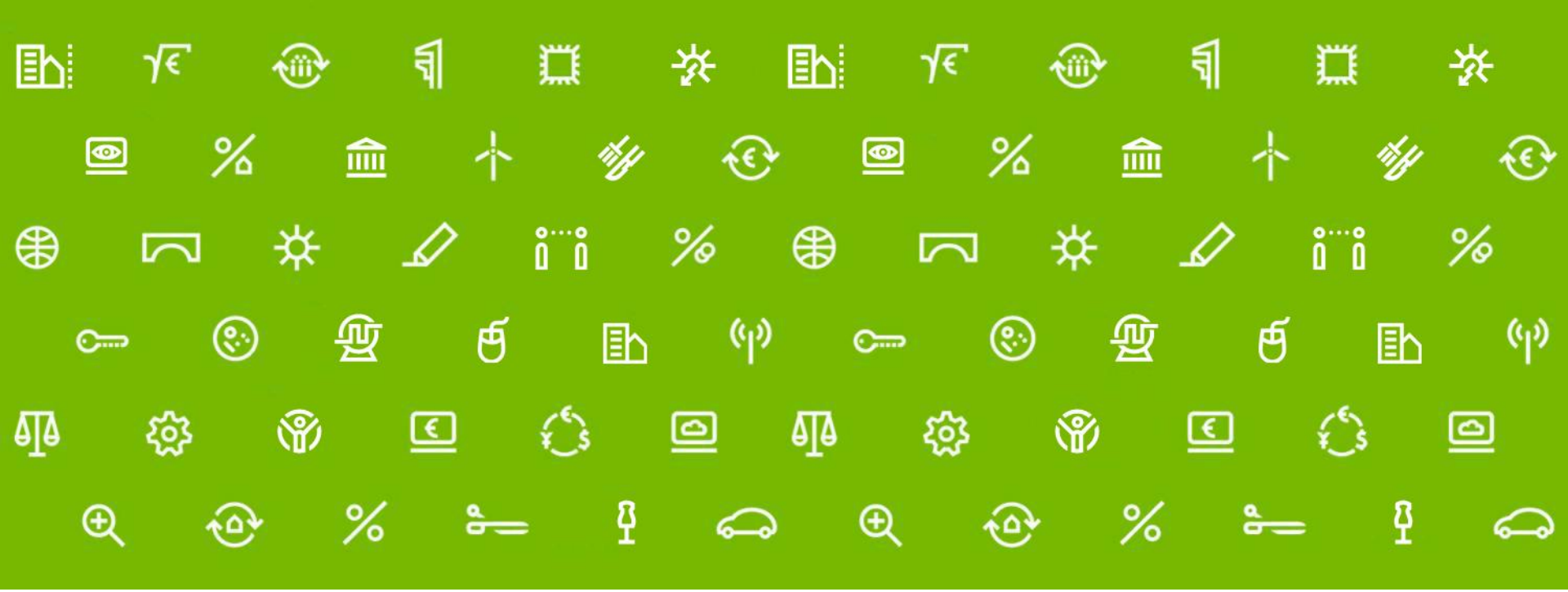


Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Zend-Framework: Design Patterns

