Abschlussprüfung Sommer 2016

Fachinformatiker Anwendungsentwicklung

Quelltexte

HBS Proxy

Jan Erik Helmle

Azubi-Ident-Nummer: 3600437

Prüfungsausschuss: FIAN 11

JSON Schema der Preise und Verfügbarkeiten 0,3 x

JSON-Schema statische Produktdaten 1,4 x

InitialDataController.php 1,5 x

TotalPriceController.php 1,3 x

Product.php (gekürzt) 3,3 x

PriceRepository.php 1,7 x

LoadAdditionalProductData.php 2 x

JSON-Schema der Preise und Verfügbarkeiten

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "identifier": {
                "type": "string"
            },
            "price": {
                "type": "string"
            }
        },
        "required": [
            "identifier",
            "price"
        ]
    }
}
```

JSON-Schema der statischen Produktdaten

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "roomtypes": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "identifier": {
                        "type": "string"
                    },
                    "subMenuText": {
                        "type": "string"
                    },
                    "listText": {
                        "type": "string"
                    },
                    "pricingBasisText": {
                        "type": "string"
                    },
                    "capacity": {
                        "type": "integer"
                    }
                },
                "required": [
                    "identifier",
                    "subMenuText",
                    "listText",
                    "pricingBasisText",
                    "capacity"
                ]
            }
        },
        "boardings": {
            "type": "array",
            "items": {
```

```
                "type": "object",
                "properties": {
                    "identifier": {
                        "type": "string"
                    },
                    "listText": {
                        "type": "string"
                    },
                    "pricingBasisText": {
                        "type": "string"
                    }
                },
                "required": [
                    "identifier",
                    "listText",
                    "pricingBasisText"
                ]
            }
        },
        "specials": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "identifier": {
                        "type": "string"
                    },
                    "listText": {
                        "type": "string"
                    },
                    "pricingBasisText": {
                        "type": "string"
                    }
                },
                "required": [
                    "identifier",
                    "listText",
                    "pricingBasisText"
                ]
            }
        }
    },
    "required": [
        "roomtypes",
        "boardings",
        "specials"
    ]
}
```

InitialDataController.php

```
<?php

namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use AppBundle\Entity\Product;
use AppBundle\Entity\RoomType;
use AppBundle\Entity\AdditionalProduct;
use AppBundle\Entity\AdditionalProductCategory;
```

```
class InitialDataController extends Controller {

    /**
     * @Route("/api/v1/initialdata", name="initialdata_v1")
     */
    public function initialDataAction_v1(Request $request) {

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

        $query = $em->createQuery('
                SELECT

                p.identifier ,
                p.subMenuText ,
                p.listText ,
                p.pricingBasisText ,
                p.capacity

                FROM AppBundle:RoomType p

                 ORDER BY p.positionInSubMenu ASC

                ');

        $roomtypes = $query->getResult();

        $query = $em->createQuery("
                SELECT

                a.identifier ,

                a.listText ,
                a.pricingBasisText

                FROM AppBundle:AdditionalProduct a
                JOIN a.additionalproductcategory p
                WHERE p.identifier = 'boardings'

                 ORDER BY a.positionInList ASC

                ");

        $boardings = $query->getResult();

        $query = $em->createQuery("
                SELECT

                a.identifier ,

                a.listText ,
                a.pricingBasisText

                FROM AppBundle:AdditionalProduct a
                JOIN a.additionalproductcategory p
                WHERE p.identifier = 'specials'

                 ORDER BY a.positionInList ASC

                ");

        $specials = $query->getResult();


        $products = array(
            'roomtypes' => $roomtypes,
```

```
            'boardings' => $boardings,
            'specials' => $specials
        );

        $productsJSON = json_encode($products, 320); // 320 : 0000000101000000 = 256 + 64 :
        // JSON_UNESCAPED_SLASHES => 64 + JSON_UNESCAPED_UNICODE => 256

        $resp = new Response($productsJSON);
        $resp->headers->set('Content-Type', 'application/json ; charset=utf-8');

        return $resp;
    }

}
```

