Entwicklerdokumentation

Modulhandbuch-Verwaltungssystem der FH Bingen (MHB)

Version 1.0

Projektteam
Mathias Dickenscheid
Ingmar Jakob
Mischa Nehrbass
Christian Wittig

Kunde
Prof. Dr. Michael Schmidt

Änderungshistorie		
Änderung	**Autor**	**Datum**
V0.5 komplette Erarbeitung	Christian Wittig	04.02.15
V0.6 Entities fertig gestellt	Christian Wittig	05.02.15
V0.7 Controlller eingefügt, Entities fertig gestellt und beschrieben	Mischa Nehrbass	06.02.15
V0.8 FormTypes eingefügt	Mischa Nehrbass	07.02.15
V0.9 FormTypes, Verwaltungs Controler eingefügt	Ingmar Jakob	07.02.15
V0.91 Views eingefügt	Ingmar Jakob	08.02.15
V0.92 weitere Entity-Beschreibungen und Funktionsbeschreibungen eingefügt	Mischa Nehrbass	08.02.15
V0.93 Grundaufbau hinzugefügt	Ingmar Jakob, Mischa Nehrbass	08.02.15
V0.94 Klassendiagramm, Grundaufbau	Mathias Dickenscheid	15.02.15
V1.0 Authentifizierung, Projekstruktur, Layout	Mischa Nehrbass, Mathias Dickenscheid	20.02.15

Inhaltsverzeichnis

1. Grundaufbau

1.1. MySQL

MySQL ist ein relationales Datenbankmanagementsystem (RDBMS).

Damit ein MySQL-Server auch außerhalb von localhost erreichbar ist, muss man folgende Konfiguration anpassen:

```
in `/etc/mysql/my.cnf`
`bind-address = 0.0.0.0`
oder `bind-address = IP_DES_SERVERS`
ggf. `#skip-networking` (auskommentieren)
```

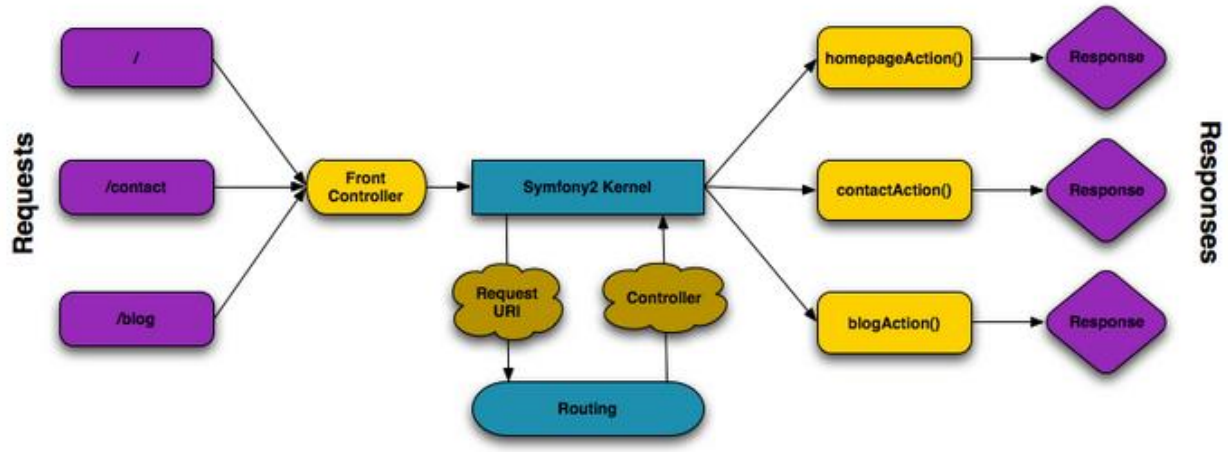
1.2. Composer

Composer ist ein Kommandozeilen-Paketmanager für PHP. Er wird benötigt um Symfony und alle zugehörigen Abhängigkeiten auf einem System zu installieren.
Composer kann von https://getcomposer.org/download bezogen werden.

Befehl für die Symfony-Installation:
composer create-project symfony/framework-standard-edition Pfad/zum/Installationsort

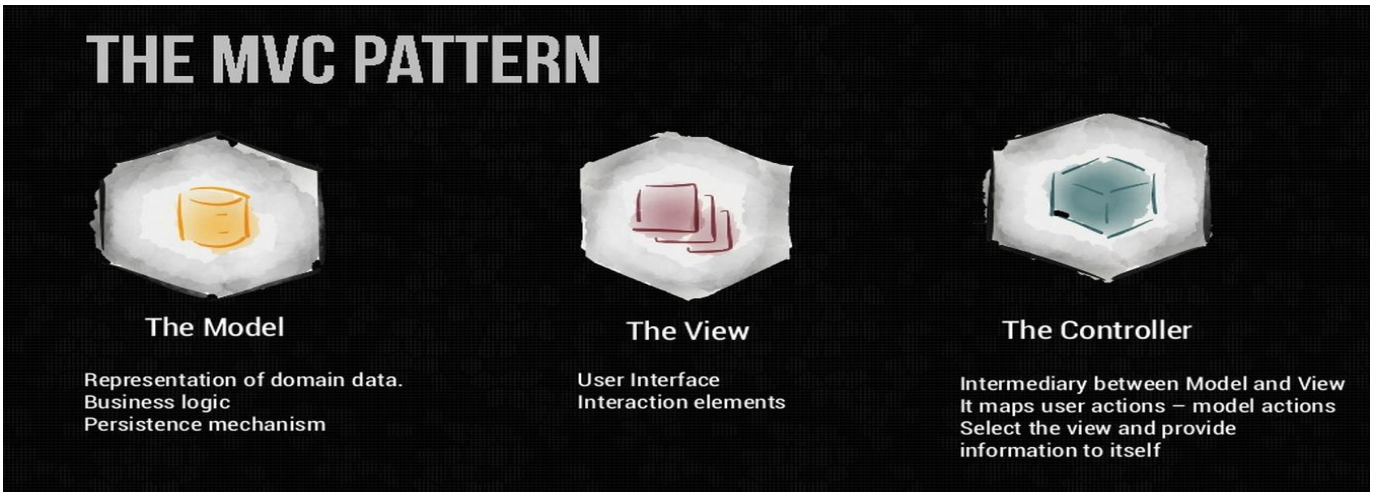
1.3. Symfony

Symfony ist ein kostenloses open-source PHP-Framework von SensioLabs.
Es verfolgt ein MVC-Pattern und implementiert damit klassisches HTTP-Request-Response-Handling.
Symfony verwendet zur Ausführung unterschiedliche Umgebungen und kapselt damit gezielt die Produktivumgebung von der Entwicklungs- oder Testumgebung ab.



Wir verwenden Symfony in der Version 2.6.3, es wird jedoch aktiv von einer großen Entwickler-Community weiterentwickelt, weshalb bereits dieses Jahr mit dem nächsten Major-Release, Version 3.0, zu rechnen ist.
(http://symfony.com/doc/current/contributing/community/releases.html)

Wir nutzen mit Symfony das Model-View-Controller-Prinzip. Grundidee ist es Daten, Verwaltung und Darstellung voneinander zu trennen. Alle drei Teile können untereinander kommunizieren.



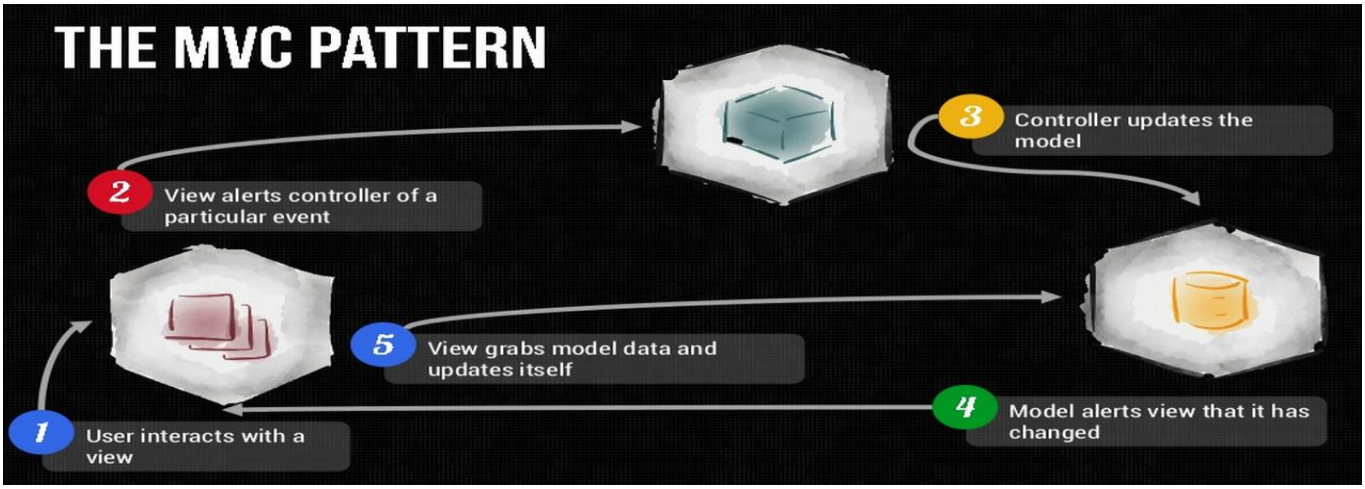
Quelle:http://de.slideshare.net/javierhumaran/model-view-controller-mvc-27875933

Das Model enthält die zu verwaltenden Daten. Es liefert bei Anfrage den Zustand der aktuellen Daten und setzt Bearbeitungsbefehle auf diese durch.

Ein View ist die für den Benutzer sichtbare Benutzeroberfläche und stellt die Daten aus dem Model dar. Die Aufgabe des Views ist die visuelle Darstellung der Daten aus dem View.

Der Controller stellt die Verbindung zwischen Model und View dar. Über ihn können die Daten manipuliert werden. Er definiert wie der Nutzer mit den Daten aus dem View interagieren kann. Er nimmt Eingaben der Nutzer entgegen und leitet diese an das Model weiter.

Das Zusammenspiel der drei Komponenten wird im nächsten Bild dargestellt.



Quelle:http://de.slideshare.net/javierhumaran/model-view-controller-mvc-27875933

Im klassischen Model-View-Controller-Prinzip existiert eine Observer-Beziehung zwischen Model und View, welche den View updatet, wenn die Daten im Model geändert werden. In Symfony kümmert sich aber der Controller darum den View mit Daten aus dem Model zu beliefern. Wird im View eine Anpassung der Daten vorgenommen, so wird der Controller alarmiert und die geänderten Daten an das Model weiterreicht. Die Daten werden angepasst und der Controller fragt das Model nach den aktuellen Daten und gibt diese zur Darstellung an den View weiter.

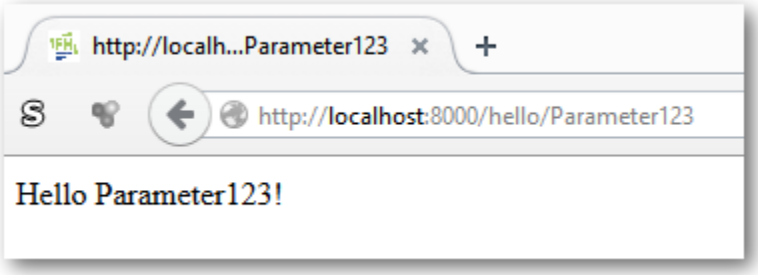
Das Model-View-Controller-Pattern hat zahlreiche Vorteile.
Der wichtigste Aspekt ist aber die Wiederverwertbarkeit. Views zum Beispiel können für beliebige Daten angewendet werden, da die drei Komponenten separat voneinander implementiert werden. Es ermöglicht auch freiere Gestaltungsmöglichkeiten beim Design. Zum Beispiel vereinfacht das MVC-Prinzip die Darstellung mehrerer unterschiedlicher Sichten auf die gleichen Daten. Außerdem wird die Wartung und Fehlerfindung ungemein vereinfacht.

1.3.1. Symfony im Einsatz

- Controller:

```php
/**
 * @Route("/hello/{name}")
 */
public function helloWorldAction($name)
{
    return new Response('Hello ' . $name . '!');
}
```

- Anwort im Browser bei Aufruf von '/hello/Parameter123'



Häufige Befehle

- alle Symfony Befehle können mit dem Switch --env=prod ausgeführt werden, um sie auf die Produktiv-Umgebung anzuwenden. (default --env=dev)

- php app/console server:run
  - Startet den internen Web-Server

```
C:\xampp\htdocs\mhb-vs>php app/console server:run
Server running on http://127.0.0.1:8000

Quit the server with CONTROL-C.
```

- php app/console assets:install
  - Installiert den Ordner 'Resources/public' (Bilder, CSS, JavaScript, usw.) in den öffentlichen Ordner 'web' des jeweiligen Bundles
  - Wenn Datei-System Symlinks unterstützt, kann der Switch --symlink angehängt werden. Dadurch vermeidet man es, den Befehl nach Änderungen der Assets neu auführen zu müssen. (Funktioniert nicht unter Windows)

```
C:\xampp\htdocs\mhb-vs>php app/console assets:install
Installing assets as hard copies.
Installing assets for Symfony\Bundle\FrameworkBundle into web/bundles/framework
Installing assets for FHBingen\Bundle\MHBBundle into web/bundles/fhbingenmhb
Installing assets for Sensio\Bundle\DistributionBundle into web/bundles/sensiodistribution
```

- php app/console cache:clear
  - Bereinigt den Cache der jeweiligen Umgebung. Kann bei Fehlern, insbesondere wenn sie bei in den Twigs auftreten, helfen.

```
C:\xampp\htdocs\mhb-vs>php app/console cache:clear
Clearing the cache for the dev environment with debug true
```

1.4. Twig

Twig ist eine Template-Engine für PHP und Bestandteil des Symfony-Frameworks und wird ebenso von SensioLabs entwickelt.

Twig im Einsatz

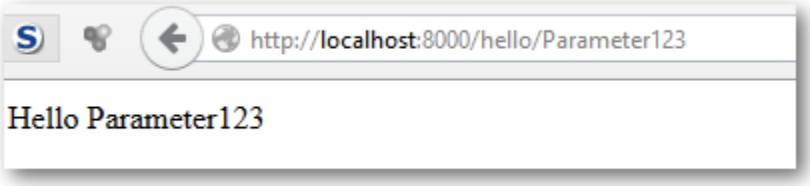
- Controller:

```php
/**
 * @Route("/hello/{name}")
 */
public function helloWorldTwigAction($name)
{
    return $this->render('@FHBingenMHB/helloWorld.html.twig', array('name' => $name));
}
```

- Twig-Template:

```
{# helloWorld.html.twig #}
<html>
<head>
    <title>
        Twig-Beispiel
    </title>
</head>
<body>
    <p>Hello {{ name }}</p>
</body>
</html>
```

- Anwort im Browser bei Aufruf von '/hello/Parameter123'



1.5. Doctrine

Doctrine stellt mit seinem Object Relational Mapper (ORM) und dem Abstract Database Layer (DBAL) eine Art Middleware zwischen PHP und SQL dar.
Doctrine ermöglicht es auf einfache Weise PHP-Objekte in Datenbanken zu hinterlegen. Statt per SQL-Befehl greift man durch das ORM objektorientiert auf die Daten zu. In Symfony wird dies durch den sogenannten Entity-Manager realisiert.
Die objektrelationale Funktionalität wird den Objekten selbst zugewiesen, ohne dass hierfür eine aufwändige Programmierung nötig wäre.

Damit Doctrine mit den Objekten korrekt umgehen kann, müssen die sogenannten Entities mit bestimmten Annotations versehen werden.

```php
// src/AppBundle/Entity/Product.php
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\Table(name="product")
 */
class Product
{
    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @ORM\Column(type="string", length=100)
     */
    protected $name;

    /**
     * @ORM\Column(type="decimal", scale=2)
     */
    protected $price;

    /**
     * @ORM\Column(type="text")
     */
    protected $description;
}
```

Im Controller sieht es dann wie folgt aus:

```php
// src/AppBundle/Controller/DefaultController.php

// ...
use AppBundle\Entity\Product;
use Symfony\Component\HttpFoundation\Response;

// ...
public function createAction()
{
    $product = new Product();
    $product->setName('A Foo Bar');
    $product->setPrice('19.99');
    $product->setDescription('Lorem ipsum dolor');

    $em = $this->getDoctrine()->getManager();

    $em->persist($product);
    $em->flush();

    return new Response('Created product id '.$product->getId());
}
```

Häufige Befehle

- php app/console doctrine:database:create
  - Erstellt eine Datenbank via Doctrine
- php app/console doctrine:generate:entity
  - Interaktiver Kommandozeilen-Wizard zum Erzeugen eines neuen Entities
- php app/console doctrine:generate:entities AppBundle/Entity/Product
  - Erzeugt Getter und Setter für ein spezielles Entity (hier Product)
- php app/console doctrine:schema:validate
  - Validiert das aktuelle Doctrine-Mapping
- php app/console doctrine:schema:update --dump-sql
  - Gibt die SQL-Befehle aus, die benötigt werden um die Datenbank mit dem Mapping zu synchronisieren
- php app/console doctrine:schema:update --force
  - Schreibt das aktuelle Mapping in die Datenbank. Diese Änderungen können nicht rückgängig gemacht werden!

Näheres hierzu unter http://symfony.com/doc/current/book/doctrine.html

1.6. KnpSnappyBundle / Snappy / wkhtmltopdf:

wkhtmltopdf ist ein open-source Kommandozeilen-Tool um HTML in PDF umzuwandeln. (http://wkhtmltopdf.org)
Snappy ist ein PHP-Wrapper für wkhtmltopdf. (https://github.com/KnpLabs/snappy)
KnpSnappyBundle bietet eine simple Integration von Snappy in ein Symfony-Projekt. (https://github.com/KnpLabs/KnpSnappyBundle).

1.7. Authentifizierung

Ziel unserer Authentifizierungslösung ist es, den Dozentenbereich vom Studiengangleiterbereich unseres Systems zu trennen. Wir setzen dabei auf die integrierte Firewall-Lösung von Symfony.

Der Login erfolgt später über die Email-Adresse des jeweiligen Dozenten.