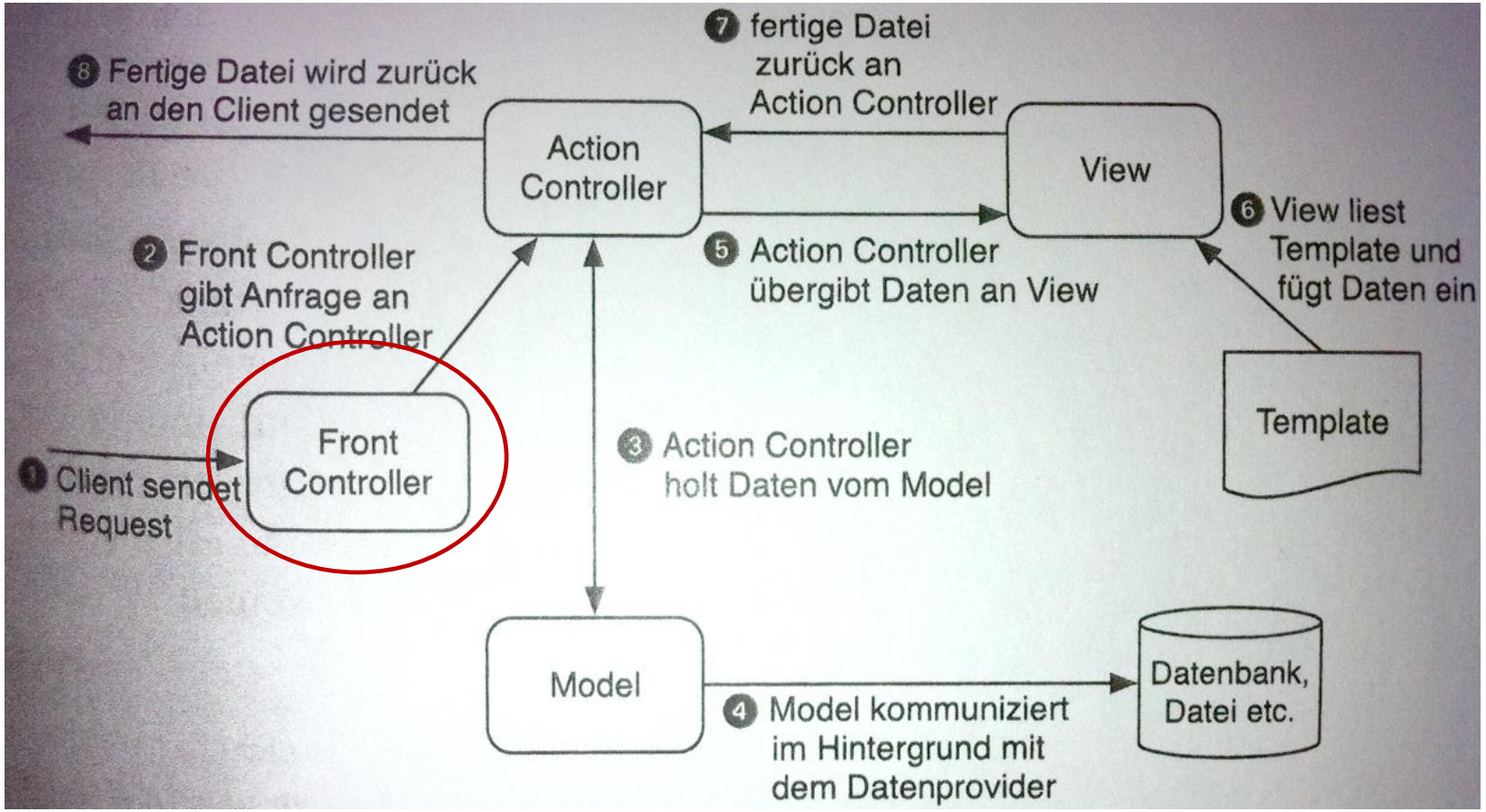
Gliederung: Design Patterns am Beispiel des Zend Frameworks

1. MVC
2. Front-Controller
3. Singleton
4. Row-Data-Gateway und Table-Data-Gateway
5. Registry

Die Implementierung des MVC im Zend Framework

1. MVC

MVC


7 fertige Datei zurück an Action Controller
8 Fertige Datei wird zurück an den Client gesendet
Action Controller
View
6 View liest Template und fügt Daten ein
2 Front Controller gibt Anfrage an Action Controller
5 Action Controller übergibt Daten an View
Front Controller
Template
1 Client sendet Request
3 Action Controller holt Daten vom Model
Model
Datenbank, Datei etc.
4 Model kommuniziert im Hintergrund mit dem Datenprovider

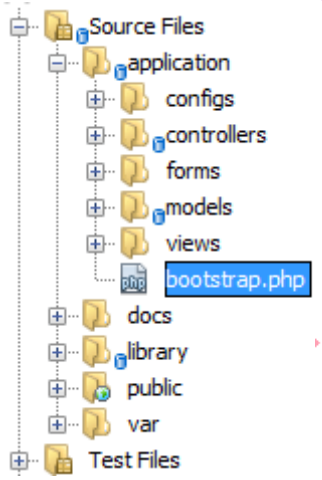
Das Front Controller Entwurfsmuster innerhalb des MVC im Zend Framework

