getView(string $name) metodien argumenttina arvoa "register":

content/user/model/register.php – luokka SomeModelRegister extends SomeModel
content/user/view/register/register.php – luokka SomeViewRegister extends SomeView

Controllerin suhteen ei ole nimeämiskäytäntöä, mutta controller voi olla esimerkiksi tiedostossa ja luokkana:
content/user/controller/default.php – class SomeContollerFoo extends SomeController.

Controller execution flow.

Controllerilla on 4 tehtävää. Lisäksi se voi kutsua joitakin mallin SomeModel metodeita. Näiden metodien kutsuminen riippuu täysin ohjelmoijan valinnoista, ja miten hän haluaa malliaan käyttää.

Tehtävä 1. Controller luo mallin.

Koska Malli luodaan joka kerta, sen luomiseen on annettu valmis metodi getModel(string), jonka ainoata argumenttia koskee yllä esitelty nimeämiskäytäntö.
Ohjelmoija voi koska tahansa ohittaa getModel() metodin kutsun. Jos SomeController->getModel('register') luo olion SomeModelRegister tiedostosta model/register.php, on seuraava koodi vaihtoehtoinen ja tekee saman asian:

```
include(CONTENT_PATH.'/model/register.php');
$model = new SomeModelRegister();
```

Tehtavä 2. Controller luo näkymän.

Koska joka kerta luodaan näkymä - eng. "view", sen luomiseen on annettu valmis metodi getView(string), jonka ainoata argumenttia koskee yllä esitelty nimeämiskäytäntö.
Ohjelmoija voi ohittaa getView() metodin kutsun. Jos SomeController->getView('register') luo olion SomeView Register tiedostosta view/register/register.php, on seuraava koodi vaihtoehtoinen ja tekee saman asian:

```
include(CONTENT_PATH.'/view/register/register.php');
$view = new SomeViewRegister();
```

Tehtävä 3. Controller antaa mallin näkymälle

Controllerin yhdistää mallin ja näkymän. Tämän se tekee siksi, että näkymä luodessaan html koodin, tarvitsee datan, jonka se saa mallista. Controlleri, jolla on siis näkymä-olio, antaa sille

mallin:
$view->setModel($model);

Tehtävä 4. view->display($tmpl) metodin kutsuminen

Controlleri kutsuu näkymän display() metodia, jonka seurauksena html koodi luodaan.
$view->display();

Huomaa, että vaikka yllä on esitetty getModel ja getView vaihtoehtoiset ohjelmakoodit, näitä metodeita pitäisi käyttää, sillä niiden sisältö voi nyt tai tulevaisuudessa tehdä jotain muutakin, kuin yksinkertaisesti yhden olion intanssin luomisen.

Controllerin omien metodien nimeäminen

SomeController kutsuu aina display() metodia, jos muuta metodia ei ole annettu. Sen execute() metodi tarkastaa kuitenkin url parametrin view, ja jos vastaavan niminen metodi löytyy, niin se kutsuu sitä (kutsun delegointi). Siten osoite http://www.example.org?app=account&view=register kutsuu SomeControllerDefault metodia register(), jos sellainen on tehty. Muuten kutsutaan metodia SomeControllerDefault::display().

SomeView ja tmpl/ kansio ja $this olion käyttö template tiedostossa.

SomeView oliolla on display() metodi, ja SomeViewRegister oliolla on display metodi. Controllerin kutsuessa SomeViewRegister olion display(tmpl) metodia, kutsutaan aina viimeksi periyttäneen luokan metodia. Siis, kun SomeViewRegister perii luokan SomeView, niin kutsutaan SomeViewRegister luokan metodia. Jos haluaa kutsua parent luokan display metodia, siis tässä tapauksessa SomeView luokan metodia, käytettään syntaksia:

```
parent::display($tmpl).
```

Luokan SomeViewRegister display() metodi hakee mallin ja sieltä sellaiset datat kun tarvitsee. Se voi myös muuttaa template tiedoston, ennen kuin kutsuu parent::display($tmpl) metodia.
Esimerkki display metodista, joka hakee mallin ja kutsuu sen getUsername() metodia.

```
public function display($tmpl=null) {
$model = $this->getModel(); //malli, jonka controller asetti
$this->username = $model->getUsername();
parent::display($tmpl);
}
```