Die Konfiguration erfolgt dabei hauptsächlich über die Datei „security.yml“.

```yaml
# mhb-vs/app/config/security.yml
security:
    encoders:
        pwenc:
            algorithm: bcrypt
            cost:      12

    role_hierarchy:
        ROLE_SGL: [ROLE_DOZENT]

    providers:
        users:
            entity: { class: FHBingenMHBBundle:Dozent }

    firewalls:
        login:
            pattern:  ^/restricted/login$
            security: false

        secured_area:
            pattern:    ^/restricted/
            form_login:
                login_path: login
                check_path: loginCheck
                default_target_path: eigeneModule
            logout:
                path:   logout
                target: login

    access_control:
        - { path: ^/restricted/sgl/, roles: ROLE_SGL }        # /restricted/sgl/*
        - { path: ^/restricted/sgl$, roles: ROLE_SGL }        # /restricted/sgl
        - { path: ^/restricted/dozent/, roles: ROLE_DOZENT }  # /restricted/dozent/*
        - { path: ^/restricted/dozent$, roles: ROLE_DOZENT }  # /restricted/dozent
```

Das Design wurde so gewählt, dass ein möglicher nachträglicher Umstieg auf eine Authentifizierung gegen einen LDAP-Server durchführbar ist, ohne dass dabei Kernfunktionalitäten des Gesamtsystems umgeschrieben werden müssen.

1.7.1. encoders

Als erstes legen wir uns einen „Encoder“ an und nennen ihn „pwenc“. Dieser legt fest, wie die Passwörter der Benutzer zu speichern sind. Wir verwenden die mitgelieferte „bcrypt“-Implementierung mit einem Kostenfaktor von 12.

Näheres zu bcrypt findet sich z.B. auf Wikipedia unter https://de.wikipedia.org/wiki/Bcrypt.

Nun weisen wir unsere Benutzer-Entities an unseren Encoder „pwenc“ zum abgleichen der Passwörter beim Login zu verwenden. Dies geschieht über den Aufruf der Funktion „getEncoderName()“

```
public function getEncoderName()
{
    //always return encoder 'pwenc' defined in security.yml
    return 'pwenc';
}
```

Außerdem müssen wir dafür sorgen, dass beim Anlegen von Passwörter diese bcrypt-gehasht in die Datenbank geschrieben werden. Dies geschieht über eine Modifikation der „setPassword()“-Funktion unserer Benutzer.

```
public function setPassword($password)
{
    //$this->password = $password;
    $this->password = password_hash($password, PASSWORD_BCRYPT, array('cost' => 12));

    return $this;
}
```

Außerdem müssen wir dafür sorgen, dass beim Anlegen von Passwörter diese bcrypt-gehasht in die Datenbank geschrieben werden. Dies geschieht über eine Modifikation der „setPassword()“-Funktion unserer Benutzer.

1.7.2. role_hierarchy

In unserem System gibt es zwei Rollen, die des Dozenten und die des Studiengangleiters. Diese sind als „ROLE_DOZENT“ bzw. „ROLE_SGL“ in unsere Datenbank eingetragen. (Merke: Symfony benötigt den Präfix „ROLE_“ vor dem eigentlichen Rollennamen.)

Da wir wollen, dass ein Studiengangleiter aber ebenfalls alle Funktionen eines Dozenten aufrufen kann, legen wir dies unter dem Punkt „role_hierarchy“ der Konfigurationsdatei fest.

1.7.3. providers

Als nächstes wird dem System gesagt, welche Entities seine Benutzer-Objekte sind.

Wir haben kein dediziertes Benutzer-Entity, sondern haben diese Funktionalität in unser Dozenten-Objekt integriert. Achtung: Es gibt keine Studiengangleiter-Entities, lediglich Dozenten-Entities mit der Rolle „ROLE_SGL“.

1.7.4. firewalls

Jetzt legen wir eine Firewall mit dem Namen „secured_area“ an, die dafür sorgt, dass man sich authentifizieren muss um auf Funktionen die hinter dem Pfad „/restricted“ liegen zuzugreifen. Alle Funktionen unseres Systems liegen hinter „/restricted/dozent“ respektive „/restricted/sgl“.

Damit die Login-Seite weiterhin ohne Authentifizierung erreichbar bleibt, erstellen wir eine Firewall namens „login“, in der wir die Login-Route freigeben, indem wir „security“ auf „false“ setzen.

1.7.5. access_control

Nun muss noch festgelegt werden, welche Bereiche des Systems für welche Rolle zugänglich sind. Dies geschieht über die Rollenzuweisungen unter dem Punkt „access_control“ der Konfigurationsdatei.

1.7.6. Login via E-Mail-Adresse

Standardmäßig würde jetzt eine Authentifizierung über einen Benutzernamen erfolgen. Da wir jedoch einen Login via E-Mail-Adresse umsetzen wollen, müssen zwei kleine Änderungen vorgenommen werden.

Zuerst müssen wir neues UserRepository schreiben, welches wir anweisen, die E-Mail-Adresse als Identifier seiner Benutzer zu verwenden. Dies geschieht, indem wir die „loadUserByName()“-Funktion wie folgt abändern:

```
public function loadUserByUsername($username)
{
    $q = $this
        ->createQueryBuilder('u')
        //->where('u.username = :username OR u.Email = :email')
        ->where('u.email = :email')
        //->setParameter('username', $username)
        ->setParameter('email', $username)
        ->getQuery();
```

Jetzt müssen wir noch unseren User-Entities unser neues UserRepository zuweisen.
Dies geschieht über die „repositoryClass“-Anweisung der „@ORM\Entity“-Annotation in unserer Dozenten-Klasse.

```
/**
 * Class Dozent
 *
 * @package FHBingen\Bundle\MHBBundle\Entity
 * @ORM\Entity(repositoryClass="FHBingen\Bundle\MHBBundle\PHP\UserRepository")
 * @ORM\Table(name="Dozent")
 * @UniqueEntity(fields="email", message="Unter dieser EMail ist bereits ein Dozent eingetragen.")
 * @ORM\HasLifecycleCallbacks
 */
class Dozent implements UserInterface, AdvancedUserInterface, \Serializable, EncoderAwareInterface
```

1.7.7. Anmerkungen

Auf Kundenwunsch gibt es nur noch jeweils ein Passwort für Dozenten und Studiengangleiter. Dieses befinden sich, zusammen mit allen anderen Passwörtern des Systems, in der Datei „Passwörter.pdf“.

1.8. Projektstruktur

```
MHBBundle/
├── Controller
│   ├── DefaultController.php
│   ├── DozentController.php
│   ├── LoginController.php
│   ├── SglController.php
│   └── VerwaltungsController.php
├── Entity
│   ├── Angebot.php
│   ├── Dozent.php
│   ├── Fachgebiet.php
│   ├── Kernfach.php
│   ├── Lehrende.php
│   ├── Modulhandbuch.php
│   ├── ModulhandbuchZuweisung.php
│   ├── Modulvoraussetzung.php
│   ├── Role.php
│   ├── Semester.php
│   ├── Semesterplan.php
│   ├── Studiengang.php
│   ├── Studienplan.php
│   ├── Veranstaltung.php
│   ├── VeranstaltungHistory.php
│   └── Vertiefung.php
├── FHBingenMHBBundle.php
├── Form
│   ├── AngebotType.php
│   ├── CodeType.php
│   ├── DozentType.php
│   ├── FachgebietType.php
│   ├── KernfachType.php
│   ├── LehrendeType.php
│   ├── ModulhandbuchType.php
│   ├── PlanungType.php
│   ├── SemesterplanType.php
│   ├── SemesterType.php
│   ├── StudiengangType.php
│   ├── StudienplanType.php
│   ├── VeranstaltungType.php
│   ├── VertiefungType.php
│   ├── VorAngebotType.php
│   └── VoraussetzungType.php
```

```
├── PHP
│   ├── ArrayValues.php
│   ├── ModulBeschreibung.php
│   ├── SortFunctions.php
│   ├── UserDependentRole.php
│   └── UserRepository.php
└── Resources
    ├── public
    │   ├── css
    │   │   ├── bootstrap.css
    │   │   ├── bootstrap.css.map
    │   │   ├── bootstrap.min.css
    │   │   ├── bootstrap-theme.css
    │   │   ├── bootstrap-theme.css.map
    │   │   ├── bootstrap-theme.min.css
    │   │   ├── jquery-ui.min.css
    │   │   ├── jquery-ui.structure.min.css
    │   │   ├── style.css
    │   │   └── tablesorter.css
    │   ├── fonts
    │   │   ├── glyphicons-halflings-regular.eot
    │   │   ├── glyphicons-halflings-regular.svg
    │   │   ├── glyphicons-halflings-regular.ttf
    │   │   └── glyphicons-halflings-regular.woff
    │   ├── images
    │   │   ├── asc.gif
    │   │   ├── bg.gif
    │   │   ├── bg-white.jpg
    │   │   ├── desc.gif
    │   │   └── logo.png
    │   ├── js
    │   │   ├── bootstrap.js
    │   │   ├── bootstrap.min.js
    │   │   ├── jquery.tablesorter.min.js
    │   │   ├── jquery-2.1.3.min.js
    │   │   ├── jquery-ui.min.js
    │   │   ├── npm.js
    │   │   └── own.js
    │   └── xsl
    │       ├── default.xsl
    │       └── toc.xsl
```

```
└── views
    ├── buttons.html.twig
    ├── Dozent
    │   ├── angebot.html.twig
    │   ├── eigeneModule.html.twig
    │   ├── modulBearbeiten.html.twig
    │   ├── planungBearbeiten.html.twig
    │   ├── planungUebersicht.html.twig
    │   └── vorAngebot.html.twig
    ├── flashbags.html.twig
    ├── layout.html.twig
    ├── Login
    │   └── login.html.twig
    ├── Navigation
    │   ├── adminNav.html.twig
    │   ├── dozentNav.html.twig
    │   ├── generalNav.html.twig
    │   ├── navigation.html.twig
    │   └── sglNav.html.twig
    ├── Prototypes
    │   └── listItem.html.twig
    ├── SGL
    │   ├── alleModule.html.twig
    │   ├── mhbErstellung.html.twig
    │   ├── mhbModul.html.twig
    │   ├── mhbUebersicht.html.twig
    │   ├── mhbZusammenstellung.html.twig
    │   ├── modulAenderungen.html.twig
    │   ├── modulCodeErstellung.html.twig
    │   ├── modulCodeUebersicht.html.twig
    │   └── modulDeaktivierung.html.twig
    └── Verwaltung
        ├── alleStudiengaenge.html.twig
        ├── benutzerErstellen.html.twig
        ├── benutzerVerwaltung.html.twig
        └── studiengangAnzeigen.html.twig
```

2. Controller

2.1. DefaultController

indexAction()

Beschreibung: -

Aufruf über:

- @Route("/")
- @Route("/index")

Übergabeparameter: -

Ablauf:

- Redirect auf Route @Route("/login")

Wichtige Infos: -

createRolesAction()

Beschreibung:
Initialerstellung von Userrollen

Aufruf über:

- @Route("/create/roles")

Übergabeparameter: -

Ablauf:

- Nach Aufruf dieser Methode werden die Userrollen initialisiert.

Wichtige Infos: -

2.2. DozentController

eigeneModuleAction()

Beschreibung:

Bei Aufruf dieser Funktion werden alle Module mit dem Status „freigegeben“, für die der aktuell eingeloggte Nutzer als Lehrender und/oder Modulbeauftragter angegeben wurde, in der Datenbank gesucht und auf der zu rendernden Seite in die entsprechende Tabelle eingetragen. Zusätzlich werden zu jedem Modul alle Lehrenden und Studiengänge in der Datenbank gesucht und in der entsprechenden Spalte in den Tabellen ausgegeben.

Aufruf über:

- @Route("/restricted/dozent/eigeneModule", name="eigeneModule")
- @Template("FHBingenMHBBundle:Dozent:eigeneModule.html.twig")

Übergabeparamter: -

Ablauf:

- Aktuell eingeloggter Dozent wird in Variable $user geschrieben
- Module mit dem Status „freigegeben“ für die der Dozent als Lehrender, bzw. Modulbeauftragter eingetragen ist, werden in der Datenbank gesucht und diese sortiert in ein Array geschrieben.
- In den Angeboten werden zu den sich im Array befindlichen Modulen Studiengänge gesucht und in ein weiteres Array geschrieben.
- Rendert „eigeneModule.html.twig".

Wichtige Infos: -

planungAnzeigenAction()

Beschreibung:

Bei Aufruf dieser Funktion werden alle Veranstaltungen des aktuell eingeloggten Dozenten mit dem Status „in Planung“ aus der Datenbank gesucht und tabellarisch geordnet.

Aufruf über:

- @Route("/restricted/dozent/planungAnzeigen", name="planungAnzeigen")
- @Template("FHBingenMHBBundle:Dozent:planungUebersicht.html.twig")

Übergabeparameter: -

Ablauf:

- Aktuell eingeloggter Dozent wird in Variable $user geschrieben
- Module mit dem Status „in Planung“ für die der Dozent als Modulbeauftragter eingetragen ist, werden in der Datenbank gesucht und diese sortiert in ein Array geschrieben.
- Rendert „planungUebersicht.html.twig"

Wichtige Infos: -

planungLoeschenAction()

Beschreibung:

Bei Aufruf dieser Funktion wird eine ausgewählte Planung in der Datenbank gesucht und unwiderruflich gelöscht.

Aufruf über:

- @Route("/restricted/dozent/planungLoeschen/{id}", name="planungLoeschen")

Übergabeparameter:

- $id → ID des Moduls, welches gelöscht werden soll

Ablauf:

- Aktuell eingeloggter Dozent wird in Variable $user geschrieben
- In der Datenbank wird das Modul mit der angegebenen ID, dem aktuell eingeloggten Dozent als Modulbeauftragtem und dem Status „in Planung“ gesucht und gelöscht.
- Redirect auf „Planung anzeigen“.

Wichtige Infos: -

planungErstellenAction()

Beschreibung:

Bei Aufruf dieser Funktion erfolgt eine Weiterleitung auf planungAction() mit ID -1 zur Identifikation einer neuen Planung.

Aufruf über:

- @Route("/restricted/dozent/planungErstellen", name="planungErstellen")

Übergabeparameter: -

Ablauf:

- Weiterleitung auf planungAction() mit ID -1.
- Maske mit eingetragenen Default-Werten zur Erstellung einer Planung erscheint.

Wichtige Infos:

- Da die Planung zur Zeit des Aufrufens noch nicht besteht wird die ID -1 übergeben.
- Bei Speicherung der Daten in die Datenbank wird jedoch eine eigene ID generiert.

planungAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Maske zur Bearbeitung einer Planung. Es erfolgt eine Prüfung auf die übergebene ID. Sollte diese -1 sein werden die vorgegeben Default-Werte in die Felder eingetragen, ansonsten werden die Daten der übergebenen ID in der Datenbank abgefragt und in die entsprechenden Felder eingetragen.

Aufruf über:

- @Route("/restricted/dozent/planungBearbeiten/{id}", name="planungBearbeiten")
- @Template("FHBingenMHBBundle:Dozent:planungBearbeiten.html.twig")

Übergabeparameter:

- $id → Entweder -1 falls eine neue Planung erstellt wird oder die jeweilige ID der angeklickten Planung

Ablauf:

- Sofern die ID -1 ist werden Default-Werte in die Maske eingetragen, ansonsten die in der Datenbank zur angegebenen ID hinterlegten Daten einer Planung.
- Rendert „planungBearbeiten.html.twig".

Wichtige Infos:

- planungErstellen ruft planungBearbeiten mit ID -1 auf.
- Die Zeiten des Selbststudiums sind nicht einsehbar und werden im Hintergrund berechnet.

modulBearbeitenAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Maske zur Bearbeitung eines freigegebenen Moduls. Anhand der übergebenen ID werden die Daten eines Moduls in der Datenbank abgefragt und in die entsprechenden Felder eingetragen.
Die Maske zur Bearbeitung einer Planung und eines freigegeben Moduls unterscheiden sich nur in wenigen Punkten. Bei einem freigegebenen Modul müssen alle Felder ausgefüllt werden, bei der Planung nur der deutsche Name. Außerdem kann in einem freigegebenen Modul der Modulbeauftragte geändert, sowie weitere Lehrende und formale Voraussetzungen hinzugefügt werden.

Aufruf über:

- @Route("/restricted/dozent/modulBearbeiten/{id}", name="modulBearbeiten")
- @Template("FHBingenMHBBundle:Dozent:modulBearbeiten.html.twig")

Übergabeparameter:

- $id → ID des ausgewählten Moduls das bearbeitet werden soll

Ablauf:

- Anhand der übergebenen ID wird das Modul in der Datenbank gesucht und die einzelnen Felder mit den entsprechend in der Datenbank hinterlegten Werten gefüllt.
- Rendert „modulBearbeiten.html.twig".

Wichtige Infos:

- Die Zeiten des Selbststudiums sind nicht einsehbar und werden im Hintergrund berechnet.

planungFreigebenAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Maske zur Bearbeitung einer Planung. In dieser Maske können nun aber auch Lehrende sowie formale Voraussetzungen gesetzt werden. Sollte kein Lehrender angegeben werden wird automatisch der Modulbeauftragte, sprich der Ersteller der Planung, als Lehrender eingetragen. Da diese Planung freigegeben wird, wird bei Klick auf speichern geprüft, ob alle Felder ordnungsgemäß ausgefüllt wurden. Ist dies der Fall erfolgt eine Weiterleitung auf die Funktion vorAngebotAction(), ansonsten erfolgen die entsprechenden Fehlermeldungen.

Aufruf über:

- @Route("/restricted/dozent/planungFreigeben/{id}", name="planungFreigeben")
- @Template("FHBingenMHBBundle:Dozent:modulBearbeiten.html.twig")

Übergabeparameter:

- $id → ID der ausgewählten Planung die frei gegeben werden soll

Ablauf:

- Anhand der übergebenen ID wird die Planung in der Datenbank gesucht und die einzelnen Felder mit den entsprechend in der Datenbank hinterlegten Werten gefüllt.
- Rendert „modulBearbeiten.html.twig".
- Weiterleitung auf vorAngebotAction(), falls die Felder nicht fehlerhaft gefüllt sind

Wichtige Infos:

- Sofern kein Lehrender eingetragen wurde, wird automatisch der Modulbeauftragte auch als Lehrender in die Datenbank eingetragen.

angebotAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Maske, mit der Möglichkeit zur Festlegung der Fachgebiete des zu erstellenden Angebots im gewähltem Studiengang, sowie zur Angabe von studiengangspezifischen Namen (deutsch und englisch). Falls der Status des Moduls, für das ein Angebot erstellt werden soll noch auf „in Planung“ steht, wird dieser auf „Freigegeben“ geändert. Nach Klick auf Speichern wird für das übergebene Modul zum angegebenen Studiengang ein Angebot angelegt, welches den Modulcode DUMMY erhält.