2. Front Controller

2. Front Controller

- Zuständigkeit
  - Alle Anfragen an zentraler Stelle entgegen zu nehmen
  - sich wiederholende Aufgaben bei allen Anfragen auszuführen
    - Konfigurationen der DB, Error Handling, Verzeichnisse, Zend_ACL, CSS und JS einbinden, Bibliotheken
  - Weiterleiten an jeweilige Action-Controller
  - bootstrap.php, Komponente: Zend_Controller_Front

2. Front Controller




```php
<?php
class Bootstrap
{
    public function __construct($configSection = 'dev')
    {
        $rootDir = dirname(dirname(__FILE__));
        define('ROOT_DIR', $rootDir);

        set_include_path(get_include_path()
            . PATH_SEPARATOR . ROOT_DIR . '/library/'
            . PATH_SEPARATOR . ROOT_DIR . '/application/models/'
                . PATH_SEPARATOR . ROOT_DIR . '/application/forms/'
        );
        require_once 'Zend/Loader/Autoloader.php';
        $autoloader = Zend_Loader_Autoloader::getInstance();
        $autoloader->setFallbackAutoloader(true);
        // Load configuration
        Zend_Registry::set('configSection', $configSection);
        $config = new Zend_Config_Ini(ROOT_DIR.'/application/configs/config.ini', $configSection);
        Zend_Registry::set('config', $config);

        date_default_timezone_set($config->date_default_timezone);

        // configure database and store to the registery
        $db = Zend_Db::factory($config->db);
        Zend_Db_Table_Abstract::setDefaultAdapter($db);
        Zend_Registry::set('db', $db);
    }

    public function runApp()
    {
        // setup front controller
        $frontController = Zend_Controller_Front::getInstance();
```

Instanzierung mit dem Singleton Pattern

3. Singleton Pattern

3. Singleton Pattern

- Sicherstellen, dass von einer Klasse nur eine Instanz zur Laufzeit existiert
- Speicherersparnis
- Zugriff immer auf dasselbe Element
- Implementierung:
  - Konstruktor kann nicht aufgerufen werden von außerhalb
  - Nur statische Methode getInstance()
  - Bei Folgeaufrufen gibt Funktion bereits instanziertes Objekt zurück
- Clone ist nicht möglich

3. Singleton Pattern: Bsp: ZEND_AUTH

```php
protected function __construct()
{}

/**
 * Singleton pattern implementation makes "clone" unavailable
 *
 * @return void
 */
protected function __clone()
{}

/**
 * Returns an instance of Zend_Auth
 *
 * Singleton pattern implementation
 *
 * @return Zend_Auth Provides a fluent interface
 */
public static function getInstance()
{
    if (null === self::$_instance) {
        self::$_instance = new self();
    }

    return self::$_instance;
}
```

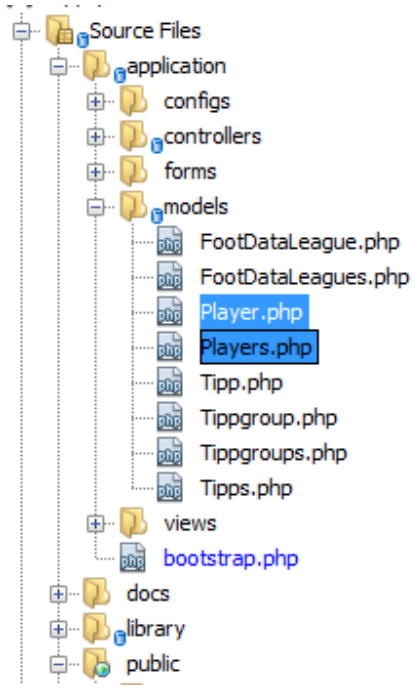
Trennung des Zugriffs auf eine Zeile und eine Tabelle in der DB

4. Row-Data-Gateway und Table-Data-Gateway

4. Row-Data-Gateway und Table-Data-Gateway

- Operationen und Zugriff einer einzelne Zeile in der DB
- Gegenstück: Table-Data-Gateway
  - Zugriff auf eine komplette Tabelle
- Löschen, ändern, einfügen
- Sinn: Trennung der Logik, Übersicht im Code
- Komponente: ZEND_DB_TABLE_ROW und ZEND_DB_TABLE