TotalPriceController.php

```
<?php

namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use AppBundle\Entity\Cart;
use AppBundle\Entity\Item;
use AppBundle;

class TotalPriceController extends Controller {

    /**
     * @Route("/api/v1/totalPrice", name="totalPrice_v1")
     */
    public function totalPriceAction_v1(Request $request) {

        $input = $request->getContent();

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

        if (
                (!$input)
        ) {


            $resp = new Response(
                    "Malformed request syntax. "
                    . " "
            );
            $resp->setStatusCode(Response::HTTP_BAD_REQUEST);
            $resp->headers->set('Content-Type', 'Content-Type: text/html; charset=utf-8');
            return $resp;
        }

        $input_sanitized = str_replace(array("\n", "\t", "\r"), '', $input); // remove newlines , tabs , carriage
                                                                                return

        $json_decoded = json_decode($input_sanitized, false); // false -> object , true -> array

        $c = new Cart();
        $c->setCheckInDate($json_decoded->checkInDate);
        $c->setCheckOutDate($json_decoded->checkOutDate);
        foreach ($json_decoded->items as $item) {
```

```
            $i = new Item();
            $i->setRoomTypeIdentifier($item->roomTypeIdentifier);
            $i->setRoomTypeQuantity($item->roomTypeQuantity);
            $i->setBoardingIdentifier($item->boardingIdentifier);
            $i->setSpecialIdentifier($item->specialsIdentifier);
            $c->addItem($i);
        }

        $totalPrice = $em->getRepository('AppBundle:Cart')->calculateTotalPrice($c); // Zugriff über

        EntityRepository

        $totalPriceJSON = json_encode($totalPrice, 320); // 320 : 0000000101000000 = 256 + 64 :
        // JSON_UNESCAPED_SLASHES => 64 + JSON_UNESCAPED_UNICODE => 256

        $resp = new Response($totalPriceJSON);
        $resp->headers->set('Content-Type', 'application/json ; charset=utf-8');

        return $resp;
    }

}
```

Product.php (gekürzt)

```
<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Doctrine\Common\Collections\ArrayCollection;

/**
 * @ORM\Entity(repositoryClass="AppBundle\Entity\ProductRepository")
 * @ORM\Table(name="product")
 * @ORM\InheritanceType("JOINED")
 * @ORM\DiscriminatorColumn(name="discriminator", type="string")
 * @ORM\DiscriminatorMap({"product" = "Product", "roomType" = "RoomType" , "additionalProduct" =
                                                                                "AdditionalProduct"})
 */
abstract class Product {

    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(type="string" , length=100 , nullable=false , unique=true)
     */
    private $identifier;

    /**
     * @ORM\Column(type="string" , length=100 , nullable=true)
     */
    private $listText;

    /**
     * @ORM\Column(type="string" , length=100 , nullable=true)
     */
    private $pricingBasis;
```

```
    /**
     * @ORM\Column(type="string" , length=100 , nullable=true)
     */
    private $pricingBasisText;

    /**
     * @ORM\OneToMany(targetEntity="Price", mappedBy="product")
     */
    private $prices; // ArrayCollection

    public function __construct() {
        $this->prices = new ArrayCollection();
    }

    /**
     * @ORM\ManyToOne(targetEntity="Hotel", inversedBy="products")
     * @ORM\JoinColumn(name="hotel_id", referencedColumnName="id")
     */
    private $hotel;

        //
        // Getter- und Settermethoden hier gekürzt
        //

    /**
     * Add price
     *
     * @param \AppBundle\Entity\Price $price
     *
     * @return Product
     */
    public function addPrice(\AppBundle\Entity\Price $price) {
        $this->prices[] = $price;

        return $this;
    }

    /**
     * Remove price
     *
     * @param \AppBundle\Entity\Price $price
     */
    public function removePrice(\AppBundle\Entity\Price $price) {
        $this->prices->removeElement($price);
    }

    /**
     * Get prices
     *
     * @return \Doctrine\Common\Collections\Collection
     */
    public function getPrices() {
        return $this->prices;
    }

    /**
     * Set hotel
     *
     * @param \AppBundle\Entity\Hotel $hotel
     *
     * @return Product
     */
    public function setHotel(\AppBundle\Entity\Hotel $hotel = null) {
        $this->hotel = $hotel;
```

```
        return $this;
    }

    /**
     * Get hotel
     *
     * @return \AppBundle\Entity\Hotel
     */
    public function getHotel() {
        return $this->hotel;
    }

}
```

PriceRepository.php

```
<?php

namespace AppBundle\Entity;

use DateTime;
use DateInterval;
use DatePeriod;

/**
 * PriceRepository
 *
 * This class was generated by the Doctrine ORM. Add your own custom
 * repository methods below.
 */
class PriceRepository extends \Doctrine\ORM\EntityRepository { // returns price object

    public function findLatestPricePerProductAndDateTime(Product $prod, DateTime $date) {

        $em = $this->getEntityManager();

        $query = $em->createQuery(
                'SELECT pri FROM AppBundle:price pri JOIN pri.product prod WHERE prod.identifier = ?1 ORDER BY
                                pri.date DESC'
        );
        $query->setParameter(1, $prod->getIdentifier());

        $result = $query->getResult(); // Array of Price Objects

        foreach ($result as $r) { // laufe durch ResultSet vom Neuesten zum Aeltesten
            if ($r->getDate() <= $date) {
                return $r;
            }
        }

        return $result[0];
    }

    public function calculateTotalAmountPerProductAndDateInterval(Product $prod, DateTime $checkIn, DateTime
                                                                                        $checkOut) { // returns double
        $em = $this->getEntityManager();

        $sum = 0;

        $interval = new DateInterval('P1D'); // 1 Tag
        $daterange = new DatePeriod($checkIn, $interval, $checkOut);

        foreach ($daterange as $date) {
```

```
            $sum += $em->getRepository('AppBundle:Price')->findLatestPricePerProductAndDateTime($prod, $date)-

        >getValue();
        }

        return $sum;
    }

    public function calculatePriceAveragePerProductAndDateInterval(Product $prod, DateTime $checkIn, DateTime
                                                                                    $checkOut) { // returns double
        $em = $this->getEntityManager();

        $sum = 0;
        $days = 0;
        $avg = 0;

        $interval = new DateInterval('P1D'); // 1 Tag
        $daterange = new DatePeriod($checkIn, $interval, $checkOut);

        foreach ($daterange as $date) {
            $days += 1;
            $sum += $em->getRepository('AppBundle:Price')->findLatestPricePerProductAndDateTime($prod, $date)-

        >getValue();
        }

        $avg = round($sum / $days, 2); // auf 2 Nachkommastellen gerundet

        return $avg; // arithmetischer Mittelwert
    }

    public function findLatestPricePerProductIdentifierAndDateTime($productString, DateTime $date) { // returns
                                                                                                price object
        $em = $this->getEntityManager();

        $query = $em->createQuery(
                'SELECT prod FROM AppBundle:product prod WHERE prod.identifier = ?1'
        );
        $query->setParameter(1, $productString);

        $productObject = $query->getResult()[0]; // 1 ProductObject

        $query = $em->createQuery(
                'SELECT pri FROM AppBundle:price pri JOIN pri.product prod WHERE prod.identifier = ?1 ORDER BY
                                pri.date DESC'
        );
        $query->setParameter(1, $productObject->getIdentifier());

        $result = $query->getResult(); // Array of Price Objects

        foreach ($result as $r) { // laufe durch ResultSet vom Neuesten zum Aeltesten
            if ($r->getDate() <= $date) {
                return $r;
            }
        }

        return $result[0];
    }

}
```