Aufruf über:

- @Route("/restricted/dozent/angebot/{studiengangID}/{modulID}/{angebotsart} /{encSS}/ {encWS}", name="angebot")
- @Template("FHBingenMHBBundle:Dozent:angebot.html.twig")

Übergabeparameter:

- $studiengangID → ID des Studienganges für den das Angebot gelten soll
- $modulID → ID des Moduls für das ein neues Angebot erstellt werden soll
- $angebotsart → Unterscheidung ob Wahl- oder Pflichtfach
- $encSS → JSON-String mit den ausgewählten Regelsemester bei Start im SS
- $encWS → JSON-String mit den ausgewählten Regelsemester bei Start im WS

Ablauf:

- Je nach Auswahl der Angebotsart im Vorangebot werden die Fachgebiete gefiltert.
- Falls Angebotsart Wahlpflichtfach gewählt wurde, besteht die Möglichkeit das Modul keiner, einer oder mehreren Vertiefungsrichtungen zu zuordnen.
- Rendert „angebot.html.twig".
- Nach Klick auf speichern wird ein neues Angebot für das ausgewählte Modul zum ausgewählten Studiengang erstellt.
- Optional ist die Auswahl eines optionalen Titels (deutsch und englisch).
- Falls der Status des angegebenen Moduls noch „in Planung“ ist, wird dieser auf „Freigegeben“ geändert.

Wichtige Infos:

- Neu angelegte Module haben standardmäßig den Modulcode „DUMMY“
- Vertiefungsrichtungen bisher nur in Informatik möglich

vorAngebotAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Maske zur Festlegung des Studiengangs in dem das Modul angeboten werden soll. Zusätzlich muss angegeben werden in welchem oder welchen Regelsemster das Modul angeboten wird. Außerdem kann der Nutzer angeben, ob das Modul in diesem Studiengang ein Wahl- oder Pflichfach sein soll.

Aufruf über:

- @Route("/restricted/dozent/vorAngebot/{modulID}", name="vorAngebot")
- @Template("FHBingenMHBBundle:Dozent:vorAngebot.html.twig")

Übergabeparameter:

- $modulID → ID des Moduls zu dem ein neues Angebot erstellt werden soll

Ablauf:

- Sofern das Modul schon in einem oder mehreren Studiengängen angeboten wird, werden diese vorab gefiltert und können nicht nochmals ausgewählt werden.
- Sollte der Nutzer keine Regelsemester angeben wird nach Klick auf speichern eine Fehlermeldung ausgegeben.

Wichtige Infos: -

2.3. LoginController

loginAction()

Siehe Symfony-Dokumentation.

securityCheckAction()

Siehe Symfony-Dokumentation.

logoutAction()

Siehe Symfony-Dokumentation.

2.4. SglController

Konstanten

2.4.1. MHB_PATH

Diese Konstante legt fest an welchem Ort die als PDF generierten Modulhandbücher abgelegt werden sollen. Standardmäßig werden diese unter dem Pfad ‚web/mhb/' abgelegt.

2.4.2. WKHTMLTOPDF_BIN_WIN

Diese Konstante beinhaltet den Pfad zur Windows-Binary von wkhtmltopdf.
Muss bei der Entwicklung unter Windows angepasst werden.

2.4.3. WKHTMLTOPDF_BIN_LIN

Diese Konstante beinhaltet den Pfad zur Linux-Binary von wkhtmltopdf.
Muss bei der Entwicklung unter Linux, bzw. beim Deployment angepasst werden

alleModuleAction()

Beschreibung:
Bei Aufruf dieser Funktion öffnet sich eine Tabelle, in der alle in der Datenbank vorhandenen Module mit dem Status „Freigegeben" sortiert ausgegeben werden. Weiterhin werden alle Studiengänge für die einzelnen Module in der Datenbank gesucht und in der entsprechenden Spalte ausgegeben.

Aufruf über:
- @Route("/restricted/sgl/alleModule", name="alleModule")
- @Template("FHBingenMHBBundle:SGL:alleModule.html.twig")

Übergabeparameter: -

Ablauf:
- Es werden alle freigegebenen Module sowie die Studiengänge der dazugehörigen Angebote in der Datenbank gesucht.
- Rendert 'alleModule.html.twig'.

Wichtige Infos: -

modulCodeUebersichtAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Tabelle, in der die im Studiengang des eingeloggten Nutzers angebotenen Module aufgelistet werden. Unterschieden wird hierbei ob es sich um neue Angebote mit dem Modulcode „DUMMY“ handelt oder um bereits bearbeitete Module die den korrekten Modulcode haben.

Aufruf über:

- @Route("/restricted/sgl/modulCodeUebersicht", name="modulCodeUebersicht")
- @Template("FHBingenMHBBundle:SGL:modulCodeUebersicht.html.twig")

Übergabeparameter: -

Ablauf:

- Aktueller Nutzer und der dazugehörige Studiengang werden ausgelesen.
- Die im Studiengang angebotenen Module mit und ohne Modulcode werden aus der Datenbank gesucht und jeweils in ein eigenes Array gespeichert.
- Rendert 'modulCodeUebersicht.html.twig'.

Wichtige Infos: -

modulCodeErstellungAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Maske, in der der Nutzer die Möglichkeit hat, den Modulcode eines in seinem Studiengang angebotenen Moduls zu ändern. Zu beachten hierbei ist, dass ein Modulcode aus genau 9 Zeichen bestehen muss.

Aufruf über:

- @Route("/restricted/sgl/modulCodeErstellung/{id}/{studiengangid}", name="modulCodeErstellung")
- @Template("FHBingenMHBBundle:SGL:modulCodeErstellung.html.twig")

Übergabeparameter:

- ID → ID des Moduls für das der Modulcode erstellt werden soll.
- Studiengangid → ID des Studienganges für das der Modulcode gelten soll.

Ablauf:

- In der Datenbank wird das Angebot, das den beiden Übergabeparametern entspricht, angefragt.
- Rendert 'modulCodeErstellung.html.twig'.
- Nach Klick auf Speichern Redirect auf 'modulCodeUebersicht'.

Wichtige Infos:

- Ein Modulcode muss aus genau 9 Zeichen bestehen.

mhbUebersichtAction()

Beschreibung:

Bei Aufruf dieser Funktion öffnet sich eine Tabelle, in der alle mit dem System erstellten Modulhandbücher aufgelistet sind. Der Nutzer hat durch Klick auf den Link „Download“ die Möglichkeit diese zu downloaden.

Aufruf über:

- @Route("/restricted/sgl/mhbUebersicht", name="mhbUebersicht")
- @Template("FHBingenMHBBundle:SGL:mhbUebersicht.html.twig")

Übergabeparameter: -

Ablauf:

- Alle Modulhandbücher werden in der Datenbank gesucht.
- Rendert 'mhbUebersicht.html.twig'

Wichtige Infos: -

modulAenderungenAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Tabelle, in der alle Module, die seit der letzten Erstellung eines Modulhandbuchs im Studiengang des eingeloggten Nutzers geändert worden sind, tabellarisch ausgegeben werden

.

Aufruf über:

- @Route("/restricted/sgl/modulAenderungen", name="modulAenderungen")
- @Template("FHBingenMHBBundle:SGL:modulAenderungen.html.twig")

Übergabeparameter: -

Ablauf:

- Aktueller Nutzer und der dazugehörige Studiengang werden ausgelesen.
- Danach wird das Erstellungsdatum des aktuellsten Modulhandbuchs des Studiengangs angefordert (($this->getNewestMHBDateForMyCourse()))
- Danach werden alle Veranstaltungen, deren Änderungsdatum neuer ist als das Erstellungsdatum des aktuellsten Modulhandbuchs des Studiengangs, gesucht und in ein Array geschrieben.
- Rendert 'modulAenderungen.html.twig'

Wichtige Infos:

- Sowohl die Änderungsdaten der Module als auch die Erstellungsdaten der Modulhandbücher werden mit Zeitstempel gespeichert.

pdfDownloadAction()

Beschreibung:

Diese Methode ermöglicht den Download bereits generierter Modulhandbücher.

Aufruf über:

- @Route("/restricted/sgl/pdfDownload/{mhbID}", name="pdfDownload")

Übergabeparameter:

- mhbID → ID des angeforderten Modulhandbuchs

Ablauf:

- In der Datenbank wird das Modulhandbuch zur übergebenen ID gesucht.
- Danach wird der Pfad des zugehörigen PDFs auf dem Server angefordert.
- Der Download des PDFs wird gestartet.

Wichtige Infos:

- Der Dateiname wird durch Aufruf der Methode $mhb->getMhbTitel() und zusätzliches anhängen der Endung '.pdf' erstellt.
- $mhb->getMhbTitel() leitet sich unter anderem vom Studiengangkürzel ab.
- **Sollte sich das Studiengangkürzel nach PDF-Erstellung ändern, funktioniert der Download nicht mehr!**

deaktivierungModuleAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Maske, in der sowohl die Module mit Status „expired“ als auch angebotene Module aufgelistet sind, unterschieden in 2 Tabellen. Zu den angebotenen Modulen werden zusätzlich noch alle Studiengänge angegeben.

Aufruf über:

- @Route("/restricted/sgl/deaktivierungAlleModule", name="deaktivierungAlleModule")
- @Template("FHBingenMHBBundle:SGL:modulDeaktivierung.html.twig")

Übergabeparameter: -

Ablauf:

- Aktueller Nutzer und der dazugehörige Studiengang werden ausgelesen.
- Aus den Angeboten werden alle Module die im eigenen Studiengang angeboten werden gesucht.
- In der Veranstaltungstabelle werden alle Module mit dem Status „expired“ gesucht.
- Zu den angebotenen Modulen werden alle Studiengänge in denen sie angeboten werden gesucht.
- Rendert 'modulDeaktivierung.html.twig'

Wichtige Infos: -

modulDeaktivierungAction()

Beschreibung:

Bei Aufruf dieser Methode wird der Status des angegebenen Moduls auf „expired“ und das Änderungsdatum auf das aktuelle Datum gesetzt. Zusätzlich werden alle Angebot-, Kernfach-, Lehrende- und Studienplan-Entities, die eine Referenz auf die Veranstaltung haben, gelöscht.

Aufruf über:

- @Route("/restricted/sgl/modulDeaktivierung/{modulID}", name="modulDeaktivierung")

Übergabeparameter:

- modulID → ID des Moduls, das deaktiviert werden soll

Ablauf:

- In der Veranstaltungstabelle wird das Modul zur übergebenen ID gesucht.
- Status des Moduls wird auf 'expired' gesetzt.
- Das Änderungsdatum des Moduls wird auf das aktuelle Datum gesetzt.
- Es werden alle Angebot-, Kernfach-, Lehrende- und Studienplan-Entities, die eine Referenz auf die Veranstaltung haben gelöscht.
- Redirect auf 'deaktivierungAlleModule'

Wichtige Infos: -

modulDeaktivierungStgAction()

Beschreibung:

Bei Aufruf dieser Methode wird das Angebot des angegbenen Moduls im angegebenen Studiengang gelöscht. Zusätzlich werden auch alle Kernfächer und Einträge im Studienplan gelöscht.

Aufruf über:

- Route("/restricted/sgl/modulDeaktivierungStg/{modulID}/{studiengangID}", name="modulDeaktivierungStg")

Übergabeparameter:

- modulID → ID des Moduls
- studiengangID → ID des Studiengangs für welchen das Angebot des Moduls gelöscht werden soll.

Ablauf:

- Löscht alle Angebot-Entities der Veranstaltung $modulID im Studiengang $studiengangID.
- Löscht alle Studienplan-Entities der Veranstaltung $modulID im Studiengang $studiengangID.
- Löscht alle Kernfach-Entities der Veranstaltung $modulID.

Wichtige Infos:

- Sofern das Modul nach der Deaktivierung noch in mindestens einem weiteren Studiengang angeboten wird, bleibt das Modul unberührt.
- Sollte jedoch der ausgewählte Studiengang der einzige sein, in dem das Modul angeboten wird und folglich danach kein weiteres Angebot mehr für das Modul bestehen, wird der Status des Moduls auf 'expired' gesetzt.

mhbErstellungAction()

Beschreibung:

Bei Aufruf dieser Methode erscheint eine Maske, in der der Nutzer eine Beschreibung zum zu generierenden Modulhandbuch angeben kann. Zudem muss der Nutzer auswählen, ab welchem Semester dieses Modulhandbuch gültig sein soll.

Aufruf über:

- @Route("/restricted/sgl/mhbErstellung", name="mhbErstellung")
- @Template("FHBingenMHBBundle:SGL:mhbErstellung.html.twig")

Übergabeparameter: -

Ablauf:

- Rendert 'mhbErstellung.html.twig'
- Nach Klick auf Speichern werden Formulardaten an 'mhbZusammenstellung' übergeben.

Wichtige Infos: -

mhbZusammenstellungAction()

Beschreibung:

Bei Aufruf dieser Methode erscheint eine Maske, in der der Nutzer per Drag and Drop festlegen muss, welche Module im neugenerierten Modulhandbuch erscheinen sollen. Standardmäßig werden alle Module mit Angebot im Studiengang des eingeloggten SGL in das Modulhandbuch eingetragen. Der Nutzer muss ein Modul, sofern dies nicht im Modulhandbuch erscheinen soll, in die nebenstehenden leeren Felder verschieben. Module, die seit der letzten Erstellung eines MHB im Studiengang geändert oder neu angelegt wurden, werden gehighlightet.

Aufruf über:

- @Route("/restricted/sgl/mhbZusammenstellung/{mhbGueltigAb}/{mhbBeschreibung}", name="mhbZusammenstellung")
- @Template("FHBingenMHBBundle:SGL:mhbZusammenstellung.html.twig")

Übergabeparameter:

- mhbGueltigab → Semester ab dem das Modulhandbuch gültig ist.
- mhbBeschreibung → angegeben Beschreibung des Modulhandbuchs.

Ablauf:

- Aktueller Nutzer und der dazugehörige Studiengang werden ausgelesen.
- Alle Angebote des Studiengangs werden in der Datenbank abgefragt.
- Die Angebote werden in die einzelnen Fachgebiete unterteilt.
- Das Erstelldatum des neuesten MHB im Studiengang wird abgefragt.
- Die Erstelldaten der Module werden mit dem Erstelldatum des neuesten MHB abgeglichen. Module die seit der letzten Erstellung eines MHB im Studiengang geändert oder neu angelegt wurden werden gehighlightet.
- Rendert 'mhbZusammenstellung.html.twig'

Wichtige Infos:

- Greift auf $this->getNewestMHBDateForMyCourse() zu
- Template macht beim Abschicken POST nach 'mhbErstellungParsen'

mhbErstellungParseAction()

Beschreibung:

Bei Aufruf dieser Methode wird geprüft ob der Nutzer bei der Modulhandbuch-Zusammenstellung mindestens ein Modul in das Modulhandbuch aufgenommen hat. Ist dies der Fall werden ModulhandbuchZuweisung-Entities erstellt und ein Modulhandbuch mit den ausgewählten Modulen erzeugt. Ansonsten wird eine Fehlermeldung ausgegeben.

Aufruf über:

- @Route("/restricted/sgl/mhbEstellungParsen", name="mhbErstellungParsen")

Übergabeparameter: -

Ablauf:

Sofern Angebote übergeben werden:

- Aktueller Nutzer und der dazugehörige Studiengang werden ausgelesen.
- Neues Modulhandbuch wird anhand übergeben Daten (POST) erstellt.
- Zugehörige ModulhandbuchZuweisung-Entities werden erstellt.
- Das generierte PDF wird auf self::MHB_PATH abgelegt ($this->pdfErstellenAction($mhb->getMHBID()))
- Redirect auf 'mhbUebersicht'

Sonst:

- Redirect auf 'mhbZusammenstellung' mit vorher eingetragenen Daten

Wichtige Infos:

- Greift auf $this->pdfErstellenAction($mhb->getMHBID($mhbID)) zu
- Wenn unerwartete Daten gepostet werden, kann es zu Fehlern kommen (bei Redirect auf 'mhbUebersicht').
- Wenn keine Daten gepostet werden, wird kein Template gerendert, sondern nur eine Fehler-Response ausgegeben.

getNewestMHBDateForMyCourse()

Beschreibung:

Bei Aufruf dieser Methode wird das Erstelldatum des neusten Modulhandbuchs des Studiengangs als DateTime zurückgegeben.

Aufruf über:

- Wird über Methoden mhbZusammenstellungAction() und modulAenderungenAction() aufgerufen.

Übergabeparameter: -

Ablauf:

- Aktueller Nutzer und der dazugehörige Studiengang werden ausgelesen.
- Holt Erstelldatum des neusten Modulhandbuchs des Studiengangs.
- Gibt das Datum als DateTime zurück.

Wichtige Infos: -

createModulBeschreibung()

Beschreibung:

Bei Aufruf dieser Methode wird zu jedem Modul, welches vorher dem neu zu generierenden PDF zugeordnet wurde, eine Modulhandbuchbeschreibung für den PDF-Export erstellt.

Aufruf über:

- Wird über Methode pdfErstellenAction() aufgerufen

Übergabeparameter:

- mhbID → ID des Modulhandbuchs

Ablauf:

- Das Modulhandbuch mit der übergebenen ID wird in der Datenbank abgefragt.
- Alle ModulhandbuchZuweisung-Entities des MHB werden in der Datenbank abgefragt.
- Generiert ein ModulBeschreibung-Objekt für jedes ModulhandbuchZuweisung-Entity
- Gibt alle ModulBeschreibung-Objekte als Array zurück.

Wichtige Infos:

- Greift auf SortFunctions::studienplanSort() zu.
- Greift auf SortFunctions::modulBeschreibungSort() zu.

pdfErstellenAction()

Beschreibung:

Bei Aufruf dieser Methode wird das neue Modulhandbuch als PDF generiert. Hierzu werden Header und Footer gesetzt, das Inhaltsverzeichnis erzeugt, sowie alle HTML Seiten in das PDF übernommen.

Aufruf über:

- Wird über Methode mhbErstellungParseAction() aufgerufen.

Übergabeparameter:

- mhbID → ID des Modulhandbuchs

Ablauf:

- Das Modulhandbuch mit der übergebenen ID wird in der Datenbank abgefragt.
- Setzt Header und Footer des PDF.
- Wenn nicht unter Windows ausgeführt: setze Optionen für Inhaltsverzeichnis
- Das generierte PDF wird auf self::MHB_PATH abgelegt

Wichtige Infos:

- Greift auf self::MHB_PATH zu.
- Greift auf self::WKHTMLTOPDF_BIN_LIN zu.
- Greift auf self::WKHTMLTOPDF_BIN_WIN zu.
- Unter Windows funktioniert die Generierung des Inhaltsverzeichnisses nicht, deswegen wird diese Option dort abgeschaltet.
- $mhb->getMhbTitel() leitet sich unter anderem vom Studiengangkürzel ab.
- **Sollte sich das Studiengangkürzel nach PDF-Erstellung ändern, funktioniert der Download nicht mehr!**

2.5. VerwaltungsController

SglShowUsersAction()

Beschreibung:

Bei Aufruf dieser Methode wird eine Maske geöffnet, in der alle in der Datenbank befindlichen Dozenten aufgelistet werden. Es wird jedoch unterschieden, ob diese aktiv oder inaktiv sind. Aktive Dozenten werden zusätzlich noch anhand der Rolle (Dozent oder SGL) unterschieden und in einer eigenen Spalte aufgelistet.