Kun tiedot on asetettu oliolle sen display metodissa, liitetään template ja data toisiinsa. Tämä tapahtuu SomeView::display($tmpl='default') metodissa. Tämä metodi kutsuu toista metodia, loadTemplate, jonka koko koodi on alla.

```
protected function loadTemplate($tmpl) {
$suffix = $this->getName();
//$this variable is also available in included file
include(CONTENT_PATH.DS.'view'.DS.$suffix.DS.'tmpl'.DS.
$tmpl.'.php');
}
```

$this->getName() on apumetodi, joka hakee luokan nimestä – muista nimeämiskäytäntö - SomeView jälkeen olevan osan, siis 'register', jos luokan nimi on SomeViewRegister.

CONTENT_PATH on vakio, jonka arvo on polku kulloiseenkin content hakemistoon, siis account tapauksessa content/account.

Jos $tmpl ei ole annettu, se on aiemmin saanut arvon default.php. Siis template tiedosto joka haetaan on:

content/account/view/register/tmpl/default.php

Template tiedostossa voidaan käyttää oliota, joka template tiedoston include lauseella sisällytti itseensä, siis oliota itseään: $this. Tämä selittyy sillä, että include tosiasiassa sisällyttää koodia aivan kuin se olisi kirjoittu include lauseen tilalle suoraan tiedostoon – osaksi metodin runkoa. Kaikki include lausetta ennen näkyvissä olevat muuttujat ovat myös template tiedossa näkyvissä. Se olio, jonka metodissa include lausetta käytetään on SomeViewRegister luokan instanssi. Siis se sama $this instanssi, joka asettaa display() metodissa valmiiksi arvoja mallista. Siksi tiedostossa

content/account/view/register/tmpl/default.php voidaan tulostaa username koodilla:

```
echo $this->username;
```

Esimerkki

Esimerkkinä käytetään rekisteröintinäkymän luomista account sovelmassa. Nämä tiedostot ovat kurssiin sovelluskehyksessä valmiina. Tiedostot ovat:

content/account/user.php
content/account/controller/default.php
content/account/model/register.php
content/account/view/register/register.php
content/account/view/register/tmpl/form.php

Tiedosto content/account/user.php

Tämän tiedoston ainoa tehtävä jokaisessa sovelmassa on luoda controller ja kutsua sen execute metodia. Jos controller on tiedostossa default.php nimellä SomeControllerDefault, se luodaan tuomalla luokka näkyviin ja luomalla siitä instanssi. Mitään apumetodeita ei ole olemassa.

Tiedosto content/account/controller/default.php

SomeControllerDefault voi toteuttaa kaiken koodin sen display() metodissa, mutta on selkeämpää luoda jokaiselle näkymälle oma metodi. Oletusmetodia voi kutsua display() metodista. Esimerkiksi register näkymän luonti voidaan suorittaa metodissa register(). On metodin nimi mikä hyvänsä, sen on 1) luotava malli, 2) luotava näkymä, 3) annettava malli näkymälle ja 4) kutsuttava näkymän display metodia. Controller voi myös kutsua mallin jotakin funktiota. Esimerkiksi jos se toteaa käyttäjän lähettäneen rekisteröitymislomakkeen, se voi kutsua $model->doregister() metodia.

Esimerkki register näkymän register() metodin toteutuksesta SomeControllerDefault luokassa:

```
public function register() {
//nimeämiskäytäntö!
```

```
$model = $this->getModel('register');
$username = SomeRequest::getVar('username',null);
//jos username on olemassa, käyttäjältä tulee lomake, yritä
käsitellä se
if ($username) {
$success = $model->doregister(); // paluu arvo voi kertoa
onnistumisesta tai epäonnistumisesta.
if ($success) {
$view = $this->getView('register'); // eli
view/register/register.php pitää löytyä
$view->setModel($model);
$view->display('succesful'); // eli
view/register/tmpl/successful.php tiedosto pitää löytyä
} else {
//epäonnistui, ehkä ei ollut validia dataa
$view = $this->getView('register'); // eli
view/register/register.php pitää löytyä
$view->setModel($model);
$view->display('form'); // eli view/register/tmpl/form.php
tiedosto pitää löytyä
}
} else {
//ei ole lomakkeen lähetys, näytä lomake
$view = $this->getView('register'); // eli
view/register/register.php pitää löytyä
$view->setModel($model);
$view->display('form'); // eli view/register/tmpl/form.php
tiedosto pitää löytyä
}
}
```

