WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

A PHP segítségével fejlesztett programok (rendszerek) megírásához sok esetben egy keretrendszert használunk. A keretrendszer egy egységes architektúra, amely gyűjtemények használatára és manipulálására szolgál. Olyan programot célszerű készíteni, amely a szoftvertermék teljes ciklusában kevés költséggel, de maximálisan megfelel a követelményeknek.

A probléma megközelítése

A keretrendszerek kiválasztásánál felmerül néhány kérdés:

- Milyen feladatra szeretném használni?
- A keretrendszer milyen környezetet igényel?
- Milyen költsége van?

Előnyök: Már kevés idő alatt is megtanulható egy-egy keretrendszer kezelése. A hozzáadott költség hamar is visszatérül. A készített programcsomag megfelel a dobozos terméknek.

Hátrányok: Egyes dobozos keretrendszernek nagyon sok korlátja van, kisebb feladat esetén is fel kell használni a teljes csomagot, ami nem mindig jó.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

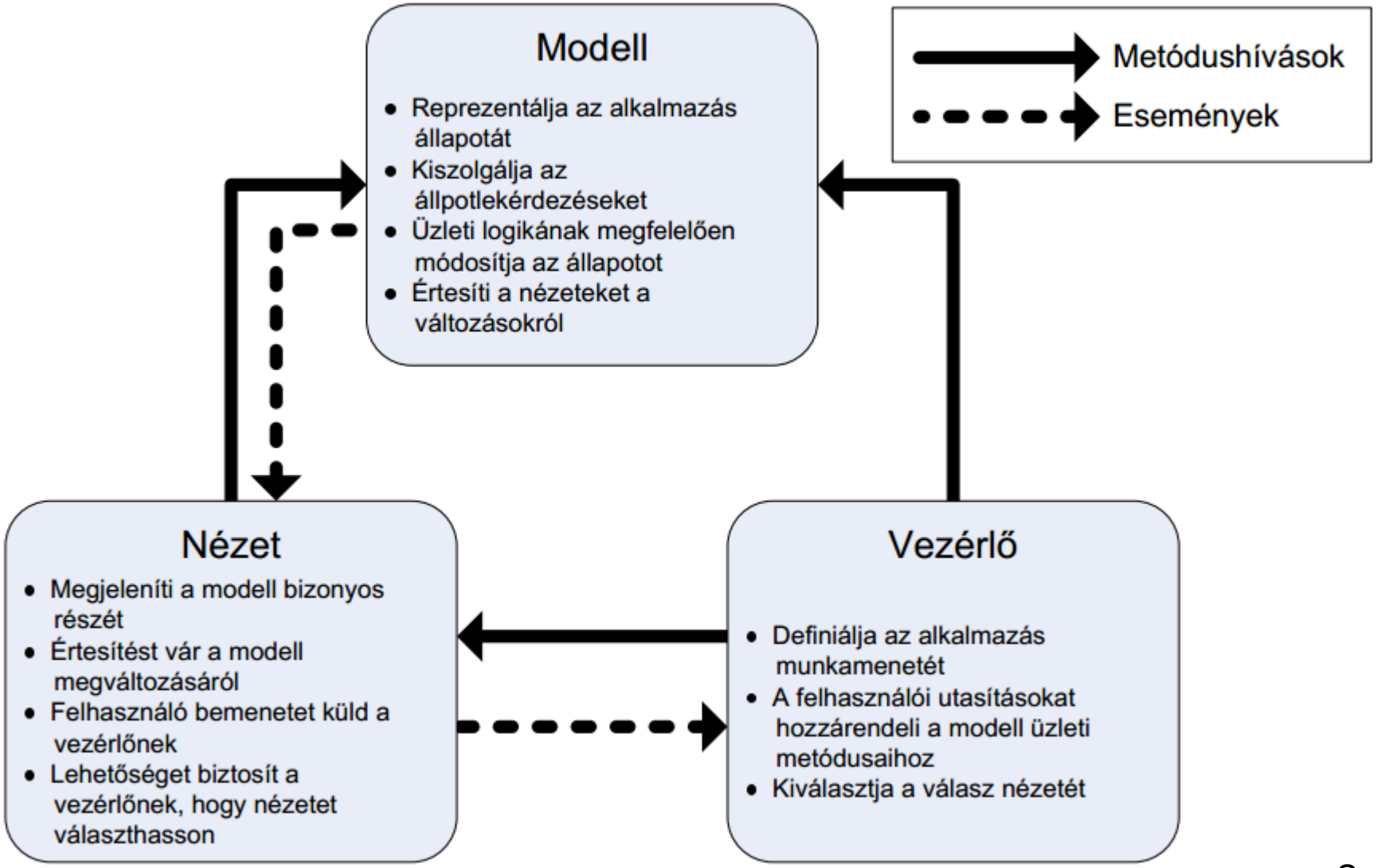
Az MVC tervezési minta (az MVC megközelítés):