Aufruf über:

- @Route("/restricted/sgl/showUsers", name="benutzerVerwaltung")

Übergabeparameter: -

Ablauf:

- Sucht alle Nutzer aus der Datenbank. Sortiert diese nach Rolle und zeigt sie in der entsprechenden Spalte an.
- Sucht alle deaktivierten Nutzer aus der Datenbank und zeigt sie in der entsprechenden Spalte an.

Wichtige Infos: -

resetAction()

Beschreibung:

Bei Aufruf dieser Methode wird für alle in der Datenbank befindlichen Dozenten das Passwort auf das mit dem Kunden festgelegte Standard-Passwort gesetzt.

Aufruf über:

- @Route("//restricted/sgl/createUsers ", name="passwdReset")

Übergabeparameter: -

Ablauf:

- Setzt für jeden in der Datenbank befindlichen Nutzer das Passwort neu.

Wichtige Infos: -

SglCreateUserAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Maske, die es ermöglicht einen neuen Dozenten anzulegen, der danach Zugriff auf das System hat.

Aufruf über:

- @Route("//restricted/sgl/createUsers ", name=" benutzerErstellen ")
- @Template("FHBingenMHBBundle:Verwaltung:benutzerErstellen.html.twig")

Übergabeparameter: -

Ablauf:

- Lädt den DozentType().
- Über diese Maske wird dem Nutzer ermöglicht einen neuen User anzulegen.
- Rendert „benutzerErstellen.html.twig"

Wichtige Infos:

- Passwort eines neu angelegten Dozenten ist standardmäßig das vom Kunden gewünschte Standard-Passwort.

SglUpdateUserAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Maske, die es ermöglicht die Angaben zu einem Dozenten zu ändern. Das Passwort kann allerdings nicht geändert werden.

Aufruf über:

- @Route("/restricted/sgl/updateUsers/{userid}", name="benutzerBearbeiten")
- @Template("FHBingenMHBBundle:Verwaltung:benutzerErstellen.html.twig")

Übergabeparameter:

- userid → ID des ausgewählten Nutzers, der bearbeitet werden soll

Ablauf:

- Anhand der übergebenen ID wird der Nutzer in der Datenbank gesucht und die einzelnen Felder mit den in der Datenbank hinterlegten Werten gefüllt.
- Rendert „benutzerErstellen.html.twig“.

Wichtige Infos: -

SglShowAllCoursesAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Maske, in der alle in der Datenbank hinterlegten Studiengänge alphabetisch sortiert aufgelistet werden.

Aufruf über:

- @Route("/restricted/sgl/showAllCourses", name="studiengangVerwaltung")
- @Template("FHBingenMHBBundle:Verwaltung:alleStudiengaenge.html.twig")

Übergabeparameter: -

Ablauf:

- Zeigt alle Studiengänge die in der Datenbank hinterlegt sind.
- Rendert „alleStudiengaenge.html.twig"

Wichtige Infos:

- Studiengänge werden per SQL-Query ermittelt und sortiert.

SglCreateCourseAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Maske, die es ermöglicht einen neuen Studiengang anzulegen.

Aufruf über:

- @Route("/restricted/sgl/createCourse", name="studiengangErstellen")
- @Template("FHBingenMHBBundle:Verwaltung:studiengangAnzeigen.html.twig")

Übergabeparameter: -

Ablauf:

- Lädt den StudiengangType.
- Über diese Maske wird dem Nutzer ermöglicht einen neuen Studiengang anzulegen.
- Rendert „studiengangAnzeigen.html.twig".

Wichtige Infos: -

SglShowCourseAction()

Beschreibung:

Bei Aufruf dieser Methode öffnet sich eine Maske, die es ermöglicht die Angaben zu einem Studiengang zu ändern.

Aufruf über:

- @Route("/restricted/sgl/updateCourse/{courseID}", name="studiengangBearbeiten")
- @Template("FHBingenMHBBundle:Verwaltung:studiengangAnzeigen.html.twig")

Übergabeparameter:

- courseID → ID des ausgewählten Studiengangs der bearbeitet werden soll

Ablauf:

- Anhand der übergebenen ID wird der Studiengang in der Datenbank gesucht und die einzelnen Felder mit den entsprechend in der Datenbank hinterlegten Werten gefüllt.
- Rendert „studiengangAnzeigen.html.twig".

Wichtige Infos: -

SglUserDeactivation()

Beschreibung:

Bei Aufruf dieser Methode wird der angegebene Dozent deaktiviert und hat danach keinen Zugrif mehr auf das System.

Aufruf über:

- @Route("/restricted/sgl/SglUserDeactivation/{userid}", name="SglUserDeactivation")

Übergabeparameter:

- userid → ID des ausgewählten Nutzers der deaktiviert werden soll

Ablauf:

- Anhand der übergebenen ID wird der Nutzer in der Datenbank gesucht und dieser auf is_active = 0 (also false) gesetzt.

Wichtige Infos: -

SglUserActivation()

Beschreibung:

Bei Aufruf dieser Methode wird der angegebene Dozent reaktiviert und hat danach wieder Zugriff auf das System.

Aufruf über:

- @Route("/restricted/sgl/SglUserActivation/{userid}",name="SglUserActivation")

Übergabeparameter:

- userid → ID des ausgewählten Nutzers der Aktiviert werden soll

Ablauf:

- Anhand der übergebenen ID wird der Nutzer in der Datenbank gesucht und dieser auf is_active = 1 (also true) gesetzt.

Wichtige Infos: -

3. Entities

3.1. Klassendiagramm

Das Klassendiagramm der Entities finden Sie unter der Datei „Klassendiagramm.pdf“

3.2. Doctrine / Entity Erläuterungen

@ORM\Id
Dieses Datenfeld ist der Primärschlüssel des Entities. Dieser kann sich auch aus mehreren Datenfeldern zusammensetzen **(compound key).**

@ORM\Column(type="string", length=20, nullable=false)
Type gibt den Datentyp, length die maximale Länge des Feldes an. Wenn nullable=true angegeben wurde, muss nicht zwingend ein Datenwert in dieses Feld geschrieben werden. Bei nullable=false hingegen schon.

@ORM\GeneratedValue(strategy="AUTO")
Die ID dieses Entities ist ein Integer und wird automatisch hochgezählt. Der Nutzer hat keinen Einfluß auf die Vergabe der ID.

@ORM\ManyToOne(targetEntity="Veranstaltung", inversedBy="angebot")
Hierbei handelt es sich um einen Fremdschlüssel. ManyToOne oder ManyToMany geben die Kardinalität an. In TargetEntity wird festgelegt, in welchem Entity der Verweis liegt.

@ORM\JoinColumn(name="modul", referencedColumnName="Modul_ID", nullable=false)
Weiterhin wird für Fremdschlüssel festgelegt, wie die Spalte mit dem Fremdschlüssel in der eigenen Tabelle heißen soll (name).
referencedColumnName gibt die Spalte in der fremden Tabelle an, auf die referenziert wird.
Mit Nullable=false wird angegeben, dass auf jeden Fall ein Fremdschlüssel gesetzt werden muss.

ORM\OneToMany(targetEntity="ModulhandbuchZuweisung", mappedBy="angebot", cascade={"all"})
Ist der Verweis auf einen Fremdschlüssel. OneToMany oder ManyToMany geben die Kardinalität an.
TargetEntity gibt das Entity an, in der das eigene Entity als Fremdschlüssel verwendet wird.
MappedBy gibt das Datenfeld an, in welchem der Fremdschlüssel in dem fremden Entity festgehalten wird.

```
@Assert\Choice(
    choices = { "Wahlpflichtfach", "Pflichtfach" },
    message = "Bitte geben Sie eine korrekte Angebotsart an!"
 )
```

Prüft, ob die Eingabe den angegebenen Auswahlmöglichkeiten entspricht.
Ansonsten wird die gegebene Fehlermeldung ausgegeben.

```
@Assert\Length(
        min = 8,
        minMessage = "Der Modulcode muss mindestens {{ limit }} Zeichen lang sein.",
        max = 9,
        maxMessage = "Der Modulcode darf maximal {{ limit }} Zeichen lang sein."
)
```

Prüft, ob die gegebene Eingabe der angegebenen Länge entspricht und gibt ansonsten die entsprechende Fehlermeldung aus.

```
@Assert\Regex(
        pattern = "/[BM]\-[A-Z]{2,2}\-[A-Z]{1,2}[0-9]{2,2}/",
        message = "Bitte verwenden Sie folgendes Muster für den Modulcode: z.B. B-IN-MN01, B-IN-V05"
)
```

Prüft, ob die Eingabe einem bestimmten Muster entspricht und gibt ansonsten die Fehlermeldung aus.

```
@Assert\NotBlank(
        message="Der Vorname darf nicht leer sein."
)
```

Prüft, ob ein Feld gefüllt wurde und gibt ansonsten die gegebene Fehlermeldung aus.

```
@Assert\Email(
        message = "Die Email '{{ value }}' ist keine gueltige Email."
)
```

Prüft, ob eine Angabe den gültigen Werten einer Email entspricht und gibt ansonsten die gegebene Fehlermeldung aus.

3.3. Angebot

Ein Angebot beschreibt ein Modulangebot in einem Studiengang.
Ein Modul kann in mehreren Studiengängen angeboten werden.
Hierzu gibt es zu jedem Angebot einen einzigartigen Modulcode, der Abhängig vom gewähltem Studiengang und Fachgebiet ist.
Weiterhin wird im Angebot festgelegt, ob das Modul im gewählten Studiengang ein Wahl- oder Pflichtfach ist. Zusätzlich gibt es die Möglichkeit den Modulen studiengangspezifische deutsche und englische Titel zu geben, dies ist aber optional.

Datenfelder

- **$Angebots_ID**
  - **@ORM\Id**
  - **@ORM\Column**(type="integer")
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$veranstaltung**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung", inversedBy="angebot")
  - **@ORM\JoinColumn**(name="modul", referencedColumnName="Modul_ID", nullable=false)

- **$fachgebiet**
  - **@ORM\ManyToOne**(targetEntity="Fachgebiet", inversedBy="angebot")
  - **@ORM\JoinColumn**(name="fachgebiet", referencedColumnName="Fachgebiets_ID", nullable=false)

- **$studiengang**
  - **@ORM\ManyToOne**(targetEntity="Studiengang", inversedBy="angebot")
  - **@ORM\JoinColumn**(name="studiengang", referencedColumnName="Studiengang_ID", nullable=false)

- **$Angebotsart**
  - **@ORM\Column**(type="string", length=20, nullable=false)
  - **@Assert\Choice**(
    choices = { "Wahlpflichtfach", "Pflichtfach" },
    message = "Bitte geben Sie eine korrekte Angebotsart an!"
)

- **$Code**
  - **@ORM\Column**(type="string", length=20, nullable=false)
  - **@Assert\Length**(
    min = 8,
    minMessage = "Der Modulcode muss mindestens {{ limit }} Zeichen lang sein.",
    max = 9,
    maxMessage = "Der Modulcode darf maximal {{ limit }} Zeichen lang sein."
)
  - **@Assert\Regex**(
    pattern = "/[BM]\-[A-Z]{2,2}\-[A-Z]{1,2}[0-9]{2,2}/",
    message = "Bitte verwenden Sie folgendes Muster für den Modulcode: z.B. B-IN-MN01, B-IN-V05"
)

- **$AbweichenderNameDE**
  - **@ORM\Column**(type="string", length=70, nullable=true, unique=true)
  - **@Assert\Length**(
    min = 5,
    minMessage = "Der abweichende deutsche Name muss mindestens {{ limit }} Zeichen lang sein.",
    max = 70,
    maxMessage = "Der abweichende deutsche Name darf maximal {{ limit }} Zeichen lang sein."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß0-9 \-]{5,70}/",
    message = "Der studiengangspezifische deutsche Name darf nur aus Buchstaben, Zahlen, Leerzeichen und Bindestrichen bestehen."
)

- **$AbweichenderNameEN**
  - **@ORM\Column**(type="string", length=70, nullable=true, unique=true)
  - **@Assert\Length**(
    min = 5,
    minMessage = "Der abweichende englische Name muss mindestens {{ limit }} Zeichen lang sein.",
    max = 70,
    maxMessage = "Der abweichende englische Name darf maximal {{ limit }} Zeichen lang sein."
)

  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß0-9 \-]{5,70}/",
    message = "Der studiengangspezifische englische Name darf nur aus Buchstaben, Zahlen, Leerzeichen und Bindestrichen bestehen."
)

- **$zuweisung**
  - **@ORM\OneToMany**(targetEntity="ModulhandbuchZuweisung", mappedBy="angebot", cascade={"all"})

Getter und Setter

- __toString() → null | string
- __construct() → Angebot
- getAngebotsID() → int
- setVeranstaltung(Veranstaltung $module = null) → Angebot
- getVeranstaltung() → Veranstaltung
- setFachgebiet(Fachgebiet $fachgebiet = null) → Angebot
- getFachgebiet() → Fachgebiet
- setStudiengang(Studiengang $studiengang = null) → Angebot
- getStudiengang() → Studiengang
- setAngebotsart(string $angebotsart) → Angebot
- getAngebotsart() → string
- setCode(string $code) → Angebot
- getCode() → string
- setAbweichenderNameDE(string $abweichenderNameDE) → Angebot
- getAbweichenderNameDE() → string
- setAbweichenderNameEN(string $abweichenderNameEN) → Angebot
- getAbweichenderNameEN() → string
- addZuweisung(ModulhandbuchZuweisung $zuweisung) → Angebot
- removeZuweisung(ModulhandbuchZuweisung $zuweisung) → Angebot
- getZuweisung() → Collection

3.4. Dozent

Das Entity Dozent bildet einen Dozenten ab. Hierzu sind Angaben zu Geschlecht, Titel, Name und Nachname, sowie eine einzigartige Email nötig.
Weiterhin kann zwischen Dozenten und Studiengangleiter unterschieden werden.
Passwörter werden automatisch je nach Rolle des Dozenten gesetzt.

Datenfelder

- **$Dozenten_ID**
  - **@ORM\Id**
  - **@ORM\Column**(type="integer")
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Anrede**
  - **@ORM\Column**(type="string", length=50, nullable=true)
  - **@Assert\Length**(
    max = 50,
    maxMessage = "Der Titel darf nicht länger als {{ limit }} Zeichen sein."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß. ()\-]{1,50}/",
    message = "Der Titel darf nur aus Buchstaben, Leerzeichen, Bindestrichen, Klammern und Punkten bestehen."
)

- **$Titel**
  - **@ORM\Column**(type="string", length=50, nullable=true)
  - **@Assert\Length**(
    max = 50,
    maxMessage = "Der Titel darf nicht länger als {{ limit }} Zeichen sein."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß. ()\-]{1,50}/",
    message = "Der Titel darf nur aus Buchstaben, Leerzeichen, Bindestrichen, Klammern und Punkten bestehen."
)

- **$Name**
  - **@ORM\Column**(type="string", length=20, nullable=false)
  - **@Assert\Length**(
    max = 20,
    maxMessage = "Der Vorname darf nicht länger als {{ limit }} Zeichen sein."
)
  - **@Assert\NotBlank**(
    message="Der Vorname darf nicht leer sein."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß \-]{1,20}/",
    message = "Der Vorname darf nur aus Buchstaben, Leerzeichen und Bindestrichen bestehen."
)

- **$Nachname**
  - **@ORM\Column**(type="string", length=30, nullable=false)
  - **@Assert\Length**(
    max = 30,
    maxMessage = "Der Nachname darf nicht länger als {{ limit }} Zeichen sein."
)
  - **@Assert\NotBlank**(
    message="Der Nachname darf nicht leer sein."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß \-]{1,30}/",
    message = "Der Nachname darf nur aus Buchstaben, Leerzeichen und Bindestrichen bestehen."
)

- **$lehrende**
  - **@ORM\OneToMany**(targetEntity="Lehrende", mappedBy="dozent", cascade={"all"})

- **$semesterplan**
  - **@ORM\OneToMany**(targetEntity="Semesterplan", mappedBy="dozent", cascade={"all"})

- **$modulbeauftragter**
  - **@ORM\OneToMany**(targetEntity="Veranstaltung", mappedBy="beauftragter")

- **$studiengang**
  - @**ORM\OneToMany**(targetEntity="Studiengang", mappedBy="sgl")

- **$email**
  - @**ORM\Column**(type="string", length=60, unique=true, nullable=false)
  - @**Assert\NotBlank**(
    message="Die eMail-Adresse darf nicht leer sein."
)
  - @**Assert\Email**(
    message = "Die Email '{{ value }}' ist keine gueltige Email."
)
  - @**Assert\Length**(
    max = 60,
    maxMessage = "Die eMail-Adresse darf nicht länger als {{ limit }} Zeichen sein."
)

- **$password**
  - @**ORM\Column**(type="string", length=64, nullable=false)

- **$isActive**
  - @**ORM\Column**(name="is_active", type="boolean", nullable=false)

- **$roles**
  - @**ORM\ManyToOne**(targetEntity="Role", inversedBy="users")
  - @**ORM\JoinColumn**(name="roles_id", referencedColumnName="id", nullable=false)

Getter und Setter

- __toString() → string
- __construct() → Dozent
- getDozentenID() → integer
- setAnrede(string $anrede) → Dozent
- getAnrede() → string
- setTitel(string $titel) → Dozent
- getTitel() → string
- setName(string $name) → Dozent
- getName() → string
- setNachname(string $nachname) → Dozent
- getNachname() → string
- setEmail(string $email) → Dozent
- getEmail() → string

- addLehrende(Lehrende $lehrende) → Dozent
- removeLehrende(Lehrende $lehrende) → Dozent
- getLehrende() → Collection
- addSemesterplan(Semesterplan $semesterplan) → Dozent
- removeSemesterplan(Semesterplan $semesterplan) → Dozent
- getSemesterplan() → Collection
- addModulbeauftragter(Veranstaltung $veranstaltung) → Dozent
- removeModulbeauftragter(Veranstaltung $veranstaltung) → Dozent
- getModulbeauftragter() → Collection
- addStudiengang(Studiengang $studiengang) → Dozent
- removeStudiengang(Studiengang $studiengang) → Dozent
- getStudiengang() → Collection
- setRole(RoleInterface $role) → $this
- setRoles(RoleInterface $role) → Dozent
- getUsername() → string
- getSalt() → null | string
- getPassword() → string
- getRoles() → array | Role
- eraseCredentials() → void
- serialize() → string
- unserialize(string $serialized) → void
- setUsername(string $username) → Dozent
- setPassword(string $password) → Dozent
- setIsActive(bool $isActive) → Dozent
- getIsActive() → bool
- getEncoderName() → string
- isAccountNonExpired() → bool
- isAccountNonLocked() → bool
- isCredentialsNonExpired() → bool
- isEnabled() → bool

3.5. Fachgebiet

Das Entity Fachgebiet bildet ein Fachgebiet mit einem Fachgebietstitel zu einem dazu gehörigem Studiengang ab.