Tiedosto content/account/model/register.php

Malli SomeModelRegister tiedostossa register.php toteuttaa sellaiset datan käsittelyt, joita tarvitaan rekisteröitymislomakkeen vastaanottamiseen, validointiin ja tallentamiseen tietokantaan.

Tiedosto content/account/view/register/register.php

View SomeViewRegister tiedostossa register.php toteuttaa sellaiset haut mallista, jotka tuovat templatelle dataa rekisteröitymislomakkeen luomiseksi. Tällaisia tietoja ovat esimerkiksi virheellistä dataa sisältäneen lomakkeen virheviestit ja lomakkeen kenttien oletusarvot.

Tiedosto content/account/view/register/tmpl/form.php

Tämä tiedosto sisältää html koodia, joka luo rekisteröitymislomakkeen.

Muut käyttäjään liittyvät tapahtumat.

Käyttäjän rekisteröitymisen jälkeen hän voi kirjautua sisään ja ulos. Loogisesti nämä voivat olla näkymiä login ja logout account modulissa. Sovelluskehyksessä on menty pidemmälle kuin luomalla omat mallit ja näkymät kirjautumistoiminnoille: ne on toteuttu omana modulinaan content/login.Tekemällä eri näkymät näille toiminnoille, saadaan tiedostot ja luokat:

/content/login/view/login/login.php, jossa luokka SomeViewLogin

Tämä tiedosto luodaan controllerissa antamalla getView metodin parametriksi 'login'.

/content/login/view/logout/logout.php

Tämä tiedosto luodaan controllerissa antamalla getView metodin parametriksi 'logout'.
Jos login tai logout tapahtumiin tarvitaan malli, Model, niin sellainen voisi olla nimetty oletusmalliksi 'SomeModelDefault'. Silloin tämä malli on tiedostossa

/content/login/model/default.php

ja tuodaan controllerissa getModel metodilla antamalla parametriksi 'default': getModel('default').

Malleja ja näkymiä voi luoda tarvittavan määrän. Mallien ja näkymien lisäksi on mahdollista luoda

eri html template tiedostoja tmpl/ kansioihin.

Esimerkiksi login ja logout voivat aina esittää oletuksenakin etsittävän default.php tiedoston:

/content/login/view/logout/tmpl/default.php
/content/login/view/login/tmpl/default.php

Jos templatetiedoston haluaa vaihtaa, sen voi controllerissa antaa $view->display() metodille parametrina:

```
public function login() {
$model = $this->getModel('login');
//use model here for what it is for. login user?
$view = $this->getView('login');
$view->setModel($view);
$view->display('other'); // will look for tmpl/other.php
}
```

Silloin template, jota login view käyttää, on

/content/login/view/login/tmpl/other.php

Tällainen template tiedoston vaihtaminen voi olla hyödyksi esimerkiksi onnistuneen rekisteröitymisen jälkeen. Silloin form -templaten sijaan voi esimerkiksi vaihtaa successful -templaten. Templaten voi vaihtaa vielä SomeViewRegister display($tmpl=null) metodissa.

Koodiesimerkki

Kommentoitu esimerkki content/example on sovelluskehyksessä. Siellä on kommentoitu niitä asioita, joita tässäkin on pyritty selvittämään, siis mitä mikäkin parametri tarkoittaa, mitä niiden perusteella oletetaan olevan olemassa, jne... Lisäksi siinä käytetään samaa mallia kahdessa eri näkymässä, ja muutetaan template tiedostoja yhden näkymän sisällä.