A Model-View-Controller (Modell-Nézet-Vezérlő) egy gyakran használatos szoftver tervezési minta, amely három meghatározó részre osztja az alkalmazást:

1. Modell: az alkalmazás által kezelt információk ábrázolására szolgál.
2. Nézet: jellemzően felhasználói felületet biztosít a modell adatainak megtekintésére, kezelésére.
3. Vezérlő: az eseményeket, felhasználói műveleteket dolgozza fel és közvetíti a megfelelő modulokhoz.

Az ilyen típusú felbontás eredményeképpen a felhasználói felület nem befolyásolja az adatkezelést, így szükség esetén könnyebben átszervezhető a struktúra, a külön csoportba tartozó modulok változtatása nélkül. Ennek hatására az MVC-n keresztül csökken az alkalmazás szerkezeti bonyolultsága és nő a rugalmasság.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta



WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

MVC megvalósítása **PHP**-ben:

Alap felépítés

Egy **PHP** alkalmazás készítésekor az alapvető **MVC** struktúra létrehozásához szükségünk lesz három mappára:
• **models**: alkalmazás belső adatszerkezeteit működtető osztályokat definiáló fájlok, például adatbázisok kezelését megvalósító fájlok,
• **views**: a böngészőben megjelenítendő tartalom, jellemzően HTML kódot tartalmazó fájlok,
• **controllers**: azok a php fájlok, amelyek a kéréseket feldolgozva eljuttatják a megfelelő adatokat a modelltől a nézetig,
• és egy index.php fájlra, amely egészen egyszerű lehet.

Érdemes az index.php-ben definiálni az alkalmazás gyökér könyvtárát a szerveren, és az URL cím elérését, hogy ezeket később kényelmesen tudjuk használni. Amire mindenképpen szükség van, az annak a vezérlőnek a futtatása, amely a megfelelő vezérlők betöltését végzi. Ez ebben a példában a router.php fájlban található.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Az **index.php** fájl:

```
<?php

// alkalmazás gyökér könyvtára a szerveren
define('SERVER_ROOT', $_SERVER['DOCUMENT_ROOT'].'/mvc-php/');

// URL cím az alkalmazás gyökeréhez
define('SITE_ROOT', 'http://localhost/mvc-php/');

// a router.php vezérlő betöltése
require_once(SERVER_ROOT . 'controllers/' . 'router.php');

?>
```

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Egy általános URL:

http://.../mvc-php/index.php?test&data=mvc

ahol:

- **mvc-php**: a webes alkalmazás gyökér könyvtára a serveren,
- **test**: az aloldal, amit megnyitni szeretnénk,
- **data=mvc**: a paraméterek (név – érték párok).

A paraméterek a **router** számára még nem fontos információk, ezeket majd a konkrét vezérlő kezeli.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Az **router.php** fájl:

```
<?php

// Felbontjuk a paramétereket. Az & elválasztó jellel végzett felbontás
// megfelelő lesz, első eleme a megtekinteni kívánt oldal neve.

$request = $_SERVER['QUERY_STRING'];
$params = explode('&', $request);
$page = array_shift($params); // a kért oldal neve
$vars = array(); // a paraméterek asszociatív tömbje létrehozása

foreach($params as $p) // a paraméterek tömbje feltöltése
{
    list($name, $value) = explode('=', $p);
    $vars[$name] = $value; // az index: a paraméter neve
}
```