Datenfelder

- **$Fachgebiets_ID**
  - **@ORM\Column**(type="integer")
  - **@ORM\ID**
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Titel**
  - **@ORM\Column**(type="string", length=50, nullable=false)
  - **@Assert\NotBlank**(
    message = "Der Fachgebietstitel darf nicht leer sein."
)
  - **@Assert\Length**(
    min = 4,
    minMessage = "Der Fachgebietstitel muss aus mindestens {{ limit }} Zeichen bestehen.",
    max = 50,
    maxMessage = "Der Fachgebietstitel darf aus maximal {{ limit }} Zeichen bestehen."
)

  - **@Assert\Regex(**
    pattern = "/[A-ZÄÖÜa-zäöüß \-]{4,50}/",
    message = "Der Fachgebietstitel darf nur aus Buchstaben, Leerzeichen und Bindestrichen bestehen."
)

- **$angebot**
  - **@ORM\OneToMany**(targetEntity="Angebot", mappedBy="fachgebiet", cascade={"all"})

- **$studiengang**
  - **@ORM\ManyToOne**(targetEntity="Studiengang",
    inversedBy="fachgebiete")
  - **@ORM\JoinColumn**(name="studiengang",
    referencedColumnName="Studiengang_ID", nullable=false)

Getter und Setter

- __toString() → string
- __construct() → Fachgebiet
- getFachgebietsID() → integer
- setTitel(string $titel) → Fachgebiet
- getTitel() → string
- addAngebot(Angebot $angebot) → Fachgebiet
- removeAngebot(Angebot $angebot) → Fachgebiet
- getAngebot() → Collection
- setStudiengang(Studiengang $studiengang = null) → Fachgebiet
- getStudiengang() → Studiengang

3.6. Kernfach

Ein Kernfach ist eine Zuordnung von Modulen zu einer gegebenen Vertiefungsrichtung.

Datenfelder

- **$Kernfach_ID**
  - **@ORM\Id**
  - **@ORM\Column**(type="integer")
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$veranstaltung**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung", inversedBy="kernfach")
  - **@ORM\JoinColumn**(name="modul", referencedColumnName="Modul_ID", nullable=false)

- **$vertiefung**
  - **@ORM\ManyToOne**(targetEntity="Vertiefung", inversedBy="vertiefung")
  - **@ORM\JoinColumn**(name="vertiefung", referencedColumnName="Vertiefung_ID", nullable=false)

Getter und Setter

- getKernfach_ID() → integer
- setVeranstaltung(Veranstaltung $modul = null) → Kernfach
- getVeranstaltung() → Veranstaltung
- setVertiefung(Vertiefung $vertiefung = null) → Kernfach
- getVertiefung() → Vertiefung

3.7. Lehrende

Im Entity Lehrende werden einer Veranstaltung lehrende Dozenten zugeordnet.

Datenfelder

- **$veranstaltung**
  - **@ORM\Id**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung", inversedBy="lehrende")
  - **@ORM\JoinColumn**(name="modul", referencedColumnName="Modul_ID", nullable=false)

- **$dozent**
  - **@ORM\Id**
  - **@ORM\ManyToOne**(targetEntity="Dozent", inversedBy="lehrende")
  - **@ORM\JoinColumn**(name="dozent", referencedColumnName="Dozenten_ID", nullable=false)

Getter und Setter

- __toString() → string
- setVeranstaltung(Veranstaltung $modul) → Lehrende
- getVeranstaltung() → Veranstaltung
- setDozent(Dozent $lehrender) → Lehrende
- getDozent() → Dozent

3.8. Modulhandbuch

Im Entity Modulhandbuch wird ein Modulhandbuch abgebildet. Hierzu wird automatisch der Studiengang des eingeloggten Nutzers verwendet. Des Weiteren muss der Nutzer eine Beschreibung des Modulhandbuches eingeben und angeben, ab welchem Semester das Modulhandbuch gültig sein soll. Im Hintergrund werden zusätzlich eine zum Studiengang gehörige Versionsnummer sowie das Erstellungsdatum gespeichert.

Datenfelder

- **$MHB_ID**
  - @**ORM**Column(type="integer")
  - @**ORM\ID**
  - @**ORM\GeneratedValue**(strategy="AUTO")

- **$Versionsnummer**
  - @**ORM\Column**(type="integer")
  - @**ORM\GeneratedValue**(strategy="AUTO")

- **$Erstellungsdatum**
  - @**ORM\Column**(type="date", nullable=false)
  - @**Assert\Date**()

- **$Beschreibung**
  - @**ORM\Column**(type="text", nullable=false)
  - @**Assert\NotBlank(**message = "Die Modulhandbuchbeschreibung darf nicht leer sein.")

- **$gueltigAb**
  - @**ORM\ManyToOne**(targetEntity="Semester", inversedBy="gueltigAbSemester")
  - @**ORM\JoinColumn**(name="gueltigAb", referencedColumnName="Semester", nullable=false)

- **$gehoertZu**
  - @**ORM\ManyToOne**(targetEntity="Studiengang", inversedBy="studiengang")
  - @**ORM\JoinColumn**(name="gehoertZu", referencedColumnName="Studiengang_ID", nullable=false)

- **$zuweisung**
  - @**ORM\OneToMany**(targetEntity="ModulhandbuchZuweisung" , mappedBy="mhb" , cascade={"all"})

Getter und Setter

- __toString() → string
- getMhbTitel() → string
- __construct() → Modulhandbuch
- getMHBID() → integer
- setVersionsnummer(integer $mHBVersionsnummer) → Modulhandbuch
- getVersionsnummer() → integer
- setErstellungsdatum(DateTime $erstellungsdatum) → Modulhandbuch
- getErstellungsdatum() → DateTime
- setBeschreibung(string $beschreibung) → string
- getBeschreibung() → string
- setGueltigAb(Semester $gueltigAb = null) → Modulhandbuch
- getGueltigAb() → Semester
- setGehoertZu(Studiengang $gehoertZu = null) → Modulhandbuch
- getGehoertZu() → Studiengang
- addZuweisung(ModulhandbuchZuweisung $zuweisung)→ Modulhandbuch
- removeZuweisung(ModulhandbuchZuweisung $zuweisung)→ Modulhandbuch
- getZuweisung() → Collection

3.9. ModulhandbuchZuweisung

Im Entity ModulhandbuchZuweisung werden Angebote den Modulhandbüchern, in denen sie vorkommen, zu geordnet. Dies kann nicht vom Nutzer bearbeitet werden.

Datenfelder

- **$mhb**
  - **@ORM\Id**
  - @**ORM\ManyToOne**(targetEntity="Modulhandbuch", inversedBy="zuweisung")
  - @**ORM\JoinColumn**(name="mhb", referencedColumnName="MHB_ID")

- **$angebot**
  - **@ORM\Id**
  - @**ORM\ManyToOne**(targetEntity="Angebot", inversedBy="zuweisung")
  - @**ORM\JoinColumn**(name="angebot", referencedColumnName="Angebots_ID")

Getter und Setter

- setMhb(Modulhandbuch $mhb) → ModulhandbuchZuweisung
- getMhb() → Modulhandbuch
- setAngebot(Angebot $angebot) → ModulhandbuchZuweisung
- getAngebot() → Angebot

3.10. Modulvoraussetzung

Im Entity Modulvoraussetzung werden die formalen Voraussetzungen zu einem Modul gespeichert. Module für die alle Module des Studienganges vorausgesetzt sind (wie die Bachelor- und Master Thesis) können nicht abgebildet werden.

Datenfelder

- **$modul**
  - **@ORM\Id**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung", inversedBy="grundmodul")
  - **@ORM\JoinColumn**(name="modul", referencedColumnName="Modul_ID")

- **$voraussetzung**
  - **@ORM\Id**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung", inversedBy="forderung")
  - **@ORM\JoinColumn**(name="voraussetzung", referencedColumnName="Modul_ID")

Getter und Setter

- __toString() → string
- setModul(Veranstaltung $modul) → Modulvoraussetzung
- getModul() → Veranstaltung
- setVoraussetzung(Veranstaltung $voraussetzung) → Modulvoraussetzung
- getVoraussetzung() → Veranstaltung

3.11. Role

Role ist eine von Symfony vorgegebene Klasse zur Authentifizierung an der internen Firewall. In unserem System werden zwei Rollen festgelegt um Nutzer zu unterscheiden. Dozenten sind der Rolle „ROLE_DOZENT“ zugewiesen, Studiengangleiter der Rolle „ROLE_SGL“.

Datenfelder

- **$id**
  - @**ORM\Column**(name="id", type="integer")
  - @**ORM\Id()**
  - @**ORM\GeneratedValue**(strategy="AUTO")

- **$name**
  - @**ORM\Column**(name="name", type="string", length=30)

- **$role**
  - @**ORM\Column**(name="role", type="string", length=20, unique=true)

- **$users**
  - @**ORM\OneToMany**(targetEntity="Dozent", mappedBy="roles")

Getter und Setter

- __toString() → string
- __construct() → Role
- getRole() → null | string
- getId() → integer
- setName(string $name) → Role
- getName() → string
- setRole(string $role) → Role
- addUser(UserInterface $users) → Role
- removeUser(UserInterface $users) → Role
- getUsers() → Collection

3.12. Semester

Im Entity Semester werden die Semester nach dem Schema „SS13“ bzw. „WS14“ vorgehalten.

Datenfelder

- **$Semester**
  - **@ORM\ID**
  - **@ORM\Column**(type="string", length=4, nullable=false, unique=true)
  - **@Assert\NotBlank**(message = "Die Bezeichnung des Semesters darf nicht leer sein.")
  - **@Assert\Length**(
    min = 4,
    max = 4,
    exactMessage = "Der Semestername muss aus genau {{ limit }} Zeichen bestehen."
)
  - **@Assert\Regex**(
    pattern = "/[SW][S][0-9]{2,2}/",
    message = "Bitte verwenden Sie folgendes Muster für den Semesternamen: z.B. SS01, WS17"
)

- **$semesterplan**
  - **@ORM\OneToMany**(targetEntity="Semesterplan", mappedBy="semester", cascade={"all"})

- **$gueltigAbSemester**
  - **@ORM\OneToMany**(targetEntity="Modulhandbuch", mappedBy="gueltigAb")

Getter und Setter

- __toString() → string
- __construct() → Semester
- setSemester($semester) → Semester
- getSemester() → string
- addSemesterplan(Semesterplan $semesterplan) → Semester
- removeSemesterplan(Semesterplan $semesterplan) → Semester
- getSemesterplan() → Collection
- addGueltigAbSemester(Modulhandbuch $sem) → Semester
- removeSem(Modulhandbuch $sem) → Semester

- getGueltigAbSemester() → Collection
- addModulStart(Veranstaltung $modulStart) → Semester
- removeModulStart(Veranstaltung $modulStart) → Semester
- getModulStart() → Collection
- addRegelsem(Studienplan $regelsem) → Semester
- removeRegelSemester(Studienplan $regelsem) → Semester
- getRegelSemester() → Collection

3.13. Semesterplan

Im Entity-Semesterplan wird die Planung zu einem Semester gespeichert. Hierzu werden zu den im ausgewählten Semester angebotenen Modulen die SWS in Übung und Vorlesung unterteilt angegeben, sowie die Anzahl und Gruppengröße der Übungsgruppen. Außerdem kann angegeben werden, welche Dozenten ein bestimmtes Modul in diesem Semester unterrichten.

Datenfelder

- **$Semesterplan_ID**
  - **@ORM\Id**
  - **@ORM\Column**(type="integer")
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$SWSUebung**
  - **@ORM\Column**(type="integer")
  - **@Assert\Range**(
    min = 0,
    max = 10,
    minMessage = "Ein Modul braucht mindestens {{ limit }} SWS Übung",
    maxMessage = "Ein Modul darf nicht mehr als {{ limit }} SWS Übung haben"
)

- **$SWSVorlesung**
  - **@ORM\Column**(type="integer")
  - **@Assert\Range**(
    min = 0,
    max = 10,
    minMessage = "Ein Modul braucht mindestens {{ limit }} SWS Vorlesung",
    maxMessage = "Ein Modul darf nicht mehr als {{ limit }} SWS Vorlesung haben"
)

- **$AnzahlUebungsgruppen**
  - **@ORM\Column**(type="integer")
  - **@Assert\Range**(
    min = 0,
    max = 5,
    minMessage = "Ein Modul braucht mindestens {{ limit }} Übungsgruppen",
    maxMessage = "Ein Modul darf nicht mehr als {{ limit }} Übungsgruppen haben"
)

- **$GroesseUebungsgruppen**
  - **@ORM\Column**(type="integer")
  - **@Assert\Range**(
    min = 0,
    max = 30,
    minMessage = "Eine Übungsgruppe muss aus mindestens {{ limit }} Studenten bestehen",
    maxMessage = "Eine Übungsgruppe darf aus nicht mehr als {{ limit }} Studenten bestehen"
)

- **$veranstaltung**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung", inversedBy="semesterplan")
  - **@ORM\JoinColumn**(name="modul", referencedColumnName="Modul_ID", nullable=false)

- **$dozent**
  - **@ORM\ManyToOne**(targetEntity="Dozent", inversedBy="semesterplan")
  - **@ORM\JoinColumn**(name="dozent", referencedColumnName="Dozenten_ID", nullable=false)

- **$semester**
  - **@ORM\ManyToOne**(targetEntity="Semester", inversedBy="semesterplan")
  - **@ORM\JoinColumn**(name="semester", referencedColumnName="Semester", nullable=false)

Getter und Setter

- __toString() → string
- getSemesterplan_ID() → integer
- setSWSUebung(integer $swsUebung) → Semesterplan
- getSWSUebung() → integer
- setSWSVorlesung(integer $swsVorlesung) → Semesterplan
- getSWSVorlesung() → integer
- setAnzahlUebungsgruppen(integer $anzahlUebungsgruppen) → Semesterplan
- getAnzahlUebungsgruppen() → integer
- setGroesseUebungsgruppen(integer $groesse) → Semesterplan
- getGroesseUebungsgruppen() → integer
- setVeranstaltung(Veranstaltung $module = null) → Semesterplan
- getVeranstaltung() → Veranstaltung
- setDozent(Dozent $lehrender = null) → Semesterplan
- getDozent() → Dozent
- setSemester(Semester $semester = null) → Semesterplan
- getSemester() → Semester

3.14. Studiengang

Das Studiengang-Entity bildet einen Studiengang ab. Hierzu sind Angaben zum Fachbereich, Bildungsgrad des Studienganges, ein einzigartiger Studiengangtitel und -Kürzel und eine Beschreibung des Studiengangs nötig. Zusätzlich muss auch ein Studiengangleiter für einen Studiengang gewählt werden.

Datenfelder

- **$Studiengang_ID**
  - **@ORM\Column**(type="integer")
  - **@ORM\ID**
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Fachbereich**
  - **@ORM\Column**(type="integer", nullable=false)
  - **@Assert\Range**(
    min = 1,
    max = 2,
    minMessage = "Fachbereich {{ limit }} ist Minimum",
    maxMessage = "Fachbereich {{ limit }} ist Maximum"
)

- **$Grad**
  - @**ORM\Column**(type="string", length=15, nullable=false)
  - @**Assert\Choice**(choices = {"Bachelor", "Master"}, message = "Wählen Sie einen gültigen Bildungsgrad")

- **$Titel**
  - @**ORM\Column**(type="string", length=40, nullable=false, unique=true)
  - @**Assert\NotBlank**(message = "Der Studiengang-Titel darf nicht leer sein.")
  - @**Assert\Length**(
    min = 2,
    minMessage = "Ein Studiengang-Titel muss aus mindestens {{ limit }} Zeichen bestehen.",
    max = 40,
    maxMessage = "Ein Studiengang-Titel darf aus maximal {{ limit }} Zeichen bestehen."
)
  - @**Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß \-]{2,40}/",
    message = "Der Studiengang-Titel darf nur aus Buchstaben, Leerzeichen und Bindestrichen bestehen."
)

- **$Kuerzel**
  - @**ORM\Column**(type="string", length=5, nullable=false, unique=true)
  - @**Assert\NotBlank**(message = "Das Studiengang-Kürzel darf nicht leer sein.")
  - @**Assert\Length**(
    min = 2,
    max = 5,
    minMessage = "Ein Studiengang-Kürzel muss aus mindestens {{ limit }} Zeichen bestehen.",
    maxMessage = "Ein Studiengang-Kürzel darf aus maximal {{ limit }} Zeichen bestehen."
)
  - @**Assert\Regex**(
    pattern = "/[BM]\-[A-Z]{2,2}/",
    message = "Bitte verwenden Sie folgendes Muster für das Studiengangkürzel: z.B. B-IN, M-IS"
)

- **$Beschreibung**
  - @**ORM\Column**(type="text", nullable=false)
  - @**Assert\NotBlank**(message = "Die Studiengang-Beschreibung darf nicht leer sein.")

- **$angebot**
  - @**ORM\OneToMany**(targetEntity="Angebot", mappedBy="studiengang", cascade={"all"})

- **$richtung**
  - @**ORM\OneToMany**(targetEntity="Vertiefung", mappedBy="studiengang")

- **$sgl**
  - @**ORM\ManyToOne**(targetEntity="Dozent", inversedBy="studiengang")
  - @**ORM\JoinColumn**(name="sgl", referencedColumnName="Dozenten_ID", nullable=false, unique=true)

- **$studiengang**
  - @**ORM\OneToMany**(targetEntity="Modulhandbuch", mappedBy="gehoertZu")

- **$fachgebiete**
  - @**ORM\OneToMany**(targetEntity="Fachgebiet", mappedBy="studiengang")

- **$studienplanZuStudienplan**
  - @**ORM\OneToMany**(targetEntity="Studienplan", mappedBy="studiengang", cascade={"all"})

Getter und Setter

- __toString() → string
- __construct() → Studiengang
- getStudiengangID() → integer
- setFachbereich(integer $fachbereich) → Studiengang
- getFachbereich() → integer
- setGrad(string $grad) → Studiengang
- getGrad() → string
- setTitel(string $titel) → Studiengang
- getTitel() → string
- setKuerzel(string $kuerzel) → Studiengang
- getKuerzel() → string
- setBeschreibung(string $beschreibung) → Studiengang
- getBeschreibung() → string

- addAngebot(Angebot $angebot) → Studiengang
- removeAngebot(Angebot $angebot) → Studiengang
- getAngebot() → Collection
- addRichtung(Vertiefung $richtung) → Studiengang
- removeRichtung(Vertiefung $richtung) → Studiengang
- getRichtung() → Collection
- setSgl(Dozent $sgl) → Studiengang
- getSgl() → Dozent
- addStudiengang(Modulhandbuch $studiengang) → Studiengang
- removeStudiengang(Modulhandbuch $studiengang) → Studiengang
- getStudiengang() → Collection
- addFachgebiete(Fachgebiet $fachgebiet) → Studiengang
- removeFachgebiete(Fachgebiet $fachgebiet) → Studiengang
- getFachgebiete() → Collection
- addStudienplanZuStudienplan(Studienplan $studienplanStgang) → Studiengang
- removeStudienplanZuStudienplan(Studienplan $studienplanStgang) → Studiengang
- getStudienplanZuStudienplan() → Collection

3.15. Studienplan

Im Studienplan-Entity werden die Regelsemester der Module in den verschiedenen Studiengängen angegeben und zwischen Start im Sommer- bzw. Wintersemester unterschieden.