4. Row-Data-Gateway und Table-Data-Gateway




```php
class Players extends Zend_Db_Table_Abstract
{
    //DB-Tabellen Name
    protected $_name = 'players';
    //Klasse der Zeilen-Objekte
    protected $_rowClass = 'Player';
    //1:n Beziehung mit Tippgroups
    protected $_referenceMap = array(
            'Tippgroups' => array(
            'columns' => array('tippgroup_id'),
            'refTableClass' => 'Tippgroups',
            'refColumns' => array('id')
            )
            );
    function fetchLatest($count = 10)
    {
        //ODer:$this->fetchAll(null,'date_created DESC', $count);
        $select = $this->select();
        $select->order("date_created DESC");
        $select->limit($count);
        return $this->fetchAll($select);
    }
}
```

zentraler Datenspeicher

5. Registry

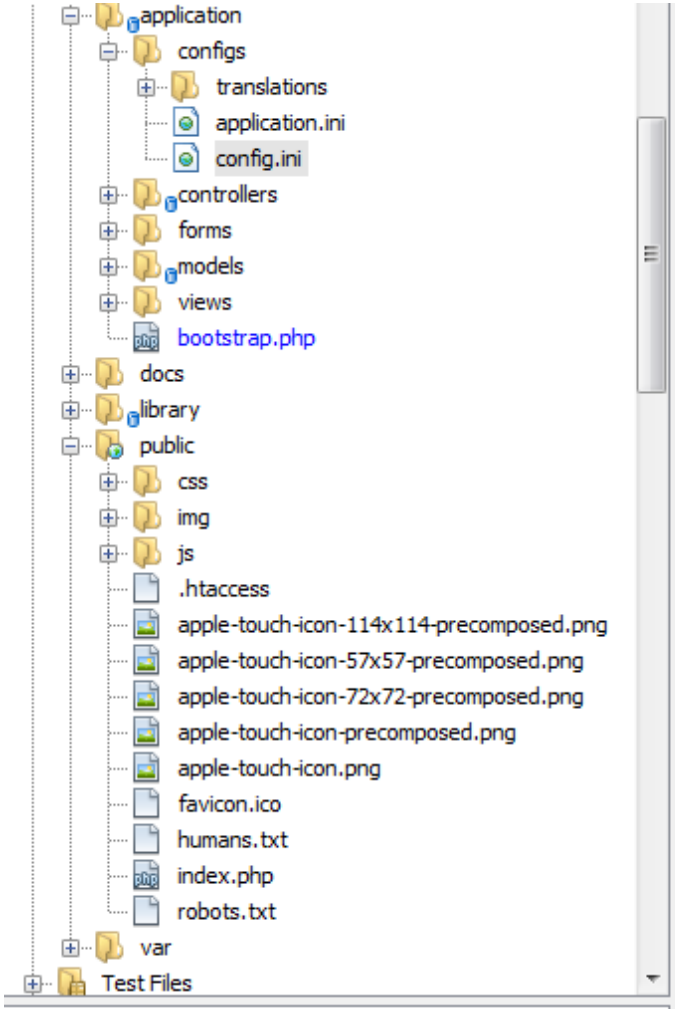
5. Registry

- Zweck:
  - Zugriff auf gemeinsam genutzte Informationen
  - Globale Variablen können überschrieben werden, Registry nicht
- Zentraler Speicher für:
  - Konfiguration
  - Lokalisierung
- Zugriff von View und Controller
- Komponente: ZEND_REGISTRY

5. Registry

```php
<?php
class Bootstrap
{
    public function __construct($configSection = 'dev')
    {
        $rootDir = dirname(dirname(__FILE__));
        define('ROOT_DIR', $rootDir);

        set_include_path(get_include_path()
            . PATH_SEPARATOR . ROOT_DIR . '/library/'
            . PATH_SEPARATOR . ROOT_DIR . '/application/models/'
                . PATH_SEPARATOR . ROOT_DIR . '/application/forms/'
        );
        require_once 'Zend/Loader/Autoloader.php';
        $autoloader = Zend_Loader_Autoloader::getInstance();
        $autoloader->setFallbackAutoloader(true);
        // Load configuration
        Zend_Registry::set('configSection', $configSection);
        $config = new Zend_Config_Ini(ROOT_DIR.'/application/configs/config.ini', $configSection);
        Zend_Registry::set('config', $config);

        date_default_timezone_set($config->date_default_timezone);

        // configure database and store to the registery
        $db = Zend_Db::factory($config->db);
        Zend_Db_Table_Abstract::setDefaultAdapter($db);
        Zend_Registry::set('db', $db);
    }

    public function runApp()
    {
        // setup front controller
        $frontController = Zend_Controller_Front::getInstance();
```

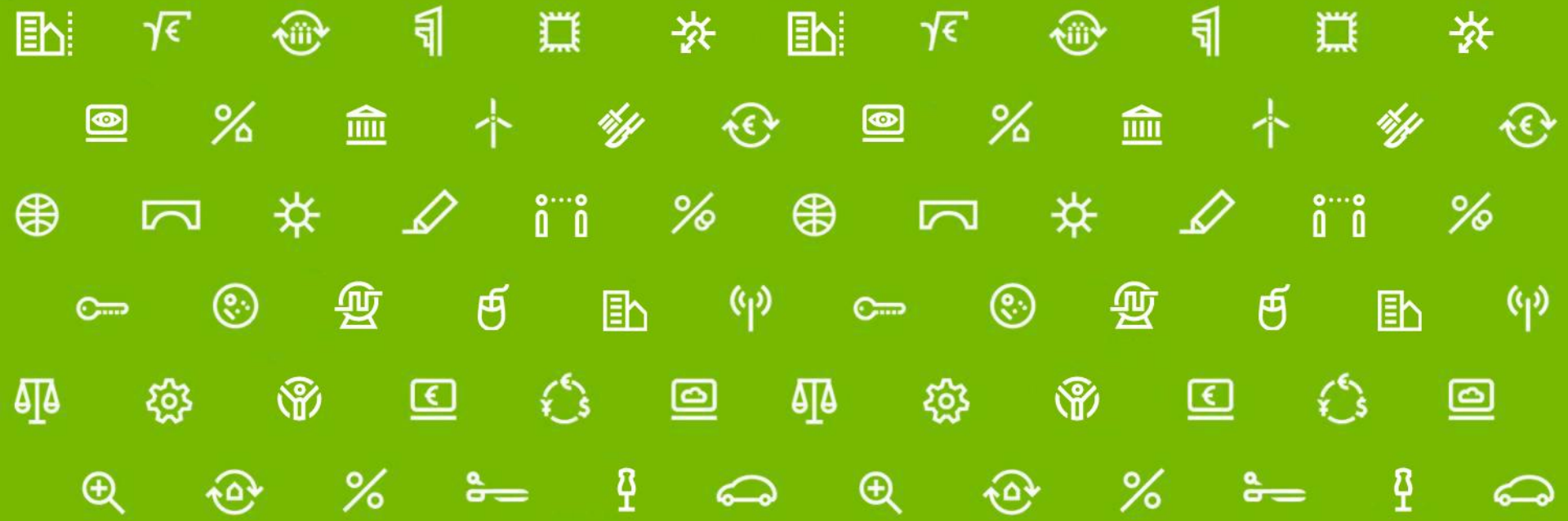
5. Registry: congi.ini




```
1  [general]
2  db.adapter = PDO_MYSQL
3  db.params.host = localhost
4  db.params.username = root
5  db.params.password = ''
6  db.params.dbname = tippspiel
7  db.params.charset = "utf8"
8  date_default_timezone = "Europe/Berlin"
9
10 //Zugriffsberechtigungen Zend_Acl
11 auth.salt = "la3"
12 acl.roles.guest = null
13 acl.roles.member = guest
14 acl.roles.admin = member
15
16 [live : general]
17 debug = 0
18 logFiles.error = "../../var/logs/ApplicationErrors.txt"
19 [dev : general]
20 debug = 1
21 logFiles.error = "php://output"
22
23 [test : general]
24 debug = 1
25 db.params.username = db
26 db.params.password = fsd
27 db.params.dbname = d
28 logFiles.error = "php://output"
```

Beispiel Projekt

http://www.sebastianviereck.de/tippspiel/public/



Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Vielen Dank für Ihre Aufmerksamkeit!