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Az **router.php** fájl folytatása:

```
// Meghatározzuk a kért oldalhoz tartozó vezérlőt. Ha megtaláltuk
// a fájlt és a hozzá tartozó vezérlő oldalt is, akkor betöltjük az
// előbbiekben lekérdezett paramétereket továbbadva.

$target = SERVER_ROOT.'controllers/'.$page.'.php';
if(file_exists($target))
{
	include_once($target);
	$class = ucfirst($page).'_Controller';
	if(class_exists($class))
	{ $controller = new $class; }
	else
	{ die('class does not exists!'); }
}
else
{ die('page does not exist!'); }
$controller->main($vars);
```

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Az **router.php** fájl befejezése:

```
// __autoload függvény, amely ismeretlen osztály használatakor,
// megpróbálja automatikusan betölteni a megfelelő fájlt.
// A modellekhez használjuk, egységesen nevezzük el fájljainkat
// (osztály nevével megegyező, csupa kisbetűs .php)

function __autoload($className)
{
	$file = SERVER_ROOT.'models/'.strtolower($className).'.php';
	if(file_exists($file))
	{ include_once($file); }
	else
	{ die("File '$filename' containing class '$className' not found."); }
}

?>
```

Konkrét vezérlők:

A konkrét vezérlő (jelen esetben **Test_Controller**) egy osztály.

A Test_Controller osztály a hívott oldal viselkedését definiálja.

A példában csak egy **main** metódusa van, amely betöltődik kezdetkor.

Szükség van egy modell és egy nézet betöltésére.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

A **test.php** konkrét vezérlő fájl:

```
<?php

class Test_Controller
{
	public $baseName = 'test';  //meghatározni, hogy melyik oldalon vagyunk
	public function main(array $vars) // a router által továbbított paramétereket kapja
	{
		$testModel = new Test_Model;  //az osztályhoz tartozó modell
		if(isset($vars['data']))
		{
			//modellből lekérdezzük a kért adatot
			$reqData = $testModel->get_data($vars['data']);
			//betöltjük a nézetet
			$view = new View_Loader($this->baseName.'_main');
			//átadjuk a lekérdezett adatokat a nézetnek
			$view->assign('title', $reqData['title']);
			$view->assign('content', $reqData['content']);
		}
		else
		{ echo "No data to show"; }
	}
}

?>
```

Modellek:

Mint láttuk a vezérlő osztályunkban, mindenképpen szükségünk lesz egy konkrétan a vezérlőhöz tartozó modellre, ez lesz a **Test_Model**.

Ezen kívül lesz egy általánosabb modell osztályunk, ami a nézetek betöltéséért felel majd, ez a példában a **View_Loader** nevet kapta.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

A **test_model.php** konkrét modell fájl:

Jelen példában az egyszerűség kedvéért csak egy privát adattagként definiált tömbben tároljuk az adatokat, de jellemzően a modell része szokott lenni valamiféle adatbázissal való kommunikáció is.

```
<?php

class Test_Model
{
	private $data = array
	('new' => array('title' => 'New Website', 'content' => 'Welcome to the site!'),
	 'mvc' => array('title' => 'PHP MVC Framework', 'content' => 'works good'));

	public function get_data($title)
	{
		$retData = $this->data[$title];
		return $retData;
	}
}

?>
```

A **View_Loader** osztály:

Felel a nézetek megfelelő módon történő betöltéséért. Ehhez:

•Be kell azonosítania a kívánt nézet fájlt, amely a **views** mappában lesz. Ezt már a konstruktorban teheti meg.

•Meg kell oldania, hogy a nézet megjelenítésekor, a kívánt értékek elérhetők legyenek. Ez egy hozzárendelő **assign** nevű függvény segítségével történik, amely egy privát tömbben tárolja le ideiglenesen az adatokat.

•Végül az osztály destruktorában átadja az adatokat egy változónak, ami elérhető lesz a nézetből és betölti a nézetet.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

A **view_loader.php** általános modell fájl:

```
<?php

class View_Loader
{
    private $data = array();
    private $render = FALSE;

    public function __construct($viewName)
    {
        $file = SERVER_ROOT . 'views/' . strtolower($viewName) . '.php';
        if (file_exists($file))
        { $this->render = $file; }
    }
```

A **view_loader.php** fájl befejezése:

```
    public function assign($variable , $value)
    {
        $this->data[$variable] = $value;
    }

    public function __destruct()
    {
        $viewData = $this->data;
        include($this->render);
    }
}

?>
```

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Nézetek:

A nézetek megvalósításánál már nagyon egyszerű, alapvetően HTML kódot kell készíteni, amelybe beszúrhatjuk a **View_Loader** által eltárolt változók értékeit. A következő példa az előbb látott **viewData** tömbbe kimentett értékeket használja a beillesztéshez.