Datenfelder

- **$Studienplan_ID**
  - **@ORM\Id**
  - **@ORM\Column**(type="integer")
  - **@ORM\GeneratedValue**(strategy="AUTO")
  - **@var integer $id**

- **$Regelsemester**
  - **@ORM\Column**(type="text"), nullable=false
  - **@Assert\NotBlank**(message = "Die Regelsemester müssen gesetzt werden.")

- **$Startsemester**
  - **@ORM\Column**(type="string"), nullable = false
  - **@Assert\Choice**(
    choices = { "SS", "WS" },
    message = "Bitte geben Sie ein korrektes Startsemester!"
)

- **$veranstaltung**
  - **@ORM\ManyToOne**(targetEntity="Veranstaltung",
    inversedBy="studienplanModul")

  - **@ORM\JoinColumn**(name="modul", referencedColumnName="Modul_ID",
    nullable=false)

- **$studiengang**
  - **@ORM\ManyToOne**(targetEntity="Studiengang",
    inversedBy="studienplanZuStudienplan")
  - **@ORM\JoinColumn**(name="studiengang",
    referencedColumnName="Studiengang_ID", nullable=false)

Getter und Setter

- __toString() → string
- getStudienplan_ID() → integer
- setRegelSemester(string $regSem) → Studienplan
- getRegelSemester() → string
- setStartsemester(string $startSem) → Studienplan
- getStartsemester() → string
- setVeranstaltung(Veranstaltung $modul = null) → Studienplan
- getVeranstaltung() → Veranstaltung
- setStudiengang(Studiengang $studiengang = null) → Studiengang
- getStudiengang() → Studiengang

3.16. Veranstaltung

Im Veranstaltung-Entity werden die Informationen zu einer Lehrveranstaltung abgebildet. Das Erstellungsdatum, die Versionsnummer, der Autor, sowie die Zeit des Selbststudiums werden systemintern gesetzt und können nicht vom Nutzer beeinflusst werden. Der Status kann nur bedingt vom Nutzer beeinflusst werden. Legt der Nutzer eine Planung an, hat diese den Status „in Planung“. Wird diese vom Nutzer freigegeben wechselt der Status auf „Freigegeben“. Freigegebene Veranstaltungen können nicht wieder auf den Status „in Planung“ zurückgesetzt werden. Studiengangleiter haben noch die Möglichkeit Veranstaltungen zu deaktivieren. Hierbei wird der Status auf „expired“ gesetzt.
Der Nutzer kann für eine Veranstaltung Informationen zu einer Veranstaltung auf folgende Felder verteilen: Kürzel, deutscher Name, englischer Name, Häufigkeit (Wie oft wird das Modul angeboten?), Dauer (Wie lange dauert ein Modul?), mögliche Lehrveranstaltungen, Kontaktzeiten unterteilt in Vorlesung und sonstige, Gruppengröße, Lehrinhalte, Lernergebnisse, Prüfungsformen, Leistungspunkte, inhaltliche Voraussetzungen, Sprache (Unterscheidung deutsch und englisch, zusätzlich noch weitere eigene Angaben möglich), Literatur, Voraussetzungen zur Vergabe von Leistungspunkten.
Da für Planungen nur der deutsche Name verpflichtend ist, sind alle weiteren Felder die der Nutzer ändern kann nicht verpflichtend. Sobald eine Planung aber freigegeben wird, bzw. bereits freigegebenen wurde, sind alle Felder verpflichtend. Ausnahmen sind lediglich weitere Angaben zur Sprache, sonstige/weitere Prüfungsformen, Erläuterungen zur Prüfungs- und Studienleistung sowie formale Voraussetzungen.

Datenfelder

- **$Modul_ID**
  - **@ORM\Column**(type="integer", nullable=false)
  - **@ORM\ID**
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Erstellungsdatum**
  - **@ORM\Column**(type="date", nullable=false)
  - **@Assert\Date()**

- **$Versionsnummer**
  - **@ORM\Column**(type="integer", nullable=false)
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Status**
  - **@ORM\Column**(type="string", length=15, nullable=false)
  - **@Assert\Choice**(
    choices = { "in Planung", "Freigegeben", "expired" },
    message = "Bitte geben Sie einen korrekten Status an!"
)

- **$Kuerzel**
  - **@ORM\Column**(type="string", length=5, nullable=true)
  - **@Assert\Length**(
    min = 2,
    max = 5,
    minMessage = "Ein Modulkürzel muss aus mindestens {{ limit }} Zeichen bestehen.",
    maxMessage = "Ein Modulkürzel muss aus maximal {{ limit }} Zeichen bestehen."
)
  - **@Assert\Regex**(
    pattern = "/[A-Z0-9]{2,5}/",
    message = "Das Modulkürzel darf nur aus Großbuchstaben und Zahlen bestehen."
)

- **$Name**
  - **@ORM\Column**(type="string", length=70, nullable=false, unique=true)
  - **@Assert\Length**(
    min = 5,
    minMessage = "Der deutsche Modul-Titel muss aus mindestens {{ limit }} Zeichen bestehen.",
    max = 70,
    maxMessage = "Der deutsche Modul-Titel darf maximal aus {{ limit }} Zeichen bestehen.",
)
  - **@Assert\NotBlank**(
    message = "Der deutsche Modultitel muss gesetzt werden."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß0-9 \-]{5,70}/",
    message = "Der deutsche Name darf nur aus Buchstaben, Zahlen, Leerzeichen und Bindestrichen bestehen."
)

- **$NameEN**
  - **@ORM\Column**(type="string", length=70, nullable=true, unique=true)
  - **@Assert\Length**(
    min = 5,
    minMessage = "Der englische Modul-Titel muss aus mindestens {{ limit }} Zeichen bestehen.",
    max = 70,
    maxMessage = "Der englische Modul-Titel darf maximal aus {{ limit }} Zeichen bestehen.",
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß0-9 \-]{5,70}/",
    message = "Der englische Name darf nur aus Buchstaben, Zahlen, Leerzeichen und Bindestrichen bestehen."
)

- **$Haeufigkeit**
  - **@ORM\Column**(type="string", length=20, nullable=true)
  - **@Assert\Choice**(
    choices = {"Wintersemester", "Sommersemester", "wechselnd", "jedes Semester"},
    message = "Bitte geben Sie eine korrekte Haeufigkeit an!"
)

- **$Dauer**
  - **@ORM\Column**(type="string", length=30, nullable=true)

- **$Lehrveranstaltungen**
  - **@ORM\Column**(type="text", nullable=true)

- **$KontaktzeitVL**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Die Kontaktzeit Vorlesung muss mindestens {{ limit }} Stunden betragen."
)

- **$KontaktzeitSonstige**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Die Kontaktzeit Sonstige muss mindestens {{ limit }} Stunden betragen."
)

- **$Selbststudium**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Das Selbststudium muss mindestens {{ limit }} Stunden betragen."
)

- **$Gruppengroesse**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Die Gruppengroesse muss mindestens {{ limit }} Studenten betragen."
)

- **$Lernergebnisse**
  - **@ORM\Column**(type="text", nullable=true)

- **$Inhalte**
  - **@ORM\Column**(type="text", nullable=true)

- **$Pruefungsformen**
  - **@ORM\Column**(type="text", nullable=true)

- **$PruefungsformSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$PruefungsleistungSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$StudienleistungSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$Sprache**
  - **@ORM\Column**(type="text", nullable=true)
  - **@Assert\Choice**(
    choices = { "Deutsch", "Englisch" },
    message = "Bitte geben Sie eine korrekte Sprache an!"
)

- **$SpracheSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$Autor**
  - **@ORM\Column**(type="text", nullable=true)

- **$Literatur**
  - **@ORM\Column**(type="text", nullable=true)

- **$Leistungspunkte**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Choice**(
    choices = { "3", "6", "9", "12", "15", "30"},
    message = "Bitte geben Sie eine korrekte Anzahl an Leistungspunkten an!"
)

- **$VoraussetzungLP**
  - **@ORM\Column**(type="text", nullable=true)

- **$VoraussetzungInhalte**
  - **@ORM\Column**(type="text", nullable=true)

- **$semesterplan**
  - **@ORM\OneToMany**(targetEntity="Semesterplan",
    mappedBy="veranstaltung", cascade={"all"})

- **$angebot**
  - **@ORM\OneToMany**(targetEntity="Angebot", mappedBy="veranstaltung",
    cascade={"all"})

- **$lehrende**
  - **@ORM\OneToMany**(targetEntity="Lehrende", mappedBy="veranstaltung",
    cascade={"all"})

- **$beauftragter**
  - @**ORM\ManyToOne**(targetEntity="Dozent", inversedBy="modulbeauftragter")
  - @**ORM\JoinColumn**(name="modulbeauftragter", referencedColumnName="Dozenten_ID", nullable=false)

- **$studienplanModul**
  - @**ORM\OneToMany(**targetEntity="Studienplan", mappedBy="veranstaltung", cascade={"all"})

- **$kernfach**
  - @**ORM\OneToMany**(targetEntity="Kernfach", mappedBy="veranstaltung", cascade={"all"})

- **$grundmodul**
  - @**ORM\OneToMany**(targetEntity="Modulvoraussetzung", mappedBy="modul", cascade={"all"})

- **$forderung**
  - @**ORM\OneToMany**(targetEntity="Modulvoraussetzung", mappedBy="voraussetzung", cascade={"all"})

Getter und Setter

- __toString() → string
- __construct() → Veranstaltung
- getModulID() → integer
- setErstellungsdatum(DateTime $erstellungsdatum) → Veranstaltung
- getErstellungsdatum() → DateTime
- setVersionsnummer(integer $versionsnummer) → Veranstaltung
- getVersionsnummer() → integer
- setStatus(string $status) → Veranstaltung
- getStatus() → string
- setKuerzel(string $kuerzel) → Veranstaltung
- getKuerzel() → string
- setName(string $name) → Veranstaltung
- getName() → string
- setNameEN(string $nameEN) → Veranstaltung
- getNameEN() → string
- setHaeufigkeit(string $haeufigkeit) → Veranstaltung
- getHaeufigkeit() → string
- setDauer(string $dauer) → Veranstaltung

- getDauer() → string
- setLehrveranstaltungen(string $lehrveranstaltungen) → Veranstaltung
- getLehrveranstaltungen() → string
- setKontaktzeitVL(integer $kontaktzeitVL) → Veranstaltung
- getKontaktzeitVL() → integer
- setKontaktzeitSonstige(integer $kontaktzeitSonstige) → Veranstaltung
- getKontaktzeitSonstige() → integer
- setSelbststudium(integer $selbststudium) → Veranstaltung
- getSelbststudium() → integer
- setGruppengroesse(integer $gruppengroesse) → Vernastaltung
- getGruppengroesse() → integer
- setLernergebnisse(string $lernergebnisse) → Veranstaltung
- getLernergebnisse() → string
- setInhalte(string $inhalte) → Veranstaltung
- getInhalte() → string
- setPruefungsformen(string $pruefungsformen) → Veranstaltung
- getPruefungsformen() → string
- setPruefungsformSonstiges(string $pruefungsformSonstiges) → Veranstaltung
- getPruefungsformSonstiges() → string
- setPruefungsleistungSonstiges(string $pruefungsleistungSonstiges) → Veranstaltung
- getPruefungsleistungSonstiges() → string
- setStudienleistungSonstiges(string $studienleistungSonstiges) → Veranstaltung
- getStudienleistungSonstiges() → string
- setSprache(string $sprache) → Veranstaltung
- getSprache() → string
- setSpracheSonstiges($spracheSonstiges) → Veranstaltung
- getSpracheSonstiges() → string
- setAutor($autor) → Veranstaltung
- getAutor() → string
- setLiteratur($literatur) → Veranstaltung
- getLiteratur() → string
- setLeistungspunkte($leistungspunkte) → Veranstaltung
- setLeistungspunkte() → integer
- setVoraussetzungLP($voraussetzungLP) → Veranstaltung
- getVoraussetzungLP() → string
- setVoraussetzungInh($voraussetzungInhalte) → Veranstaltung
- getVoraussetzungInh() → string
- addSemesterplan(Semesterplan $semesterplan) → Veranstaltung
- removeSemesterplan(Semesterplan $semesterplan) → Veranstaltung
- getSemesterplan() → Collection

- addAngebot(Angebot $angebot → Veranstaltung
- removeAngebot(Angebot $angebot) → Veranstaltung
- getAngebot() → Collection
- addLehrende(Lehrende $lehrende) → Veranstaltung
- removeLehrende(Lehrende $lehrende) → Veranstaltung
- getLehrende() → Collection
- setBeauftragter(Dozent $beauftragter) → Veranstaltung
- getBeauftragter() → Dozent
- addStudienplanModul(Studienplan $studienplanModul) → Veranstaltung
- removeStudienplanModul(Studienplan $studienplanModul) → Veranstaltung
- getStudienplanModul() → Collection
- addKernfach(Kernfach $kernfach) → Veranstaltung
- removeKernfach(Kernfach $kernfach) → Veranstaltung
- getKernfach() → Collection
- setVoraussetzungInhalte($voraussetzungInhalte) → Veranstaltung
- getVoraussetzungInhalte() → string
- addForderung(Modulvoraussetzung $forderung) → Veranstaltung
- removeForderung(Modulvoraussetzung $forderung) → Veranstaltung
- getForderung() → Collection
- addGrundmodul(Modulvoraussetzung $grundmodul) → Veranstaltung
- removeGrundmodul(Modulvoraussetzung $grundmodul) → Veranstaltung
- getGrundmodul() → Collection

3.17. VeranstaltungHistory

VeranstaltungHistory bildet eine Schattentabelle zur eigentlichen Veranstaltung. Bei jeder Änderung an einer Veranstaltung wird die ursprüngliche Version hier abgespeichert.

Datenfelder

- **$Modul_ID**
  - **@ORM\Column**(type="integer", nullable=false)
  - **@ORM\ID**
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Erstellungsdatum**
  - **@ORM\Column**(type="date", nullable=false)
  - **@Assert\Date()**

- **$Versionsnummer**
  - **@ORM\Column**(type="integer", nullable=false)
  - **@ORM\ID**
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Kuerzel**
  - **@ORM\Column**(type="string", length=5, nullable=true)
  - **@Assert\Length**(
    min = 2,
    max = 5,
    minMessage = "Ein Modulkürzel muss aus mindestens {{ limit }} Zeichen bestehen.",
    maxMessage = "Ein Modulkürzel muss aus maximal {{ limit }} Zeichen bestehen."
)
  - **@Assert\Regex**(
    pattern = "/[A-Z0-9]{2,5}/",
    message = "Das Modulkürzel darf nur aus Großbuchstaben und Zahlen bestehen."
)

- **$Name**
  - **@ORM\Column**(type="string", length=70, nullable=false, unique=true)
  - **@Assert\Length**(
    min = 5,
    minMessage = "Der deutsche Modul-Titel muss aus mindestens {{ limit }} Zeichen bestehen.",
    max = 70,
    maxMessage = "Der deutsche Modul-Titel darf maximal aus {{ limit }} Zeichen bestehen.",
)
  - **@Assert\NotBlank**(
    message = "Der deutsche Modultitel muss gesetzt werden."
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß0-9 \-]{5,70}/",
    message = "Der deutsche Name darf nur aus Buchstaben, Zahlen, Leerzeichen und Bindestrichen bestehen."
)

- **$NameEN**
  - **@ORM\Column**(type="string", length=70, nullable=true, unique=true)
  - **@Assert\Length**(
    min = 5,
    minMessage = "Der englische Modul-Titel muss aus mindestens {{ limit }} Zeichen bestehen.",
    max = 70,
    maxMessage = "Der englische Modul-Titel darf maximal aus {{ limit }} Zeichen bestehen.",
)
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß0-9 \-]{5,70}/",
    message = "Der englische Name darf nur aus Buchstaben, Zahlen, Leerzeichen und Bindestrichen bestehen."
)

- **$Haeufigkeit**
  - **@ORM\Column**(type="string", length=20, nullable=true)
  - **@Assert\Choice**(
    choices = {"Wintersemester", "Sommersemester", "wechselnd", "jedes Semester"},
    message = "Bitte geben Sie eine korrekte Haeufigkeit an!"
)

- **$Dauer**
  - **@ORM\Column**(type="string", length=30, nullable=true)

- **$Lehrveranstaltungen**
  - **@ORM\Column**(type="text", nullable=true)

- **$KontaktzeitVL**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Die Kontaktzeit Vorlesung muss mindestens {{ limit }} Stunden betragen."
)

- **$KontaktzeitSonstige**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Die Kontaktzeit Sonstige muss mindestens {{ limit }} Stunden betragen."
)

- **$Selbststudium**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Das Selbststudium muss mindestens {{ limit }} Stunden betragen."
)

- **$Gruppengroesse**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Range**(
    min = 0,
    minMessage = "Die Gruppengroesse muss mindestens {{ limit }} Studenten betragen."
)

- **$Lernergebnisse**
  - **@ORM\Column**(type="text", nullable=true)

- **$Inhalte**
  - **@ORM\Column**(type="text", nullable=true)

- **$Pruefungsformen**
  - **@ORM\Column**(type="text", nullable=true)

- **$PruefungsformSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$PruefungsleistungSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$StudienleistungSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$Sprache**
  - **@ORM\Column**(type="text", nullable=true)
  - **@Assert\Choice**(
    choices = { "Deutsch", "Englisch" },
    message = "Bitte geben Sie eine korrekte Sprache an!"
)

- **$SpracheSonstiges**
  - **@ORM\Column**(type="text", nullable=true)

- **$Autor**
  - **@ORM\Column**(type="text", nullable=true)

- **$Literatur**
  - **@ORM\Column**(type="text", nullable=true)

- **$Leistungspunkte**
  - **@ORM\Column**(type="integer", nullable=true)
  - **@Assert\Choice**(
    choices = { "3", "6", "9", "12", "15", "30"},
    message = "Bitte geben Sie eine korrekte Anzahl an Leistungspunkten an!"
)

- **$VoraussetzungLP**
  - **@ORM\Column**(type="text", nullable=true)

