Keretrendszerek

Miért van szükség keretrendszerre?

- ömlesztett kód: HTML(, CSS), PHP, egyben van
- ömlesztett logika: vezérlés, adatelérés, megjelenítés, feldolgozás, alkalmazáslogika egyben van
- csoportmunka nem lehetséges
- konfiguráció kódba épített
- karakterkódolás eltérő lehet a HTML-ben, PHP-ban és adatbázisban, nincs egységesen kezelve
- több belépési pontja van az alkalmazásnak, ami biztonsági és jogosultsági kérdéseket vet fel

Mi az a keretrendszer?

- bizonyos filozófiának megfelelő szabályok gyűjteménye
- ahány keretrendszer, annyi féle
- szabályok korlátokat is jelentenek
- egységesebb alkalmazásfejlesztés
- szétválasztott kód és logika
- meghatározott könyvtárszerkezet
- csoportmunka támogatott: külön file-o az egyes szakembereknek (designer, adatbázisos, felületfejlesztő, stb.)
- rétegek szétválasztása

Problémák a jelenlegi fejlesztéssel

- nagyobb munkák írása egyre nagyobb nehézségekbe ütközik
- a kód felesleges elágazásokat tartalmaz, nem lehet tudni, mikor mi hívódik meg

Front Controller

- minden oldalt érintő közös műveletek egy helyen szerepelnek
  - konfiguráció beolvasása
  - autentikáció
  - autorizáció
  - input paraméterek előfeldolgozása, szűrése
  - karakterkódolás
  - stb.
- 1 belépési pont: index.php
- Hogyan jelezhető, melyik oldalt kell megjeleníteni:
  - hidden mező
  - GET paraméterrel: index.php?oldal=main

Példa: Front Controller 1.

```
switch ($oldal) {
    case "index":
        index();
        break;
    case "page1":
        page1();
        break;
    case "page2":
        page2();
        break;
    case "page3":
        page3();
        break;
    default:
        index();
}
```

```
function index() {

}

function page1() {

}

function page2() {

}

function page3() {

}
```

Példa: Front Controller 2.

```
if (function_exists($oldal))
{
   call_user_func($oldal);
}
```

```
function index() {

}

function page1() {

}

function page2() {

}

function page3() {

}
```

Példa: Front Controller 3.

```
switch ($oldal) {
    case "index":
        include "v3_index.php";
        break;
    case "page1":
        include "v3_page1.php";
        break;
    case "page2":
        include "v3_page2.php";
        break;
    case "page3":
        include "v3_page3.php";
        break;
    default:
        include "v3_index.php";
}
```

Példa: Front Controller 4.

```
if (file_exists("v3_" . $oldal . ".php")) {
    include_once("v3_" . $oldal . ".php");
}
```

Példa: Zenealkalmazás átírás Front Controllerrel

- index.php (Front Controller)
- linkek átírása (az alkalmazás eddig is funkcionálisan elkülönülő modulokból állt)
- ...

Példa: Zenealkalmazás átírás Front Controllerrel (v.2.)

Beszúráskor, módosításkor és törléskor rögtön a listaoldalra dobjon vissza

Front Controller: megjegyzés

A FrontController használata még nem segít az egyes rétegek szétválasztásában. A kód ugyanúgy ömlesztett marad az egyes részmodulokban

MVC: Model - View - Controller

Egyik lehetséges alkalmazási modell.

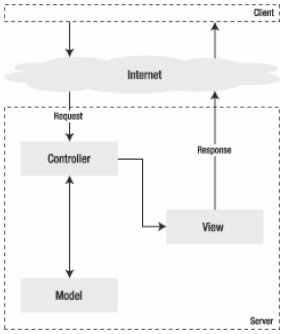
- **Model**: az adatokat, praktikusan az adatbázisbeli táblák, néha beleértik ezek elérését segítő kódot vagy magát az üzleti réteget is
- **View**: megjelenítésért szükséges réteg
- **Controller**: vezérlést végzi, értesíti a Modelt és a Viewt is a változásokról, kezeli az input adatokat.

Össze szokták kapcsolni a Front Controller mintával.

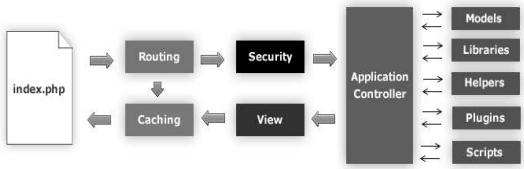
MVC példa: symphony project

Symphony MVC

MVC: Oldal életciklusa 1.



MVC: Oldal életciklusa 2.



Saját MVC-s keretrendszer készítése

- elvárások, szabályok megfogalmazása
- könyvtárszerkezet
- view, template (PHP nyelvi elemek: echo, for, foreach, while)
- front controller (index.php)

Feladat: Zenealkalmazás keretrendszerben

...