LoadAdditionalProductData.php

```
<?php

namespace AppBundle\DataFixtures\ORM;

use Doctrine\Common\DataFixtures\AbstractFixture;
use Doctrine\Common\DataFixtures\OrderedFixtureInterface;
use Doctrine\Common\Persistence\ObjectManager;
use AppBundle\Entity\AdditionalProduct;

class LoadAdditionalProductData extends AbstractFixture implements OrderedFixtureInterface {

    public function load(ObjectManager $manager) {

        $ap1 = new AdditionalProduct();
        $ap1->setIdentifier("halfpension");
        $ap1->setListText("Halbpension (mit Frühstück)");
        $ap1->setPricingBasis("person,night");
        $ap1->setPricingBasisText("/Pers. u. Nacht");
        $ap1->setPositionInList(1);
        $ap1->setHotel($this->getReference('testhotel'));
        $this->addReference('halfpension', $ap1);
        $ap1->setAdditionalProductcategory($this->getReference('Verpflegung'));
        $manager->persist($ap1);
        $manager->flush();

        $ap2 = new AdditionalProduct();
        $ap2->setIdentifier("fullpension");
        $ap2->setListText("Vollpension (3 Mahlzeiten)");
        $ap2->setPricingBasis("person,night");
        $ap2->setPricingBasisText("/Pers. u. Nacht");
        $ap2->setPositionInList(2);
        $ap2->setHotel($this->getReference('testhotel'));
        $this->addReference('fullpension', $ap2);
        $ap2->setAdditionalProductcategory($this->getReference('Verpflegung'));
        $manager->persist($ap2);
        $manager->flush();

        $ap3 = new AdditionalProduct();
        $ap3->setIdentifier("breakfast");
        $ap3->setListText("Nur Frühstück");
        $ap3->setPricingBasis("person,night");
        $ap3->setPricingBasisText("/Pers. u. Nacht");
        $ap3->setPositionInList(3);
        $ap3->setHotel($this->getReference('testhotel'));
        $this->addReference('breakfast', $ap3);
        $ap3->setAdditionalProductcategory($this->getReference('Verpflegung'));
        $manager->persist($ap3);
        $manager->flush();

        $ap4 = new AdditionalProduct();
        $ap4->setIdentifier("noboarding");
        $ap4->setListText("Ohne Verpflegung");
        $ap4->setPricingBasis("");
        $ap4->setPricingBasisText("");
        $ap4->setPositionInList(4);
        $ap4->setHotel($this->getReference('testhotel'));
        $this->addReference('noboarding', $ap4);
        $ap4->setAdditionalProductcategory($this->getReference('Verpflegung'));
        $manager->persist($ap4);
        $manager->flush();

        $ap5 = new AdditionalProduct();
```

```
        $ap5->setIdentifier("champagnebreakfast");
        $ap5->setListText("Sektfrühstück");
        $ap5->setPricingBasis("person,night");
        $ap5->setPricingBasisText("/Pers. u. Nacht");
        $ap5->setPositionInList(1);
        $ap5->setHotel($this->getReference('testhotel'));
        $this->addReference('champagnebreakfast', $ap5);
        $ap5->setAdditionalProductcategory($this->getReference('Specials'));
        $manager->persist($ap5);
        $manager->flush();

        $ap6 = new AdditionalProduct();
        $ap6->setIdentifier("rosesinrooms");
        $ap6->setListText("Rosen auf das Zimmer");
        $ap6->setPricingBasis("");
        $ap6->setPricingBasisText("");
        $ap6->setPositionInList(2);
        $ap6->setHotel($this->getReference('testhotel'));
        $this->addReference('rosesinrooms', $ap6);
        $ap6->setAdditionalProductcategory($this->getReference('Specials'));
        $manager->persist($ap6);
        $manager->flush();

        $ap7 = new AdditionalProduct();
        $ap7->setIdentifier("raftingtour");
        $ap7->setListText("Rafting-Tour");
        $ap7->setPricingBasis("person");
        $ap7->setPricingBasisText("/Person");
        $ap7->setPositionInList(3);
        $ap7->setHotel($this->getReference('testhotel'));
        $this->addReference('raftingtour', $ap7);
        $ap7->setAdditionalProductcategory($this->getReference('Specials'));
        $manager->persist($ap7);
        $manager->flush();
    }

    public function getOrder() {
        // the order in which fixtures will be loaded
        // the lower the number, the sooner that this fixture is loaded
        return 3;
    }

}
```