- **$VoraussetzungInhalte**
  - **@ORM\Column**(type="text", nullable=true)

Getter und Setter

- __toString() → string
- __construct() → Veranstaltung
- getModulID() → integer
- setErstellungsdatum(DateTime $erstellungsdatum) → Veranstaltung
- getErstellungsdatum() → DateTime
- setVersionsnummer(integer $versionsnummer) → Veranstaltung
- getVersionsnummer() → integer
- setStatus(string $status) → Veranstaltung
- getStatus() → string
- setKuerzel(string $kuerzel) → Veranstaltung
- getKuerzel() → string
- setName(string $name) → Veranstaltung
- getName() → string
- setNameEN(string $nameEN) → Veranstaltung
- getNameEN() → string
- setHaeufigkeit(string $haeufigkeit) → Veranstaltung
- getHaeufigkeit() → string
- setDauer(string $dauer) → Veranstaltung
- getDauer() → string
- setLehrveranstaltungen(string $lehrveranstaltungen) → Veranstaltung
- getLehrveranstaltungen() → string
- setKontaktzeitVL(integer $kontaktzeitVL) → Veranstaltung
- getKontaktzeitVL() → integer
- setKontaktzeitSonstige(integer $kontaktzeitSonstige) → Veranstaltung
- getKontaktzeitSonstige() → integer
- setSelbststudium(integer $selbststudium) → Veranstaltung
- getSelbststudium() → integer
- setGruppengroesse(integer $gruppengroesse) → Vernastaltung
- getGruppengroesse() → integer
- setLernergebnisse(string $lernergebnisse) → Veranstaltung
- getLernergebnisse() → string

- setInhalte(string $inhalte) → Veranstaltung
- getInhalte() → string
- setPruefungsformen(string $pruefungsformen) → Veranstaltung
- getPruefungsformen() → string
- setPruefungsformSonstiges(string $pruefungsformSonstiges) → Veranstaltung
- getPruefungsformSonstiges() → string

- setPruefungsleistungSonstiges(string $pruefungsleistungSonstiges) → Veranstaltung
- getPruefungsleistungSonstiges() → string
- setStudienleistungSonstiges(string $studienleistungSonstiges) → Veranstaltung
- getStudienleistungSonstiges() → string
- setSprache(string $sprache) → Veranstaltung
- getSprache() → string
- setSpracheSonstiges($spracheSonstiges) → Veranstaltung
- getSpracheSonstiges() → string
- setAutor($autor) → Veranstaltung
- getAutor() → string
- setLiteratur($literatur) → Veranstaltung
- getLiteratur() → string
- setLeistungspunkte($leistungspunkte) → Veranstaltung
- setLeistungspunkte() → integer
- setVoraussetzungLP($voraussetzungLP) → Veranstaltung
- getVoraussetzungLP() → string
- setVoraussetzungInh($voraussetzungInhalte) → Veranstaltung
- getVoraussetzungInh() → string

3.18. Vertiefung

Dieses Entity speichert Vertiefungsrichtungen mit Fremdschlüssel auf den zugehörigen Studiengang ab.

Datenfelder

- **$Vertiefung_ID**
  - **@ORM\Column**(type="integer")
  - **@ORM\ID;**
  - **@ORM\GeneratedValue**(strategy="AUTO")

- **$Name**
  - **@ORM\Column**(type="string", length=30, unique=true, nullable=false)
  - **@Assert\NotBlank**(message = "Ein Vertiefungsrichtungstitel darf nicht leer sein.")
  - **@Assert\Length**(
    min = 4,
    minMessage = "Ein Vertiefungsrichtungstitel darf aus maximal {{ limit }} Zeichen bestehen.",
    max = 30,
    maxMessage = "Ein Vertiefungsrichtungstitel darf aus maximal {{ limit }} Zeichen bestehen.")
  - **@Assert\Regex**(
    pattern = "/[A-ZÄÖÜa-zäöüß \-]{4,30}/",
    message = "Ein Vertiefungsrichtungstitel darf nur aus Buchstaben, Leerzeichen und Bindestrichen bestehen."
)

- **$vertiefung**
  - **@ORM\OneToMany**(targetEntity="Kernfach", mappedBy="vertiefung", cascade={"all"})

- **$studiengang**
  - **@ORM\ManyToOne**(targetEntity="Studiengang", inversedBy="richtung")
  - **@ORM\JoinColumn**(name="studiengang", referencedColumnName="Studiengang_ID", nullable=false)

Getter und Setter

- __toString() → string
- __construct() → Veranstaltung
- getVertiefungID() → integer
- setName($name) → Vertiefung
- getName() → string
- addVertiefung(Kernfach $vertiefung) → Vertiefung
- removeVertiefung(Kernfach $vertiefung) → Vertiefung
- getVertiefung() → Vertiefung
- setStudiengang(Studiengang $stgang = null) → Studiengang
- getStudiengang() → Studiengang

4. Forms

4.1. AngebotType

FormType für Entity Angebot.

verwendeter Constructor:

public function __construct($studiengangID, $isWahl)

Parameter:

- $studiengangID → Typ integer
  - Wird zur Filterung der zur Auswahl stehenden Fachgebiete verwendet

- $isWahl → Typ Boolean
  - Gibt an, ob das Angebot ein Wahl- oder ein Pflichtfach ist
  - Wird zur Filterung der zur Auswahl stehenden Fachgebiete verwendet
  - **Wichtig:** Filterung der Fachgebiete nach Wahl- bzw Pflichtfach soll bei zukünftigen Code-Iterationen entfernt werden.

Formularfelder:

- **Fachgebiet**
  - Typ Fachgebiet-Entity
  - required = true
  - Ein Fachgebiet kann ausgewählt werden (Dropdown).

- **Kernfach**
  - Typ Kernfach-Entity
  - required = true
  - Erscheint nur wenn $isWahl == true
  - Gibt an, welchen Vertiefungsrichtungen das Angebot Kernfach zugeordnet wird.
  - Mehrfachauswahl möglich (Checkboxen).

- **abweichenderNameDE**
  - Typ text
  - required = false
  - Optionaler abweichender deutscher Titel des Angebots.

- **abweichenderNameEN**
  - Typ text
  - required = false
  - Optionaler abweichender englischer Titel des Angebots.

4.2. CodeType

FormType für Entity Angebot.

Formularfelder:

- **code**
  - Typ: text
  - required = true
  - Setzt den Modulcode eines Angebots

4.3. DozentType

FormType für Entity Dozent.

Formularfelder:

- **anrede**
  - Typ choice (ArrayValues::$gender)
  - required = true
  - Auswahl Herr/Frau (Dropdown)
- **titel**
  - Typ text
  - required = false
  - Beschreibt den Titel eines Dozenten.

- **name**
  - Typ text
  - required = true
  - Beschreibt den Vornamen eines Dozenten.

- **nachname**
  - Typ text
  - required = true
  - Beschreibt den Nachnamen eines Dozenten.

- **email**
  - Typ text
  - required = true
  - Beschreibt die E-Mail-Adresse eines Dozenten.

- **Roles**
  - Typ choice (Role-Entity)
  - required = true
  - Auswahl der Rolle des Dozenten im System (Dropdown)
  - Unterschieden wird zwischen den Rollen Dozent (ROLE_DOZENT) und Studiengangleiter (ROLE_SGL)

4.4. FachgebietType

FormType für Entity Fachgebiet.

Formularfelder:

- **titel**
  - Typ text
  - required = true
  - Beschreibt den Titel eines Fachgebiets

4.5. LehrendeType

FormType für Entity Lehrende.

Formularfelder:

- **dozent**
  - Typ choice (Dozent-Entity)
  - required = false
  - Dropdown-Liste zur Auwahl von Lehrenden Dozenten zu einem Modul.
  - Ein Dozent kann nicht mehrfach als Lehrender eines Moduls eingetragen sein.
  - Ein Dozent kann für mehrere verschiedene Module als Lehrender eingetragen sein.
  - Ein Modul kann mehrere Dozenten als Lehrende haben.
  - Falls kein Lehrender Dozent angegeben wurde wird automatisch der Modulbeauftragte als Lehrender des Moduls eingetragen.
  - Die Veranstaltung des Lehrende-Entity wird nicht über LehrendeType gesetzt.

4.6. ModulhandbuchType

FormType für Entity Modulhandbuch.

Formularfelder:

- **Beschreibung**
  - Typ text
  - required = true
  - Beschreibung des Modulhandbuchs

- **gueltigAb**
  - Typ choice (Semester-Entity)
  - required = true
  - Dropdown-Liste zur Auswahl des Geltungssemesters.
  - Gibt an, ab wann das Modulhandbuch gültig sein soll.

4.7. PlanungType

FormType für Entity Veranstaltung.

Formularfelder:

- **Kuerzel**
  - Typ text
  - required = false
  - Beschreibt das Kürzel einer in Planung befindlichen Veranstaltung.

- **Name**
  - Typ text
  - required = true
  - Beschreibt den deutschen Namen einer in Planung befindlichen Veranstaltung.

- **nameEN**
  - Typ text
  - required = false
  - Beschreibt den englischen Namen einer in Planung befindlichen Veranstaltung.

- **Selbststudium**
  - Typ integer
  - required = false
  - hidden = true
  - Wird im Hintergrund automatisch berechnet.
  - Beschreibt die Anzahl an Stunden für das Selbststudium einer in Planung befindlichen Veranstaltung.

- **Lernergebnisse**
  - Typ textarea
  - required = false
  - Beschreibt die Lernergebnisse einer in Planung befindlichen Veranstaltung.

- **Inhalte**
  - Typ textarea
  - required = false
  - Beschreibt die Lehrinhalte einer in Planung befindlichen Veranstaltung.

- **Literatur**
  - Typ textarea
  - required = false
  - Beschreibt die Literatur einer in Planung befindlichen Veranstaltung.

- **PruefungsformSonstiges**
  - Typ text
  - required = false
  - Beschreibt eine zusätzliche Information zum Feld 'pruefungsformen'.

- **Haeufigkeit**
  - Typ choice (ArrayValues::$frequency)
  - required = false
  - Default: Sommersemester
  - Beschreibt wann die Veranstaltung angeboten wird.

- **Dauer**
  - Typ integer
  - required = false
  - Default: 1
  - Beschreibt die Dauer der Veranstaltung.
  - Hier wird nur der Zahlenwert angegeben. Die Einheit (Wochen, Monate, Semester) existiert ebenfalls im Formular, wird aber an anderer Stelle im Quellcode definiert.

- **Leistungspunkte**
  - Typ choice (ArrayValues:: $lp)
  - required = false
  - Default: 6
  - Beschreibt wie viele Leistungspunkte (ECTS) die Veranstaltung hat.

- **kontaktzeitVL**
  - Typ integer
  - required = false
  - Default: 30
  - Beschreibt wie viele Stunden der Veranstaltung für Vorlesung vorgesehen sind.

- **kontaktzeitSonstige**
  - Typ integer
  - required = false
  - Default: 30
  - Beschreibt wie viele Stunden die Veranstaltung für sonstige Kontaktzeiten vorgesehen sind.

- **Gruppengroesse**
  - Typ integer
  - required = false
  - Default: 25
  - Beschreibt wie viele Studenten in der Vorlesung vorgesehen sind.

- **Sprache**
  - Typ choice (ArrayValues:: $lang)
  - required = false
  - Default: Deutsch
  - Beschreibt in welcher Sprache die Veranstaltung gehalten wird.

- **spracheSonstiges**
  - Typ text
  - required = false
  - Beschreibt weitere Details zur Sprache der Veranstaltung falls nötig.

- **voraussetzungInh**
  - Typ textarea
  - required = false
  - Beschreibt die inhaltlichen Vorrausetzungen der Veranstaltung.

- **Lehrveranstaltungen**
  - Typ choice (ArrayValues:: $lehrveranstaltungen)
  - required = false
  - Default: Vorlesung, Übung
  - Beschreibt den Aufbau der Veranstaltung.

- **Pruefungsformen**
  - Typ choice (ArrayValues:: $pruefungsformen)
  - required = false
  - Default: Schriftliche Klausur
  - Beschreibt die Prüfungsform der Veranstaltung.

- **voraussetzungLP**
  - Typ choice (ArrayValues:: $voraussetzungLP)
  - required = false
  - Default: Prüfungsleistung
  - Beschreibt die Voraussetzung für die Vergabe der Leistungspunkte der Veranstaltung.

- **PruefungsleistungSonstiges**
  - Typ text
  - required = false
  - Default: Bestandene Modulprüfung
  - Beschreibt mögliche Sonderfälle der Prüfungsform einer Veranstaltung.

4.8. SemesterplanType

FormType für Entity SemesterPlan.

Formularfelder:

- **Semester**
  - Typ text
  - required = true
  - Gibt das Semester an in dem die Veranstaltung angeboten werden soll.

- **Lehrender**
  - Typ text
  - required = true
  - Gibt an welcher Dozent die Veranstaltung im angegebenen Semester lehrt.

- **Module**
  - Typ text
  - required = true
  - Gibt die Veranstaltung an.

- **SWSUebung**
  - Typ text
  - required = true
  - Gibt an wie viele SWS für Übungen im angegebenen Semester vorgesehen sind.

- **SWSVorlesung**
  - Typ text
  - required = true
  - Gibt an wie viele SWS für Vorlesungen im angegebenen Semester vorgesehen sind.

- **anzahlUebungsgruppen**
  - Typ text
  - required = true
  - Gibt an wie viele Übungsgruppen im angegebenen Semester vorgesehen sind.

- **groesseUebungsgruppen**
  - Typ text
  - required = true
  - Gibt an welche Gruppengröße pro Übungsgruppe im angegebenen Semester geplant ist.

4.9. SemesterType

FormType für Entity Semester.

Formularfelder:

- **Semester**
  - Typ text
  - required = true
  - Gibt das Semester an.

4.10. StudiengangType

FormType für Entity Studiengang.

Formularfelder:

- **Fachbereich**
  - Typ choice (ArrayValues:: $faculty)
  - required = true
  - Gibt an in welchem Fachbereich sich der Studiengang befindet.

- **Grad**
  - Typ choice (ArrayValues:: $level)
  - required = true
  - Gibt den Bildungsgrad des Studiengangs an.

- **Titel**
  - Typ text
  - required = true
  - Gibt den Titel des Studiengangs an.

- **Kuerzel**
  - Typ text
  - required = true
  - Gibt das Kürzel des Studiengangs an.

- **Beschreibung**
  - Typ textarea
  - required = true
  - Gibt eine Beschreibung für den Studiengang an.

- **Sgl**
  - Typ Dozent-Entity
  - required = true
  - Ein Dozent kann ausgewählt werden (Dropdown).
  - Nur Dozenten mit der Rolle „ROLE_SGL“ werden in dem Dropdown Menü angezeigt.

- **Richtung**
  - Typ Collection (VertiefungType)
  - required = true
  - Hier können eine oder mehrere Vertiefungsrichtungen angeben werden.

- **Fachgebiete**
  - Typ Collection (VertiefungType)
  - required = true
  - Hier können ein oder mehrere Fachgebiete angeben werden.

4.11. StudienplanType

FormType für Entity Studienplan.

Formularfelder:

- **RegelSemester**
  - Typ choice (ArrayValues:: $regelsem)
  - required = true
  - Gibt das oder die Regelsemester des Moduls im Studienplan an.

- **StartSemester**
  - Typ Semester-Entity
  - required = true
  - Unterscheidung zwischen Start im Sommer- und Wintersemester.

- **Veranstaltung**
  - Typ Veranstaltung-Entity
  - required = true
  - Die Veranstaltung, für die ein Studienplan-Entity angelegt werden soll.

- **Studiengang**
  - Typ Studiengang-Entity
  - required = true
  - Ein Studiengang kann ausgewählt werden (Dropdown).

4.12. VeranstaltungType

FormType für Entity Veranstaltung.

Formularfelder:

- **Beauftragter**
  - Typ Veranstaltung -Entity
  - required = true
  - Ein Dozent, der für die Verwaltung des Moduls beauftragt sein soll, kann ausgewählt werden (Dropdown)

- **Kuerzel**
  - Typ text
  - required = true
  - Beschreibt das Kürzel der Veranstaltung

- **Name**
  - Typ text
  - required = true
  - Beschreibt den Namen der Veranstaltung

- **NameEN**
  - Typ text
  - required = true
  - Beschreibt den englischen Namen der Veranstaltung

- **Haeufigkeit**
  - Typ choice (ArrayValues:: $frequency)
  - required = true
  - Beschreibt wie oft eine Veranstaltung angeboten wird.

- **KontaktzeitVL**
  - Typ integer
  - required = true
  - Beschreibt wie viele Stunden der Veranstaltung für Vorlesungen vorgesehen sind.

- **KontaktzeitSonstige**
  - Typ integer
  - required = true
  - Beschreibt wie viele Stunden der Veranstaltung für sonstige Kontaktzeiten vorgesehen sind.

- **Selbststudium**
  - Typ integer
  - required = true
  - Beschreibt wie viele Stunden der Veranstaltung für Selbststudium vorgesehen sind.

- **Gruppengroesse**
  - Typ integer
  - required = true
  - Beschreibt wie viele Studenten in der Veranstaltung vorgesehen sind.

- **Lernergebnisse**
  - Typ integer
  - required = true
  - Gibt die Lernergebnisse der Veranstaltung an.

- **Inhalte**
  - Typ textarea
  - required = true
  - Gibt die Lehrinhalte der Veranstaltung an.

- **Sprache**
  - Typ choice (ArrayValues:: $lang)
  - required = true
  - Gibt die Sprache der Veranstaltung an.

- **SpracheSonstiges**
  - Typ text
  - required = true
  - Gibt sonstige Informationen zur Sprache der Veranstaltung an.

- **Literatur**
  - Typ textarea
  - required = true
  - Gibt mögliche Literatur der Veranstaltung an.

- **Leistungspunkte**
  - Typ choice (ArrayValues:: $lp)
  - required = true
  - Gibt die Leistungspunkte des Moduls an.

- **VoraussetzungInh**
  - Typ textarea
  - required = true
  - Gibt die Inhaltlichen Vorrausetzungen an.

- **PruefungsleistungSonstiges**
  - Typ text
  - required = true
  - Gibt zusätzliche Informationen zu der Prüfungsleistung an.

- **Dauer**
  - Typ integer
  - required = true
  - Default: 1
  - Gibt die Dauer der Veranstaltung an.

- **VorausetzungenLP**
  - Typ choice (ArrayValues:: $voraussetzungLP)
  - required = true
  - Gibt die Voraussetzungen für Leistungspunkte des Moduls an.

- **Pruefungsformen**
  - Typ choice (ArrayValues:: $pruefungsformen)
  - required = true
  - Gibt die Prüfungsformen des Moduls an.

- **PruefungsformSonstiges**
  - Typ text
  - required = true
  - Gibt zusätzliche Informationen zu den Prüfungsformen an.

- **Lehrveranstaltungen**
  - Typ choice (ArrayValues: $lehrveranstaltungen)
  - required = true
  - Gibt die Lehrveranstaltungsart des Moduls an.

- **Lehrende**
  - Typ Collection (LehrendenTyp)
  - required = true
  - Hier kann ein oder mehrere Lehrende angeben werden.

- **Grundmodul**
  - Typ Collection (VoraussetzungsTyp)
  - required = true
  - Hier kann eine oder mehrere Veranstaltungen angegeben werden, die der aktuelle Veranstaltung vorausgesetzt werden.

4.13. VertiefungType

FormType für Entity Vertiefung.

Formularfelder:

- **Name**
  - Typ text
  - required = false
  - Gibt den Namen der Vertiefungsrichtung an.

4.14. VorAngebotType

verwendeter Constructor:

public function __construct($studiengangIDs)
Parameter:

- $studiengangIDs → Typ integer
  - Wird zur Filterung der zur Auswahl stehenden Studiengänge verwendet.

Formularfelder:

- **Studiengang**
  - Typ Studiengang-Entity
  - required = true
  - Ein Studiengang kann ausgewählt werden (Dropdown)

- **Angebotsart**
  - Typ choice (ArrayValues: $offerTypes)
  - required = true
  - Gibt die Angebotsart des Angebots an.

- **Studienplan_ws**
  - Typ choice (ArrayValues: $regelsem)
  - required = true
  - Gibt das Regelsemester bei Start im Wintersemester für das Modul an.

- **Studienplan_ss**
  - Typ choice (ArrayValues: $regelsem)
  - required = true
  - Gibt das Regelsemester bei Start im Sommersemester für das Modul an.

4.15. VoraussetzungType

FormType für Entity Modulvoraussetzung.

Formularfelder:

- **Voraussetzung**
  - Typ Veranstaltung-Entity (Dropdown)
  - required = false
  - Gibt die Veranstaltung an die als Voraussetzung gelten soll.

5. Eigene PHP-Klassen

5.1. ArrayValues

In der Datei **ArrayValues.php** wurden alle nötigen Arrays ausgelagert. Zu beachten ist hierbei, dass es sich um Key-Value-Pairs handelt. Zusätzlich müssen bei Änderungen an den Arrays die entsprechenden Assertions im entsprechenden Entity angepasst werden.
In ArrayValues ausgelagerte Values sind:

- Häufigkeit der Anbietung eines Moduls ($frequency)
- Angaben zur Dauer eines Moduls ($duration)
- Sprachen ($lang)
- Leistungspunkte ($lp)
- Fachbereiche ($faculty)
- Studienganggrade ($level)
- Regelsemester ($regelsem)
- Angebotsarten ($offerTypes)
- Anreden ($gender)
- Voraussetzungen für die Vergabe von Leistungspunkten ($voraussetzungLP)
- Prüfungsformen ($pruefungsformen)
- Lehrveranstaltungen ($lehrveranstaltungen)

5.2. ModulBeschreibung

Diese Klasse wird als Container für das Generieren des Modulhandbuch-PDF verwendet.

5.2.1. Datenfelder

- $angebot
- $fremdeStudiengaenge
- $studienplaene
- $voraussetzungen
- $pruefungsformen
- $lehrveranstaltungen
- $voraussetzungenLP

5.2.2. Getter und Setter

- public function setAngebot(Angebot $angebot) → Angebot-Entity
- public function getAngebot() → Angebot-Entity
- public function setFremdeStudiengaenge(array $fremdeStudiengaenge) → Studiengang-Entity-Array
- public function getStudienplaene() → Studiengang-Entity-Array
- public function setStudienplaene(array $studienplaene) → Studienplan-Entity-Array
- public function getStudienplaene() → Studienplan-Entity-Array
- public function setVoraussetzungen($grundmodul) → Veranstaltung-Entity-Array
- public function getVoraussetzungen() → Veranstaltung-Entity-Array
- public function setPruefungsformen($pruefungsformen) → string-Array oder null
- public function getPruefungsformen() → string-Array oder null
- public function setLehrveranstaltungen($lehrveranstaltungen) → string-Array oder null
- public function getLehrveranstaltungen() → string-Array oder null
- public function setVoraussetzungenLP($voraussetzungenLP) → string-Array oder null
- public function getVoraussetzungenLP() → string-Array oder null

5.3. SortFunctions

In der Datei SortFunctions.php wurden alle Sortierfunktionen für eigene Entites ausgelagert. Diese werden innerhalb des Codes via „bool uasort (array &$array , callable $value_compare_func)“ aufgerufen.
Beispielsweise wird die Sortierfunktion für Dozenten folgendermaßen aufgerufen: „uasort($dozentenArray,array('FHBingen\Bundle\MHBBundle\PHP\SortFunctions', 'dozentSort'));“

- **public static function angebotSort(Angebot $angebotA, Angebot $angebotB)**
  Diese Funktion sortiert die übergebenen Angebot-Entities. Sofern ein abweichender deutscher Titel angegeben wurde, wird auf diesen geprüft, ansonsten wird auf den Titel des Moduls geprüft.

- **public static function dozentSort(Dozent $dozA, Dozent $dozB)**
  Diese Funktion sortiert die übergebenen Dozent-Entities nach den Nachnamen der Dozenten.

- **public static function modulBeschreibungSort(ModulBeschreibung $descA, ModulBeschreibung $descB)**
  Diese Funktion sortiert die übergebenen ModulBeschreibung-Entites. Zuerst wird nach Fachgebieten sortiert, danach innerhalb der Fachgebiete nach Modulcode. Die hieraus resultierende Reihenfolge wird im zu generierenden Modulhandbuch beibehalten.

- **public static function studienplanSort(Studienplan $planA, Studienplan $planB)**
  Diese Funktion sortiert die übergebenen Studienplan-Entites nach deren Startsemester. Zu beachten ist hierbei, dass eine alphabetische Reihenfolge angewendet wird, also z.B. SS12 vor WS15 aufgelistet wird.

- **public static function veranstaltungSort(Veranstaltung $modulA, Veranstaltung $modulB)**
  Diese Funktion sortiert die übergebenen Veranstaltung-Entities nach deren deutschen Titel.

5.4. UserDependentRole

- implements RoleInterface

wurde eingeführt, damit jeder Benutzer eine eigene Rolle erhält, z.B. 'ROLE_USER@EMAIL.COM', wurde dann aber in unserem Security-Modell nicht konsequent umgesetzt

- public function __construct(Dozent $user)
  - Setzt privates Datenelement $dozent auf übergebenes Dozent-Objekt

- public function getRole()
  - return-Wert: String
  - 'ROLE_'.strtoupper($this->user->getEmail())

5.5. UserRepository

- extends EntityRepository
- implements UserProviderInterface

Diese Klasse wird verwendet um den Login via Email-Adresse zu ermöglichen. Sie implementiert nur die vom Interface benötigten Funktionen.

Weitere Infos:
http://api.symfony.com/2.6/Symfony/Component/Security/Core/User/UserProviderInterface.html

public function loadUserByUsername($username)

- Gibt an, an welcher Stelle der übergebene Benutzername zu suchen ist (Feld 'email').
- Parameter 'username' vom Typ string
- Return-Typ: UserInterface

public function refreshUser(UserInterface $user)

- Lädt die Datenelemente des Benutzers neu, wenn der übergebene Benutzer kompatibel ist.
- Parameter 'user' vom Typ UserInterface
- Return-Typ: UserInterface oder null

public function supportsClass($class)

- Prüft ob die übergegebene Klasse zum UserRepository kompatibel ist.
- Parameter 'class' vom Typ string
- Return-Typ: boolean

6. Views

6.1. layout.htm.twig

Initiales Template von dem alle weiteren Templates abgeleitet werden.

Blocks:

- **userLogout Block**
  Beinhaltet die Logout Funktion und Navigationsbalken.
- **Content Block**
  Kapselt den eigentlichen Seiteninhalt.

6.2. flashbags.html.twig

Template das eingebunden wird um Fehler- oder Informationsmeldungen zu werfen.

Blocks: -

Message:

- Alert: Zeigt den Meldungstext als Javascript alert an.

6.3. buttons.html.twig

Template das eingebunden wird wenn Buttons benötigt werden

Blocks: -

Buttons:

- Zurücksetzen
- Speichern

6.4. eigeneModule.html.twig

Startseite des Systems. Je nach Rolle wird die entsprechende Navigationsleiste und der Startcontent gerendert.

extends 'FHBingenMHBBundle::layout.html.twig'

Blocks:

- **Content Block**
  Zwei Tabellen die die Veranstaltungen anzeigen, in denen der angemeldete Nutzer entweder Modulverantwortlicher oder Lehrender ist.

Übergebene Daten:

- veranstaltung: Array enthält die Veranstaltungen
- name: Array enthält die Studiengänge denen die einzelnen Veranstaltungen zugeordnet sind

6.5. angebot.html.twig

Maske um die Informationen zu Angeboten in die Datenbank zu schreiben.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/buttons.html.twig'

Blocks:

- **form_row Block**
  Manuelle Anpassung des Templates form_row um den Inhalt in eine Tabelle zu schreiben.

- **form_label Block**
  Enthält die Spalten Beschreibungen des Formulars

- **form_widget_compound Block**
  Notwendig für die Formatierung

- **Content Block**
  Enthält den inhaltlichen Aufbau des Formulars. Formulartype ist AngebotType.

Übergebene Daten:

modul: Enthält den Name der Veranstaltungen
studiengang: Enthält den Name des Studiengangs
form: Daten des Formulars

6.6. planungBearbeitung.html.twig

Maske um die Informationen zu einem geplanten Modul in die Datenbank zu schreiben.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'
include '@FHBingenMHB/buttons.html.twig'

Blocks:

- **form_row Block**
  Enthält den Aufbau des Formulars.

- **form_label Block**
  Enthält die Spalten Beschreibungen des Formulars

- **form_widget_compound Block**
  Notwendig für die Formatierung.
- **Content Block**
  Enthält den inhaltlichen Aufbau des Formulars. Formulartyp ist VeranstaltungType.
- **voraussetzung Block**
  Leerer Block, da die Voraussetzungen in der Planung nicht gesetzt werden.
- **lehrende Block**
  Leerer Block, da die Lehrenden in der Planung nicht gesetzt werden.

6.7. modulBearbeitung.html.twig

Maske um die Informationen zu einem Modul in die Datenbank zu schreiben.

extends 'FHBingenMHBBundle:Dozent:planungBearbeiten.html.twig'

Blocks:

- **jsBlock Block**
  Enthält Javascript um mehrere Lehrende und Voraussetzungen auf einmal hinzuzufügen.

- **formStart Block**
  Startet das Formular um Lehrende oder Voraussetzungen hinzuzufügen.

- **beauftragter Block**
  Notwendig für die Formatierung.

- **voraussetzung Block**
  Fügt die Voraussetzungen hinzu.

- **lehrende Block**
  Fügt die Lehrenden hinzu.

6.8. planungUebersicht.html.twig

Übersichtsseite der geplanten Module.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Eine Tabelle, die die geplanten Veranstaltungen anzeigt, die der angemeldete Nutzer erstellt hat. Zusätzlich besteht die Möglichkeit ein neues Modul zu planen, frei zu geben oder zu löschen.

Übergebene Daten:
planung: Array enthält die geplanten Veranstaltungen

6.9. vorAngebot.html.twig

Maske um die Informationen zu einem Vorangebot in die Datenbank zu schreiben.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'
include '@FHBingenMHB/buttons.html.twig'

Blocks:

- **form_row Block**
  Enthält den Aufbau des Formulars.
- **form_label Block**
  Enthält die Spalten Beschreibungen des Formulars
- **form_widget_compound Block**
  Notwendig für die Formatierung.
- **Content Block**
  Enthält den inhaltlichen Aufbau des Formulars. Formulartyp ist VorAngebotType.

Übergebene Daten:
modul: Enthält den Name der Veranstaltungen

6.10. login.html.twig

Login Seite des Systems

extends 'FHBingenMHBBundle::layout.html.twig'

Blocks:

- **Content Block**
  Enthält die Login Funktionen.

6.11. adminNav.html.twig

Enthält die Links die in der Navigationsleiste angezeigt werden für den Administrations Bereich.

Blocks: -

Links:

- **benutzerverwaltungLink**
  path: benutzerVerwaltung
- **studiengangverwaltungLink:**
  path: studiengangVerwaltung

6.12. DozentNav.html.twig

Enthält die Links, die in der Navigationsleiste angezeigt werden für den Dozenten Bereich.

Blocks: -

Links:

- **eigeneModuleLink**
  path: eigeneModule
- **modulplanungLink:**
  path: planungAnzeigen

6.13. generalNav.html.twig

Enthält die Links, welche in der Navigationsleiste angezeigt werden für den allgemeinen Bereich.

Blocks: -

Links:

- **dokuLink**
  path: doku

6.14. sglNav.html.twig

Enthält die Links, die in der Navigationsleiste angezeigt werden für den Studiengangleiter Bereich.

Blocks: -

Links:

- **alleModuleLink:**
  path: alleModule
- **modulAenderungenLink:**
  path: modulAenderungen
- **modulcodeLink:**
  path: modulCodeUebersicht
- **mhbUebersichtLink:**
  path: mhbUebersicht
- **mhbErstellungLink:**
  path: mhbErstellung
- **moduldeaktivierungLink:**
  path: deaktivierungAlleModule

6.15. navigation.html.twig

Haupttemplate für die Navigation.

include 'FHBingenMHBBundle:Navigation:dozentNav.html.twig'
include 'FHBingenMHBBundle:Navigation:sglNav.html.twig‘
include 'FHBingenMHBBundle:Navigation:adminNav.html.twig'
include 'FHBingenMHBBundle:Navigation:generalNav.html.twig'

6.16. alleModule.html.twig

Übersicht über alle Module, studiengangübergreifend.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Eine Tabelle mit allen Veranstaltungen und den Studiengängen, denen diese zugeordnet sind.

Übergebene Daten:

modul:	Array enthält die Veranstaltungen
name:	Array enthält die Studiengänge denen die einzelnen Veranstaltungen zugeordnet sind

6.17. mhbModul.html.twig

Formatiert die Informationen einer Veranstaltung die einem Modulhandbuch zugeordnet ist, sodass sie für den PDF Export geeignet ist.

Erstellt mit Hilfe der Übergabeparameter modulBeschreibung das Layout für den PDF Export(Tabellarisch).

6.18. mhbUebersicht.html.twig

Übersicht über alle Modulhandbücher, studiengangübergreifend.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Eine Tabelle mit allen Modulehandbüchern.

6.19. mhbZusammenstellung.html.twig

Maske die es dem Nutzer ermöglicht Module zu einem neuen Modulhandbuch abzuwählen bzw. hinzuzufügen.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Listet die Angebote auf, die dem MHB zugordnet werden sollen. Zuordnung erfolgt per Drag and Drop. Alle Angebote sind per default dem Modulhandbuch zugeordnet.

Übergebene Daten:

angebot: Array enthält die Angebote

6.20. modulAenderung.html.twig

Übersicht über alle Veranstaltungen die seit Erstellung des letzten Modulhandbuchs im Studiengang des eingeloggten Dozenten geändert wurden.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Eine Tabelle mit allen Veranstaltungen die seit Erstellung des letzten Modulhandbuchs im Studiengang des eingeloggten Dozenten geändert wurden.

Übergebene Daten:
veranstaltung: Array enthält die Veranstaltungen

6.21. modulCodeErstellung.html.twig

Maske um die Informationen zu einem Modulcode in die Datenbank zu schreiben.

extends FHBingenMHBBundle::layout.html.twig

Blocks:

- **form_row Block**
  Enthält den Aufbau des Formulars.

- **Content Block**
  Enthält den inhaltlichen Aufbau des Formulars. Formulartyp ist CodeType.

6.22. moduleCodeUebersicht.html.twig

Übersicht über alle Module des eigenen Studiengangs, inklusive der Module ohne gesetzten Modulcode, die im eigenen Studiengang angeboten werden sollen.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Zwei Tabellen mit den Angeboten deren Modulcode nicht gesetzt ist und der Angebote deren Code schon gesetzt wurde.

Übergebene Daten:
dummy: Array enthält die Angboten deren Modulcode noch DUMMY ist.
normal: Array enthält die Angboten deren Modulcode gesetzt wurde.

6.23. moduleDeaktiverung.html.twig

Übersicht über alle deaktivierten Module des Systems und der aktiven Module des eigenen Studiengangs mit der Möglichkeit Module zu reaktivieren bzw. zu deaktivieren.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Zwei Tabellen mit aktiven und deaktivierten Modulen.

Übergebene Daten:

deaktiv: Array enthält die Veranstaltungen die deaktiviert sind
studiengang: Enthät den Name des Studiengangs.
modul: Array enthält die Veranstaltungen die Aktiv sind

6.24. alleStudiengänge.html.twig

Übersicht über alle Studiengänge.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Auflistung aller im System verwalteten Studiengänge und deren Leiter.

Übergebene Daten:

courses: Array enthält die Studiengänge

6.25. benutzerVerwaltung.html.twig

Übersicht über alle Benutzer. Nutzer können hier deaktiviert oder aktiviert werden.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/flashbags.html.twig'

Blocks:

- **Content Block**
  Auflistung aller im System vorhandenen Benutzer, sortiert nach deren Rolle.

Übergebene Daten:

deaktivierteNutzer:	Array enthält die deaktivierten Nutzer
sgl:	Array enthält die Studiengangleiter
dozent.	Array enthält die Dozenten

6.26. benutzerErstellung.html.twig

Rendert das Formular zur Benutzererstellung.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/buttons.html.twig'

Blocks:

- **form_row Block**
  Enthält den Aufbau des Formulars.

- **Content Block**
  Enthält den inhaltlichen Aufbau des Formulars. Formulartyp ist DozentType.

6.27. studiengangAnzeigen.html.twig

Maske um die Informationen zu einem Studiengang in die Datenbank zu schreiben.

extends 'FHBingenMHBBundle::layout.html.twig'
include '@FHBingenMHB/buttons.html.twig'

Blocks:

- **form_row Block**
  Enthält den Aufbau des Formulars.
- **form_label Block**
  Enthält die Spalten Beschreibungen des Formulars.
- **form_widget_compound Block**
  Notwendig für die Formatierung.
- **Content Block**
  Enthält den inhaltlichen Aufbau des Formulars. Formulartyp ist StudiengangType.

Übergebene Daten:
Studiengang: Enhält den Name des Studiengangs.

6.28. listelement.html.twig

Prototypetemplate für die dynamischen Felder Modulvoraussetzung und Lehrende.

Blocks: **-**

7. Weiterführende Links

- Doctrine-Doku: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/
- Twig-Doku: http://twig.sensiolabs.org/documentation
- Symfony-Doku: http://symfony.com/doc/current/book/index.html