A **test_main.php** nézet fájl:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>MVC - PHP</title>
    </head>
    <body>
        <h1> Welcome! </h1>
        <hr/>
        <h2> Requested data: </h2>
        <h4> <?php echo $viewData['title']; ?> </h4>
        <p> <?php echo $viewData['content']; ?> </p>
    </body>
</html>
```

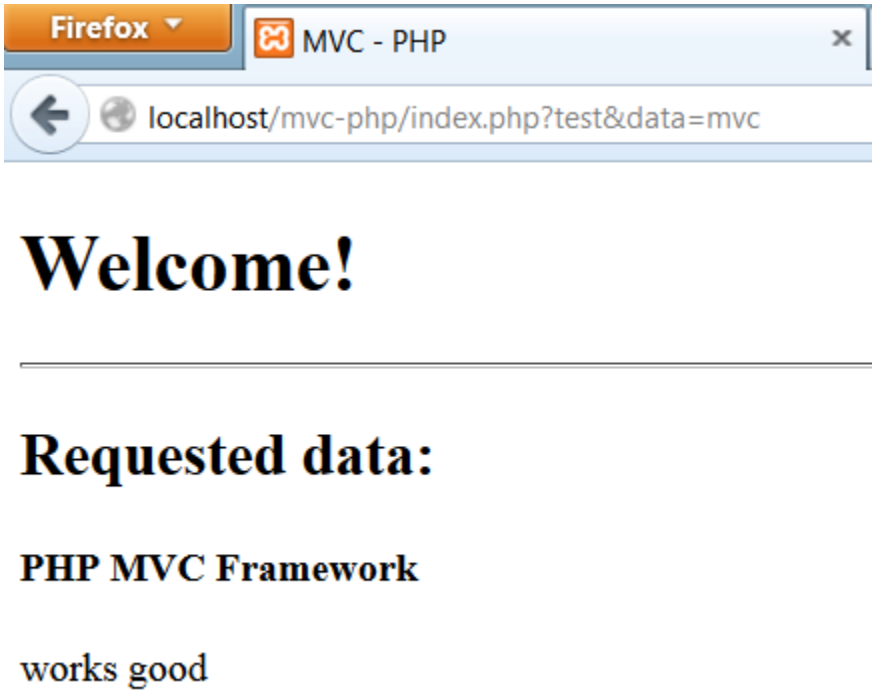
WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

A test nézet definíciója után a

http://.../mvc-php/index.php?test&data=mvc

hivatkozásra meg kell, hogy jelenjenek a modellben az **'mvc'** azonosítóhoz definiált adatok, ami jelen esetben a **'works well'** szöveg.

A várt eredmény a böngészőben:



WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Felhasználó- és keresőbarát URL-ek használata:

http://.../mvc-php/index.php?test&data=mvc

helyett

http://.../mvc-php-url/test/mvc

Megjegyzés: A felhasználó- és keresőbarát url-eket használó példa az **mvc-php-url** könyvtárban található.

Ehhez az mvc-php-url könyvtárban helyeztük el egy .htaccess fájlt, amelynek tartalma:

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?$1 [L,QSA]
```

Jelentése: Ne alkalmazzuk az átalakító szabály, ha az URL fizikai fájlt vagy könyvtárt jelöl, különben az url elé helyezzük az index.php? –t.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

A forrásokban végrehajtott módosítások:

A http://.../mvc-php-url/test/mvc url az átalakítás után http://.../mvc-php-url/index.php?test/mvc lesz, ami miatt át kell alakítani a router.php-ben a paraméterek felbontását:

```
$request = $_SERVER['QUERY_STRING'];
$params = explode('/', $request);
$page = array_shift($params); // a kért oldal neve
$vars = array(); // a paraméterek tömbje létrehozása

foreach($params as $p) // a paraméterek tömbje feltöltése
{
	$vars[ ] = $p;
}
```

A $vars['data'] változó most nem létezik, helyette a $vars[0] változót használjuk a test.php kontrollerben.

WEB-programozás II (2. előadás)
PHP keretrendszerek – MVC tervezési minta

Eredmények különböző URL-ekre: