educationorganizer – mit CDI, REST, JPA 2, JEE7 auf WildFly

Das MVC Framework 'Angular JS' ermöglicht ebenfalls unter Einsatz des **Router Design Patterns** und des **Front Controller Design Patterns** das Erstellen von responsiven (Mobile) Applikationen zur Stammdatenpflege von über ein Service Interface verfügbaren Daten aus einem RESTful WebService. Die erstellte JEE7 Beispiel-Applikation 'educationorganizer' arbeitet auf Basis der Template-Technologie und ermöglicht die leichte Pflege von und die Suche nach Binaris Events, Medien, Lokationen, etc. An einem einfachen, übersichtlichen JEE6 bzw. JEE7 Beispielprojekt wird die Verwendung der eingesetzten Technologien gezeigt und dieses als .war Archiv-Deployment auf dem JBoss WildFly 8.2 Application Server zum Test zur Verfügung gestellt.

Die Beispiel Applikation

Die Beispiel-Applikation wurde mit Hilfe von JBoss Forge erstellt. Über den Einsatz von JBoss Forge wird es noch weitere Beispiele für JSF Applikationen geben.

Die Technologien:

Die Beispiel-Applikation 'educationorganizer' verwendet im Frontend Angular.js und HTML5-/CSS3, als auch die JavaScript-Bibliotheken Bootstrap.js, Backbone.js, jQuery und jQuery Mobile, sowie die Underscore.js Tools.

Für das JAX-RS Service-Backend kommt Java EE zum Einsatz (Stateful/Stateless EJB 3.2 und JEE 7) und im Backend Model JPA 2.1 Entities. Die Applikation wurde auf dem 'JBoss WildFly 8.2' und dem 'JBoss EAP 6.3' Applikationsserver erfolgreich deployt und getestet und kann gerne weiterverwendet und als Open Source weiterentwickelt werden.

Für die Weiterentwicklung, den Build und das Deployment des Projekts ist mindestens Java 6 und Maven 3 erforderlich. Als Entwicklungsumgebung wurde Eclipse 4.4 (Luna) in Form des "JBoss Developer Studios 8.1.0" mit Java 7 und Java 8 verwendet und die FireFox WebDeveloper IDE/Tools.

Hier die verwendete pom.xml (mit den WildFly JEE 7 Dependencies):

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>de.binaris</groupId>
   <artifactId>educationorganizer</artifactId>
   <version>1.0.0-SNAPSHOT</version>
   <packaging>war</packaging>

   <name>educationorganizer</name>

   <description>Responsive Angular.js, REST, JEE7 webapp for use on WildFly, JBoss EAP</description>

   <properties>
      <!-- Explicitly declaring the source encoding eliminates the following
         message: -->
      <!-- [WARNING] Using platform encoding (UTF-8 actually) to copy filtered
         resources, i.e. build is platform dependent! -->
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <!-- Timestamp format for the maven.build.timestamp property -->
      <maven.build.timestamp.format>yyyyMMdd'T'HHmmss</maven.build.timestamp.format>
      <!-- Specify the JBoss AS directory to be the JBOSS_HOME environment -->
      <jboss.home>${env.JBOSS_HOME}</jboss.home>
      <!-- Define the version of JBoss' Java EE 6 APIs and Tools -->
      <!-- <jboss.bom.version>1.0.4.Final-redhat-9</jboss.bom.version> -->
      <!-- Alternatively, comment out the above line, and un-comment the
         line below to use version 1.0.4.Final-redhat-9 which is a release certified
         to work with JBoss EAP 6.2. It requires you have access to the JBoss EAP 6.2
         maven repository. -->
      <jboss.bom.version>6.2.3.GA</jboss.bom.version>
      <jboss.wfk.bom.version>2.6.0-redhat-1</jboss.wfk.bom.version>
      <buildhelper.plugin.version>1.7</buildhelper.plugin.version>
   </properties>

   <dependencyManagement>
      <dependencies>
         <!-- JBoss distributes a complete set of Java EE 6 APIs including a Bill
            of Materials (BOM). A BOM specifies the versions of a "stack" (or a collection)
            of artifacts. We use this here so that we always get the correct versions
            of artifacts. Here we use the jboss-javaee-6.0-with-tools stack (you can
            read this as the JBoss stack of the Java EE 6 APIs, with some extras tools
            for your project, such as Arquillian for testing) and the jboss-javaee-6.0-with-hibernate
            stack you can read this as the JBoss stack of the Java EE 6 APIs, with extras
            from the Hibernate family of projects) -->
            <dependency>
                    <groupId>org.wildfly.bom</groupId>
                    <artifactId>jboss-javaee-7.0-with-tools</artifactId>
                    <version>8.1.0.Final</version>
                    <type>pom</type>
                    <scope>import</scope>
            </dependency>
            <dependency>
                    <groupId>org.wildfly.bom</groupId>
                    <artifactId>jboss-javaee-7.0-with-hibernate</artifactId>
                    <version>8.1.0.Final</version>
                    <type>pom</type>
                    <scope>import</scope>
            </dependency>
            <dependency>
                    <groupId>org.wildfly.bom</groupId>
                    <artifactId>jboss-javaee-7.0-with-resteasy</artifactId>
                    <version>8.1.0.Final</version>
                    <type>pom</type>
                    <scope>import</scope>
            </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>

   <!-- First declare the APIs we depend on and need for compilation. All
      of them are provided by JBoss AS 7 -->

   <!-- Import the CDI API, we use provided scope as the API is included in
      WildFly -->
   <dependency>
      <groupId>javax.enterprise</groupId>
      <artifactId>cdi-api</artifactId>
      <scope>provided</scope>
   </dependency>

   <!-- Import the Common Annotations API (JSR-250), we use provided scope
      as the API is included in WildFly -->
   <dependency>
      <groupId>org.jboss.spec.javax.annotation</groupId>
      <artifactId>jboss-annotations-api_1.1_spec</artifactId>
      <scope>provided</scope>
   </dependency>

   <!-- Import the JAX-RS API, we use provided scope as the API is included
      in WildFly -->
   <dependency>
      <groupId>org.jboss.spec.javax.ws.rs</groupId>
      <artifactId>jboss-jaxrs-api_2.0_spec</artifactId>
      <version>1.0.0.Alpha1</version>
      <scope>provided</scope>
   </dependency>

   <!-- Import the JPA API, we use provided scope as the API is included in
      WildFly -->
   <dependency>
      <groupId>org.hibernate.javax.persistence</groupId>
      <artifactId>hibernate-jpa-2.1-api</artifactId>
      <scope>provided</scope>
   </dependency>

   <!-- Import the EJB API, we use provided scope as the API is included in
      WildFly -->
   <dependency>
      <groupId>org.jboss.spec.javax.ejb</groupId>
      <artifactId>jboss-ejb-api_3.2_spec</artifactId>
      <scope>provided</scope>
   </dependency>

   <!-- JSR-303 (Bean Validation) Implementation -->
   <!-- Provides portable constraints such as @Email et al -->
   <!-- Hibernate Validator is shipped in JBoss EAP -->
   <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-validator</artifactId>
      <scope>provided</scope>
      <exclusions>
         <exclusion>
```

```
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
         </exclusion>
      </exclusions>
   </dependency>

   <!-- Now we declare any tools needed -->

   <!-- Annotation processor to generate the JPA 2.0 metamodel classes
      for typesafe criteria queries -->
   <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-jpamodelgen</artifactId>
      <scope>provided</scope>
   </dependency>

   <!-- Needed for running tests (you may also use TestNG) -->
   <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <scope>test</scope>
   </dependency>

   <!-- Optional, but highly recommended -->
   <!-- Arquillian allows you to test enterprise code such as EJBs and
      Transactional(JTA) JPA from JUnit/TestNG -->
   <dependency>
      <groupId>org.jboss.arquillian.junit</groupId>
      <artifactId>arquillian-junit-container</artifactId>
      <scope>test</scope>
   </dependency>

   <dependency>
      <groupId>org.jboss.arquillian.protocol</groupId>
      <artifactId>arquillian-protocol-servlet</artifactId>
      <scope>test</scope>
   </dependency>

   <dependency>
      <groupId>org.jboss.shrinkwrap.resolver</groupId>
      <artifactId>shrinkwrap-resolver-depchain</artifactId>
      <type>pom</type>
      <scope>test</scope>
   </dependency>

   <!-- RESTEasy dependencies that bring in Jackson Core and RESTEasy APIs+SPIs ->
   <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-jackson2-provider</artifactId>
      <scope>provided</scope>
   </dependency>
   <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-jaxrs</artifactId>
      <scope>provided</scope>
   </dependency>

   <dependency>
      <groupId>org.jboss.spec.javax.servlet</groupId>
      <artifactId>jboss-servlet-api_3.1_spec</artifactId>
      <scope>provided</scope>
   </dependency>

</dependencies>

<build>
   <!-- Maven will append the version to the finalName (which is the
  name given to the generated war, and hence the context root) -->
   <finalName>${project.artifactId}</finalName>
   <pluginManagement>

      <plugins>
         <!-- Compiler plugin enforces Java 1.6 compatibility and activates
        annotation processors -->
         <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.3.1</version>
            <configuration>
               <source>1.7</source>
               <target>1.7</target>
            </configuration>
         </plugin>
         <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.1.1</version>
            <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
               <archive>
                  <manifestEntries>
                     <Dependencies>org.jboss.as.naming,org.jboss.as.server,org.jboss.msc</Dependencies>
                  </manifestEntries>
               </archive>
            </configuration>
         </plugin>

         <!-- The JBoss AS plugin deploys your war to a local JBoss AS container -->
         <!-- To use run: mvn package jboss-as:deploy -->
         <plugin>
            <groupId>org.jboss.as.plugins</groupId>
            <artifactId>jboss-as-maven-plugin</artifactId>
            <version>7.4.Final</version>
         </plugin>

      </plugins>
   </pluginManagement>
</build>

<profiles>
   <profile>
      <!-- The default profile skips all tests, though you can tune
         it to run just unit tests based on a custom pattern -->
      <!-- Seperate profiles are provided for running all tests, including
         Arquillian tests that execute in the specified container -->
      <id>default</id>
      <activation>
         <activeByDefault>true</activeByDefault>
      </activation>
      <build>
         <plugins>
            <plugin>
               <artifactId>maven-surefire-plugin</artifactId>
               <version>2.4.3</version>
               <configuration>
                  <skip>true</skip>
               </configuration>
            </plugin>
         </plugins>
      </build>
   </profile>

   <profile>
      <!-- An optional Arquillian testing profile that executes tests in your
         JBoss AS instance -->
      <!-- This profile will start a new JBoss AS instance, and execute the
         test, shutting it down when done -->
      <!-- Run with: mvn clean test -Parq-jbossas-managed -->
      <id>arq-wildfly-managed</id>
      <dependencies>
                  <dependency>
                          <groupId>org.wildfly</groupId>
                          <artifactId>wildfly-arquillian-container-managed</artifactId>
                          <scope>test</scope>
                  </dependency>
      </dependencies>
   </profile>

   <profile>
      <!-- An optional Arquillian testing profile that executes tests in a remote
         JBoss AS instance -->
      <!-- Run with: mvn clean test -Parq-jbossas-remote -->
      <id>arq-wildfly-remote</id>
      <dependencies>
                  <dependency>
                          <groupId>org.wildfly</groupId>
```

```
                        <artifactId>wildfly-arquillian-container-remote</artifactId>
                        <scope>test</scope>
                    </dependency>
    </dependencies>
</profile>
<profile>
    <!-- An optional profile that enables a server managed mysql data source -->
    <id>mysql</id>
    <build>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
            </resource>
            <resource>
                <directory>src/main/resources</directory>
                <includes>
                    <include>**/*</include>
                </includes>
                <excludes>
                    <exclude>META-INF/*</exclude>
                </excludes>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.4.3</version>
                <configuration>
                    <skip>true</skip>
                </configuration>
            </plugin>
        </plugins>
    </build>
</profile>

<profile>
    <!-- Create a release distribution with the right directory layout. -->
    <!-- product build specific assembly -->
    <id>release-dist</id>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <descriptors>
                        <descriptor>src/main/assembly/assembly.xml</descriptor>
                    </descriptors>
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <configuration>
                    <skip>true</skip>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.2.1</version>
                <executions>
                    <execution>
                        <phase>compile</phase>
                        <goals>
                            <goal>exec</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <executable>git</executable>
                <arguments>
                  <argument>init</argument>
                </arguments>
              </configuration>
            </plugin>
          </plugins>
        </build>
      </profile>

    </profiles>

    <repositories>
      <repository>
        <id>jboss-ga-repository</id>
        <url>http://maven.repository.redhat.com/techpreview/all</url>
        <releases>
          <enabled>true</enabled>
        </releases>
        <snapshots>
          <enabled>false</enabled>
        </snapshots>
      </repository>
    </repositories>
    <pluginRepositories>
      <pluginRepository>
        <id>jboss-ga-plugin-repository</id>
        <url>http://maven.repository.redhat.com/techpreview/all</url>
        <releases>
          <enabled>true</enabled>
        </releases>
        <snapshots>
          <enabled>false</enabled>
        </snapshots>
      </pluginRepository>
    </pluginRepositories>

</project>
```

Wie in der pom.xml erkennbar ist, werden hier die JEE7 Artefakte für den JBoss WildFly verwendet und importiert:

WildFly JBoss Java EE 7 Specification APIs with Tools:
- jboss-javaee-7.0-with-tools

WildFly JBoss Java EE 7 Specification APIs with Resteasy:
- jboss-javaee-7.0-with-resteasy

WildFly JBoss Java EE 7 Specification APIs with Hibernate:
- jboss-javaee-7.0-with-hibernate

Weiterhin werden die folgenden JEE Dependencies verwendet:

- jboss-annotations-api_1.1_spec
- jboss-jaxrs-api_2.0_spec
- resteasy-jackson2-provider

- hibernate-jpa-2.1-api
- jboss-ejb-api_3.2_spec
- hibernate-jpamodelgen
- jboss-servlet-api_3.1_spec

Über CDI und JavaEE gibt es bereits Blog-Einträge in diesem Blog hier (http://binaris-informatik.de/?p=577) und hier (http://binaris-informatik.de/?p=2591). Die Aktivierung von CDI erfolgt, wie beschrieben, mittels **beans.xml** im WEB-INF Verzeichnis.

Die Architektur:

a) Frontend:

Hier nun der Router für die Desktop-Applikation in der app/router/desktop/router.js:

```
/**
 * A module for the router of the desktop application
 */
define("router", [
    'jquery',
    'underscore',
    'configuration',
    'utilities',
    'app/models/event',
    'app/models/venue',
    'app/collections/events',
    'app/collections/venues',
    'app/views/desktop/home',
    'app/views/desktop/events',
    'app/views/desktop/venues',
    'app/views/desktop/event-detail',
    'app/views/desktop/venue-detail',
    'text!../templates/desktop/main.html'
],function ($,
        _,
        config,
        utilities,
        Event,
        Venue,
        Events,
        Venues,
        HomeView,
        EventsView,
        VenuesView,
        EventDetailView,
        VenueDetailView,
        MainTemplate) {

    $(document).ready(new function() {
       utilities.applyTemplate($('body'), MainTemplate)
    })

    /**
     * The Router class contains all the routes within the application -
     * i.e. URLs and the actions that will be taken as a result.
     *
     * @type {Router}
     */

    var Router = Backbone.Router.extend({
        initialize: function() {
            //Begin dispatching routes
            Backbone.history.start();
        },
        routes:{
            "":"home",
            "about":"home",
            "events":"events",
            "events/:id":"eventDetail",
            "venues":"venues",
            "venues/:id":"venueDetail",
            "ignore":"ignore",
            "*actions":"defaultHandler"
        },
        events:function () {
            var events = new Events();
            var eventsView = new EventsView({model:events, el:$("#content")});
            events.on("reset",
                function () {
                    utilities.viewManager.showView(eventsView);
                }).fetch({
                    reset : true,
                    error : function() {
                        utilities.displayAlert("Failed to retrieve events from the Education Organizer server.");
                    }
                });
        },
        venues:function () {
            var venues = new Venues;
            var venuesView = new VenuesView({model:venues, el:$("#content")});
            venues.on("reset",
                function () {
                    utilities.viewManager.showView(venuesView);
                }).fetch({
                    reset : true,
                    error : function() {
                        utilities.displayAlert("Failed to retrieve venues from the Education Organizer server.");
                    }
                });
        },
        home:function () {
            utilities.viewManager.showView(new HomeView({el:$("#content")}));
        },
        eventDetail:function (id) {
            var model = new Event({id:id});
            var eventDetailView = new EventDetailView({model:model, el:$("#content")});
            model.on("change",
                function () {
                    utilities.viewManager.showView(eventDetailView);
                }).fetch({
                    error : function() {
                        utilities.displayAlert("Failed to retrieve the event from the Education Organizer server.");
                    }
                });
        },
        venueDetail:function (id) {
            var model = new Venue({id:id});
            var venueDetailView = new VenueDetailView({model:model, el:$("#content")});
            model.on("change",
                function () {
                    utilities.viewManager.showView(venueDetailView);
                }).fetch({
                    error : function() {
                        utilities.displayAlert("Failed to retrieve the venue from the Education Organizer server.");
                    }
                });
        }
    });

    // Create a router instance
    var router = new Router();

    return router;
});
```

Und hier der Router für die Mobile-Applikation in der app/router/mobile/router.js:

```
/**
 * A module for the router of the mobile application.
 */
define("router",[
    'jquery',
    'jquerymobile',
    'underscore',
    'utilities',
    'app/models/event',
    'app/models/venue',
    'app/collections/events',
    'app/collections/venues',
    'app/views/mobile/events',
    'app/views/mobile/venues',
    'app/views/mobile/event-detail',
    'app/views/mobile/venue-detail',
    'text!../templates/mobile/home-view.html'
],function ($,
            jqm,
            _,
            utilities,
            Event,
            Venue,
            Events,
            Venues,
            EventsView,
            VenuesView,
            EventDetailView,
            VenueDetailView,
            HomeViewTemplate) {

    /**
     * The Router class contains all the routes within the application - i.e. URLs and the actions
     * that will be taken as a result.
     *
     * @type {Router}
     */
    var Router = Backbone.Router.extend({
            initialize: function() {
```

```
        //Begin dispatching routes
                    Backbone.history.start();
            },
        routes:{
            "":"home",
            "events":"events",
            "events/:id":"eventDetail",
            "venues":"venues",
            "venues/:id":"venueDetail",
            "ignore":"ignore",
            "*actions":"defaultHandler"
        },
        defaultHandler:function (actions) {
            if ("" != actions) {
                $("body").pagecontainer( "change", "#" + actions, {transition:'slide', changeHash:false,
allowSamePageTransition:true});
            }
        },
        home:function () {
            utilities.applyTemplate($("#container"), HomeViewTemplate);
            try {
                $("#container").enhanceWithin();
            } catch (e) {
                // workaround for a spurious error thrown when creating the page initially
            }
        },
        events:function () {
            var events = new Events;
            var eventsView = new EventsView({model:events, el:$("#container")});
            events.on("reset", function() {
                utilities.viewManager.showView(eventsView);
            }).fetch({
                reset : true,
                error : function() {
                    utilities.displayAlert("Failed to retrieve events from the Education Organizer server.");
                }
            });
        },
        venues:function () {
            var venues = new Venues;
            var venuesView = new VenuesView({model:venues, el:$("#container")});
            venues.on("reset",
                function () {
                    utilities.viewManager.showView(venuesView);
                }).fetch({
                    reset : true,
                    error : function() {
                        utilities.displayAlert("Failed to retrieve venues from the Education Organizer server.");
                    }
                });
        },
        eventDetail:function (id) {
            var model = new Event({id:id});
            var eventDetailView = new EventDetailView({model:model, el:$("#container")});
            model.on("change",
                function () {
                    utilities.viewManager.showView(eventDetailView);
                    $("body").pagecontainer("change", "#container", {transition:'slide', changeHash:false});
                }).fetch({
                    error : function() {
                        utilities.displayAlert("Failed to retrieve the event from the Education Organizer server.");
                    }
                });
        },
        venueDetail:function (id) {
            var model = new Venue({id:id});
            var venueDetailView = new VenueDetailView({model:model, el:$("#container")});
            model.on("change",
                function () {
                    utilities.viewManager.showView(venueDetailView);
                    $("body").pagecontainer("change", "#container", {transition: 'slide', changeHash: false});
                }).fetch({
                    error : function() {
                        utilities.displayAlert("Failed to retrieve the venue from the Education Organizer server.");
                    }
                });
        }
```

```
    });

    // Create a router instance
    var router = new Router();

    return router;
});
```

<u>Hier das Frontend Model für die Events /app/models/event.js:</u>

```
/**
 * The frontend Event model
 */
define([
    'configuration',
    'backbone'
], function (config) {
    /**
     * The Event model class definition
     * Used for CRUD operations against individual events
     */
    var Event = Backbone.Model.extend({
        urlRoot: config.baseUrl + 'rest/events' // the URL for performing CRUD operations
    });
    // export the Event class
    return Event;
});
```

<u>Hier die Frontend Collection mit Comparator zum Sortieren der Events /app/collections/events.js:</u>

```
/**
 * Module for the Events collection
 */
define([
    // The collection element type and configuration are dependencies
    'app/models/event',
    'configuration',
    'backbone'
], function (Event, config) {
    /**
     *  Here we define the Bookings collection
     *  We will use it for CRUD operations on Bookings
     */
    var Events = Backbone.Collection.extend({
        url: config.baseUrl + "rest/events", // the URL for performing CRUD operations
        model: Event,
        id:"id", // the 'id' property of the model is the identifier
        comparator:function (model) {
            return model.get('category').id;
        }
    });
    return Events;
});
```

<u>Hier das Frontend Model für die Venues (Schulungsorte) /app/models/venue.js:</u>

```
/**
 * The frontend Venue model
 */
define([
    'configuration',
    'backbone'
], function (config) {

    /**
     * The Venue model class definition
     * Used for CRUD operations against individual events
     */
    var Venue = Backbone.Model.extend({
        urlRoot: config.baseUrl + 'rest/venues'
    });

    return Venue;
});
```

<u>Hier die Frontend Collection mit Comparator zum Sortieren der Venues /app/collections/venues.js:</u>

```
/**
 * Module for the Events collection
 */
define([
    // Configuration and the collection element type are dependencies
    'app/models/venue',
    'configuration',
    'backbone'
], function (Venue, config) {

    return Backbone.Collection.extend({
        url: config.baseUrl + "rest/venues",
        model:Venue,
        id:"id",
        comparator:function (model) {
            return model.get('address').city;
        }
    });
});
```

Die JavaScript Konfigurationen finden sich im Verzeichnis /configurations:

- desktop.js
- hybrid.js
- loader.js
- mobile.js

Hier die desktop.js:

```
/**
 * Shortcut alias definitions - will come in handy when declaring dependencies
 * Also, they allow you to keep the code free of any knowledge about library
 * locations and versions
 */
requirejs.config({
    baseUrl: "resources/js",
    paths: {
        jquery:'libs/jquery-2.0.3',
        underscore:'libs/underscore',
        text:'libs/text',
        bootstrap: 'libs/bootstrap',
        backbone: 'libs/backbone',
        utilities: 'app/utilities',
        router:'app/router/desktop/router'
    },
    // We shim Backbone.js and Underscore.js since they don't declare AMD modules
    shim: {
        'backbone': {
            deps: ['jquery', 'underscore'],
            exports: 'Backbone'
        },

        'underscore': {
                exports: '_'
        }
    }
});

define("initializer", ["jquery"], function ($) {
    // Configure jQuery to append timestamps to requests, to bypass browser caches
    // Important for MSIE
            $.ajaxSetup({cache:false});
    $('head').append('<link rel="stylesheet" href="resources/css/bootstrap.css" type="text/css" media="all"/>');
    $('head').append('<link rel="stylesheet" href="resources/css/bootstrap-theme.css" type="text/css" media="all"/>');
    $('head').append('<link rel="stylesheet" href="resources/css/screen.css" type="text/css" media="all"/>');
    $('head').append('<link href="http://fonts.googleapis.com/css?family=Rokkitt" rel="stylesheet" type="text/css">');
});

// Now we declare all the dependencies
// This loads and runs the 'initializer' and 'router' modules.
require([
    'initializer',
    'router'
], function(){
});

define("configuration", {
    baseUrl : ""
});
```

Hier die hybrid.js mit der baseRESTUrl (ist beim produktiven Deployment anzupassen):

```
// override configuration for RESTful services
var EducationOrganizer = {
    config:{
        baseRESTUrl:"http://localhost:8080/educationorganizer"
    }
};

require(['../../../cordova'], function() {

    var bootstrap = {
        initialize: function() {
            document.addEventListener('deviceready', this.onDeviceReady, false);
        },
        onDeviceReady: function() {
            // Detect if iOS 7 or higher and disable overlaying the status bar
            if(window.device && window.device.platform.toLowerCase() == "ios" &&
                parseFloat(window.device.version) >= 7.0) {
                StatusBar.overlaysWebView(false);
                StatusBar.styleDefault();
                StatusBar.backgroundColorByHexString("#e9e9e9");
            }
            // Load the mobile module
            require (["mobile"]);
        }
    };

    bootstrap.initialize();
});
```

Hier die loader.js:

```
//detect the appropriate module to load
define(function () {

    /*
     A simple check on the client. For touch devices or small-resolution screens)
     show the mobile client. By enabling the mobile client on a small-resolution screen
     we allow for testing outside a mobile device (like for example the Mobile Browser
     simulator in JBoss Tools and JBoss Developer Studio).
     */

    var environment;

    if (document.URL.indexOf("mobileapp.html") > -1) {
        environment = "hybrid";
    }
    else if (Modernizr.touch || Modernizr.mq("only all and (max-width: 768px)")) {
        environment = "mobile";
    } else {
        environment = "desktop";
    }

    require([environment]);
});
```

Hier die mobile.js:

```
/**
 * Shortcut alias definitions - will come in handy when declaring dependencies
 * Also, they allow you to keep the code free of any knowledge about library
 * locations and versions
 */
require.config({
    baseUrl:"resources/js",
    paths: {
        jquery:'libs/jquery-2.0.3',
        jquerymobile:'libs/jquery.mobile-1.4.2',
        text:'libs/text',
        underscore:'libs/underscore',
        backbone: 'libs/backbone',
        utilities: 'app/utilities',
        router:'app/router/mobile/router'
    },
    // We shim Backbone.js and Underscore.js since they don't declare AMD modules
    shim: {
        'backbone': {
            deps: ['underscore', 'jquery'],
            exports: 'Backbone'
        },

        'underscore': {
                exports: '_'
        }
    }
});

define("configuration", function() {
    if (window.EducationOrganizer != undefined && EducationOrganizer.config != undefined) {
        return {
            baseUrl: EducationOrganizer.config.baseRESTUrl
        };
    } else {
        return {
            baseUrl: ""
        };
    }
});

define("initializer", [
    'jquery',
    'utilities',
    'text!../templates/mobile/main.html'
], function ($,
             utilities,
             MainTemplate) {
    // Configure jQuery to append timestamps to requests, to bypass browser caches
    // Important for MSIE
                $.ajaxSetup({cache:false});
    $('head').append('<link rel="stylesheet" href="resources/css/jquery.mobile-1.4.2.css"/>');
    $('head').append('<link rel="stylesheet" href="resources/css/m.screen.css"/>');

    // Bind to mobileinit before loading jQueryMobile
    $(document).bind("mobileinit", function () {

        // Prior to creating and starting the router, we disable jQueryMobile's own routing mechanism
        $.mobile.hashListeningEnabled = false;
        $.mobile.linkBindingEnabled = false;
        $.mobile.pushStateEnabled = false;

        // Fix jQueryMobile header and footer positioning issues for iOS.
        // See: https://github.com/jquery/jquery-mobile/issues/4113 and
        // https://github.com/jquery/jquery-mobile/issues/5532
        $(document).on('blur', 'input, textarea, select', function() {
            setTimeout(function() {
            window.scrollTo(document.body.scrollLeft, document.body.scrollTop);
            }, 0);
        });

        utilities.applyTemplate($('body'), MainTemplate);
    });
    // Then (load jQueryMobile and) start the router to finally start the app
    require(['router']);
});

// Now we declare all the dependencies
// This loads and runs the 'initializer' module.
require(['initializer']);
```

Da es sich um eine responsive Applikation zur Administration der Binaris Schulungs-Stammdaten handelt, gibt es beim Start der App einen Redirect in das /admin-Unterverzeichnis, wo sich die Angular.js **MVC-Design Pattern Implementierung** unter Verwendung des **FrontController Design Patterns** befindet.

Hier also die index.html bzw. die mobileapp.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Education Organizer</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
   <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no"/>
   <meta http-equiv="refresh" content="0;url=./admin/app.html#/" />

   <script type="text/javascript" src="resources/js/libs/modernizr-2.6.2.min.js"></script>
   <script type="text/javascript" src="resources/js/libs/require.js"
         data-main="resources/js/configurations/loader"></script>
</head>
<body>
</body>
</html>
```

Die /admin/index.html hat denselben Redirect:

```
<meta http-equiv="refresh" content="0;url=./app.html#/" />
```

Und die Ziel-Seite des Redirects /admin/app.html hat folgenden Inhalt:

```
<!DOCTYPE html>
<html lang="en" ng-app="educationorganizer">
<head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
            <title>Education Organizer</title>
   <link href='http://fonts.googleapis.com/css?family=Rokkitt' rel='stylesheet' type='text/css'/>
   <link href="styles/bootstrap.css" rel="stylesheet" media="screen">
   <link href="styles/bootstrap-theme.css" rel="stylesheet" media="screen">
   <link href="styles/main.css" rel="stylesheet" media="screen">
   <link href="styles/custom-forge.css" rel="stylesheet" media="screen">
</head>
<body>
   <div id="wrap">

         <div id="logo" class="hidden-xs"><div class="wrap"><h1>Education Organizer</h1></div></div>
         <div class="navbar">
         <div class="navbar-header">
            <button type="button" class="navbar-toggle pull-left" data-toggle="collapse" data-target="#navbar-items">
               <span class="glyphicon glyphicon-list"> Links</span>
            </button>
            <button type="button" class="navbar-toggle" data-toggle="offcanvas">
               Create/Change Data <span class="glyphicon glyphicon-th text-right"></span>
            </button>
         </div>

         <!-- Collect the nav links, forms, and other content for toggling -->
         <div id="navbar-items" class="collapse navbar-collapse">
            <ul class="nav navbar-nav">
               <li><a href="../index.html#about">About</a></li>
               <li><a href="../admin/app.html#/Events">Events</a></li>
               <li><a href="../admin/app.html#/EventCategories">EventCategories</a></li>
               <li><a href="../admin/app.html#/MediaItems">MediaItems</a></li>
               <li><a href="../admin/app.html#/Venues">Venues</a></li>
               <li><a href="../admin/app.html#/Sections">Sections</a></li>
            </ul>
         </div>
      </div>

      <div class="container">
         <div class="row row-offcanvas row-offcanvas-left">
            <!-- sidebar -->
            <div class="col-xs-6 col-sm-3 well sidebar-offcanvas">
               <nav class="sidebar-nav" ng-controller="NavController" role="navigation">
                  <div id="sidebar-entries" class="list-group">
                        <a class="list-group-item" ng-class="{active: matchesRoute('Events')}" href="#/Events" data-
toggle="offcanvas">Events</a>
                        <a class="list-group-item" ng-class="{active: matchesRoute('EventCategories')}"
href="#/EventCategories" data-toggle="offcanvas">EventCategories</a>
                        <a class="list-group-item" ng-class="{active: matchesRoute('MediaItems')}" href="#/MediaItems"
data-toggle="offcanvas">MediaItems</a>
                        <a class="list-group-item" ng-class="{active: matchesRoute('Venues')}" href="#/Venues" data-
toggle="offcanvas">Venues</a>
                        <a class="list-group-item" ng-class="{active: matchesRoute('Sections')}" href="#/Sections" data-
toggle="offcanvas">Sections</a>
                  </div>
               </nav>
            </div>
            <!-- main area-->
            <div class="col-sm-offset-1 col-xs-12 col-sm-8 well mainarea">
```

```
                <div id="main" ng-view>
                </div>
            </div>
        </div>
    </div>
</div>

<div id="footer">
    <div class="container">
        <p>Powered by <a href="http://binaris.de">binaris informatik GmbH</a> on JBoss WildFly</p>
    </div>
</div>

<script src="scripts/vendor/modernizr-2.6.2.min.js"></script>
<script src="scripts/vendor/jquery-2.0.3.js"></script>
<script src="scripts/vendor/bootstrap.js"></script>
<script src="scripts/vendor/angular.js"></script>
<script src="scripts/vendor/angular-route.js"></script>
<script src="scripts/vendor/angular-resource.js"></script>
<script src="scripts/app.js"></script>
<script src="scripts/offcanvas.js"></script>
<script src="scripts/filters/startFromFilter.js"></script>
<script src="scripts/filters/genericSearchFilter.js"></script>
<script src="scripts/services/locationParser.js"></script>
<script src="scripts/services/EventFactory.js"></script>
<script src="scripts/controllers/newEventController.js"></script>
<script src="scripts/controllers/searchEventController.js"></script>
<script src="scripts/controllers/editEventController.js"></script>
<script src="scripts/services/EventCategoryFactory.js"></script>
<script src="scripts/controllers/newEventCategoryController.js"></script>
<script src="scripts/controllers/searchEventCategoryController.js"></script>
<script src="scripts/controllers/editEventCategoryController.js"></script>
<script src="scripts/services/MediaItemFactory.js"></script>
<script src="scripts/controllers/newMediaItemController.js"></script>
<script src="scripts/controllers/searchMediaItemController.js"></script>
<script src="scripts/controllers/editMediaItemController.js"></script>
<script src="scripts/services/SectionFactory.js"></script>
<script src="scripts/controllers/newSectionController.js"></script>
<script src="scripts/controllers/searchSectionController.js"></script>
<script src="scripts/controllers/editSectionController.js"></script>
<script src="scripts/services/PaymentCategoryFactory.js"></script>
<script src="scripts/controllers/newPaymentCategoryController.js"></script>
<script src="scripts/controllers/searchPaymentCategoryController.js"></script>
<script src="scripts/controllers/editPaymentCategoryController.js"></script>
<script src="scripts/services/VenueFactory.js"></script>
<script src="scripts/controllers/newVenueController.js"></script>
<script src="scripts/controllers/searchVenueController.js"></script>
<script src="scripts/controllers/editVenueController.js"></script>
</body>
</html>
```

Und die Landing Page der App ‘educationorganizer‘ unter /admin/views/landing.html hat folgenden Inhalt:

```
<h2>
    Binaris Education Organizer running.
</h2>
<p>
    <a target="_blank"
        href="http://binaris-informatik.de/category/architektur/">Documentation</a>
    | <a target="_blank"
        href="http://www.testdrivendevelopment.de/">Schulung Test Driven Development</a> | <a target="_blank"
href="http://www.binaris-education.com/?portfolio=test-driven-development-mit-java">Test Driven Development mit
Java</a> <br /> <a target="_blank" href="http://scrum-experience.de/"><br />
        Schulung Scrum</a> | <a target="_blank"
        href="http://binaris-informatik.de/category/scrum/">Fachartikel
        Scrum</a> | <a target="_blank"
        href="http://www.rheinjug.de/videos/gse.lectures.app/Talk.html#Scrum">Scrum
        Vortrag</a> | <a target="_blank"
        href="http://binaris-informatik.de/category/architektur/">Fachartikel Architektur</a>
</p>
```

Wie in der app.js in den JavaScript imports erkennbar, gibt es unter /admin die folgende Struktur:

```
    /scripts/vendor/modernizr-2.6.2.min.js
    /scripts/vendor/jquery-2.0.3.js
    /scripts/vendor/bootstrap.js
    /scripts/vendor/angular.js
```

```
/scripts/vendor/angular-route.js
/scripts/vendor/angular-resource.js

/scripts/app.js
/scripts/offcanvas.js

/scripts/filters/startFromFilter.js
/scripts/filters/genericSearchFilter.js

/scripts/services/locationParser.js

/scripts/services/EventFactory.js
/scripts/controllers/newEventController.js
/scripts/controllers/searchEventController.js
/scripts/controllers/editEventController.js

/scripts/services/EventCategoryFactory.js
/scripts/controllers/newEventCategoryController.js
/scripts/controllers/searchEventCategoryController.js
/scripts/controllers/editEventCategoryController.js

/scripts/services/MediaItemFactory.js
/scripts/controllers/newMediaItemController.js
/scripts/controllers/searchMediaItemController.js
/scripts/controllers/editMediaItemController.js

/scripts/services/SectionFactory.js
/scripts/controllers/newSectionController.js
/scripts/controllers/searchSectionController.js
/scripts/controllers/editSectionController.js

/scripts/services/PaymentCategoryFactory.js
/scripts/controllers/newPaymentCategoryController.js
/scripts/controllers/searchPaymentCategoryController.js
/scripts/controllers/editPaymentCategoryController.js

/scripts/services/VenueFactory.js
/scripts/controllers/newVenueController.js
/scripts/controllers/searchVenueController.js
/scripts/controllers/editVenueController.js
```

Somit gibt es für jedes Frontend Model erkennbar eine Factory, z.B. die EventFactory.js:

```
angular.module('educationorganizer').factory('EventResource', function($resource){
    var resource =
$resource('../rest/forge/events/:EventId',{EventId:'@id'},{'queryAll':{method:'GET',isArray:true},'query':{method:'GET',isArray:false},'update':{method:'PUT'}});
    return resource;
});
```

Diese Factory befüllt beim Lesen per queryAll oder query-Methode die Model-Daten für das Event Object aus den entsprechenden RESTful Webservice-Aufrufen und übergibt beim Update per 'update'-Methode die Event Model-Daten an den entsprechenden REST Webservice-Aufruf unter Verwendung der http-PUT Methode.

Das Mapping zwischen dem entsprechenden Click-Event auf eine Schaltfläche 'Cancel', 'Save' oder 'Delete' erfolgt unter Verwendung des Command DesignPatterns in der editEventController.js, bzw. der newEventController.js oder der searchEventController.js. und unter Verwendung des Router DesignPatterns

in der /admin/scripts/app.js:

```
'use strict';

angular.module('educationorganizer',['ngRoute','ngResource'])
  .config(['$routeProvider', function($routeProvider) {
    $routeProvider

.when('/',{templateUrl:'views/landing.html',controller:'LandingPageController'})
.when('/Events',{templateUrl:'views/Event/search.html',controller:'SearchEventController'})
.when('/Events/new',{templateUrl:'views/Event/detail.html',controller:'NewEventController'})
.when('/Events/edit/:EventId',{templateUrl:'views/Event/detail.html',controller:'EditEventController'})
.when('/EventCategories',{templateUrl:'views/EventCategory/search.html',controller:'SearchEventCategoryController'})
.when('/EventCategories/new',{templateUrl:'views/EventCategory/detail.html',controller:'NewEventCategoryController'})
.when('/EventCategories/edit/:EventCategoryId',{templateUrl:'views/EventCategory/detail.html',controller:'EditEventCategoryController'})
.when('/MediaItems',{templateUrl:'views/MediaItem/search.html',controller:'SearchMediaItemController'})
.when('/MediaItems/new',{templateUrl:'views/MediaItem/detail.html',controller:'NewMediaItemController'})
.when('/MediaItems/edit/:MediaItemId',{templateUrl:'views/MediaItem/detail.html',controller:'EditMediaItemController'})
.when('/Sections',{templateUrl:'views/Section/search.html',controller:'SearchSectionController'})
.when('/Sections/new',{templateUrl:'views/Section/detail.html',controller:'NewSectionController'})
.when('/Sections/edit/:SectionId',{templateUrl:'views/Section/detail.html',controller:'EditSectionController'})
.when('/Venues',{templateUrl:'views/Venue/search.html',controller:'SearchVenueController'})
.when('/Venues/new',{templateUrl:'views/Venue/detail.html',controller:'NewVenueController'})
.when('/Venues/edit/:VenueId',{templateUrl:'views/Venue/detail.html',controller:'EditVenueController'})

.otherwise({
  redirectTo: '/'
});

  }])
  .controller('LandingPageController', function LandingPageController() {
  })
  .controller('NavController', function NavController($scope, $location) {
    $scope.matchesRoute = function(route) {
      var path = $location.path();
      return (path === ("/" + route) || path.indexOf("/" + route + "/") == 0);
    };
  });
```

Hier die /scripts/controllers/newEventController.js:

```
angular.module('educationorganizer').controller('NewEventController', function ($scope, $location, locationParser,
EventResource , MediaItemResource, EventCategoryResource) {
  $scope.disabled = false;
  $scope.$location = $location;
  $scope.event = $scope.event || {};

  $scope.mediaItemList = MediaItemResource.queryAll(function(items){
    $scope.mediaItemSelectionList = $.map(items, function(item) {
      return ( {
        value : item.id,
        text : item.url
      });
    });
  });
  $scope.$watch("mediaItemSelection", function(selection) {
    if ( typeof selection != 'undefined') {
      $scope.event.mediaItem = {};
      $scope.event.mediaItem.id = selection.value;
    }
  });

  $scope.categoryList = EventCategoryResource.queryAll(function(items){
    $scope.categorySelectionList = $.map(items, function(item) {
      return ( {
        value : item.id,
        text : item.description
      });
    });
  });
  $scope.$watch("categorySelection", function(selection) {
    if ( typeof selection != 'undefined') {
      $scope.event.category = {};
      $scope.event.category.id = selection.value;
    }
  });

  $scope.save = function() {
    var successCallback = function(data,responseHeaders){
      var id = locationParser(responseHeaders);
      $location.path('/Events/edit/' + id);
      $scope.displayError = false;
    };
    var errorCallback = function() {
      $scope.displayError = true;
    };
    EventResource.save($scope.event, successCallback, errorCallback);
  };

  $scope.cancel = function() {
    $location.path("/Events");
  };
});
```

Hier die /scripts/controllers/editEventController.js

```
angular.module('educationorganizer').controller('EditEventController', function($scope, $routeParams, $location,
EventResource , MediaItemResource, EventCategoryResource) {
    var self = this;
    $scope.disabled = false;
    $scope.$location = $location;

    $scope.get = function() {
        var successCallback = function(data){
            self.original = data;
            $scope.event = new EventResource(self.original);
            MediaItemResource.queryAll(function(items) {
                $scope.mediaItemSelectionList = $.map(items, function(item) {
                    var wrappedObject = {
                        id : item.id
                    };
                    var labelObject = {
                        value : item.id,
                        text : item.url
                    };
                    if($scope.event.mediaItem && item.id == $scope.event.mediaItem.id) {
                        $scope.mediaItemSelection = labelObject;
                        $scope.event.mediaItem = wrappedObject;
                        self.original.mediaItem = $scope.event.mediaItem;
                    }
                    return labelObject;
                });
            });
            EventCategoryResource.queryAll(function(items) {
                $scope.categorySelectionList = $.map(items, function(item) {
                    var wrappedObject = {
                        id : item.id
                    };
                    var labelObject = {
                        value : item.id,
                        text : item.description
                    };
                    if($scope.event.category && item.id == $scope.event.category.id) {
                        $scope.categorySelection = labelObject;
                        $scope.event.category = wrappedObject;
                        self.original.category = $scope.event.category;
                    }
                    return labelObject;
                });
            });
        };
        var errorCallback = function() {
            $location.path("/Events");
        };
        EventResource.get({EventId:$routeParams.EventId}, successCallback, errorCallback);
    };

    $scope.isClean = function() {
        return angular.equals(self.original, $scope.event);
    };

    $scope.save = function() {
        var successCallback = function(){
            $scope.get();
            $scope.displayError = false;
        };
        var errorCallback = function() {
            $scope.displayError=true;
        };
        $scope.event.$update(successCallback, errorCallback);
    };

    $scope.cancel = function() {
        $location.path("/Events");
    };

    $scope.remove = function() {
        var successCallback = function() {
            $location.path("/Events");
            $scope.displayError = false;
        };
        var errorCallback = function() {
            $scope.displayError=true;
        };
        $scope.event.$remove(successCallback, errorCallback);
    };

    $scope.$watch("mediaItemSelection", function(selection) {
        if (typeof selection != 'undefined') {
            $scope.event.mediaItem = {};
            $scope.event.mediaItem.id = selection.value;
        }
    });
    $scope.$watch("categorySelection", function(selection) {
        if (typeof selection != 'undefined') {
            $scope.event.category = {};
            $scope.event.category.id = selection.value;
        }
    });

    $scope.get();
});
```

Und hier die /scripts/controllers/searchEventController.js:

```
angular.module('educationorganizer').controller('SearchEventController', function($scope, $http, EventResource ,
MediaItemResource, EventCategoryResource) {

    $scope.search={};
    $scope.currentPage = 0;
    $scope.pageSize= 10;
    $scope.searchResults = [];
    $scope.filteredResults = [];
    $scope.pageRange = [];
    $scope.numberOfPages = function() {
        var result = Math.ceil($scope.filteredResults.length/$scope.pageSize);
        var max = (result == 0) ? 1 : result;
        $scope.pageRange = [];
        for(var ctr=0;ctr<max;ctr++) {
            $scope.pageRange.push(ctr);
        }
        return max;
    };
    $scope.mediaItemList = MediaItemResource.queryAll();
    $scope.categoryList = EventCategoryResource.queryAll();

    $scope.performSearch = function() {
        $scope.searchResults = EventResource.queryAll(function(){
            $scope.numberOfPages();
        });
    };

    $scope.previous = function() {
       if($scope.currentPage > 0) {
           $scope.currentPage--;
       }
    };

    $scope.next = function() {
       if($scope.currentPage < ($scope.numberOfPages() - 1) ) {
           $scope.currentPage++;
       }
    };

    $scope.setPage = function(n) {
       $scope.currentPage = n;
    };

    $scope.performSearch();
});
```

Und der /scripts/filters/genericSearchFilter.js:

```
'use strict';

angular.module('educationorganizer').filter('searchFilter', function() {

    function matchObjectProperties(expectedObject, actualObject) {
        var flag = true;
        for(var key in expectedObject) {

            if (expectedObject.hasOwnProperty(key)) {
                var expectedProperty = expectedObject[key];
                if (expectedProperty == null || expectedProperty === "") {
                    continue;
                }
                var actualProperty = actualObject[key];

                if (angular.isUndefined(actualProperty)) {
                    continue;
                }
                if (actualProperty == null) {
                    flag = false;
                } else if (angular.isObject(expectedProperty)) {
                    flag = flag && matchObjectProperties(expectedProperty, actualProperty);
                } else {
                    flag = flag && (actualProperty.toString().indexOf(expectedProperty.toString()) != -1);
                }
            }
        }
        return flag;
    }

    return function(results) {

        this.filteredResults = [];
        for (var ctr = 0; ctr < results.length; ctr++) {
            var flag = true;
            var searchCriteria = this.search;
            var result = results[ctr];

            for (var key in searchCriteria) {
                if (searchCriteria.hasOwnProperty(key)) {
                    var expected = searchCriteria[key];
                    if (expected == null || expected === "") {
                        continue;
                    }
                    var actual = result[key];
                    if (actual == null) {
                        flag = false;
                    } else if (angular.isObject(expected)) {
                        flag = flag && matchObjectProperties(expected, actual);
                    } else {
                        flag = flag && (actual.toString().indexOf(expected.toString()) != -1);
                    }
                }
            }
            if (flag == true) {
                this.filteredResults.push(result);
            }
        }
        this.numberOfPages();
        return this.filteredResults;
    };
});
```

b) Der REST-Webservice:

Über den Einsatz von JBoss RESTEasy für RESTful Webservices für SPAs gibt es bereits hier einen Blog-Eintrag in diesem Blog hier. Die Aktivierung des RESTEasy-Webservices geschieht dort (Servlet 2.3, 2,4 oder 2.5) über die web.xml:

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <context-param>
    <param-name>javax.ws.rs.core.Application</param-name>
    <param-value>de.binaris.rest.samples.service.SumOfMultiplesApplication</param-value>
  </context-param>

  <context-param>
    <param-name>resteasy.servlet.mapping.prefix</param-name>
    <param-value>/resteasy</param-value>
  </context-param>
  <listener>
    <listener-class>
      org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap
    </listener-class>
  </listener>

  <servlet>
    <servlet-name>Resteasy</servlet-name>
    <servlet-class>
      org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
    </servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Resteasy</servlet-name>
    <url-pattern>/resteasy/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Stattdessen wird nun die folgende Klasse JaxRsActivator verwendet, die von Application abgeleitet ist und mit der **@ApplicationPath** Annotation annotiert ist, da seit **servlet-3.0** keine web.xml mehr benötigt wird, sondern alles über Annotationen deklarierbar ist:

```
package de.binaris.educationorganizer.rest;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("/rest")
public class JaxRsActivator extends Application {
   /* neither extended class attributes nor methods required */
}
```

Hier als Beispiel die REST-Webservice Facade als Stateless EJB zum Bearbeiten eines Venues:

```
package de.binaris.educationorganizer.rest;

import java.util.ArrayList;
import java.util.List;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.OptimisticLockException;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;
import javax.ws.rs.core.UriBuilder;

import de.binaris.educationorganizer.model.Venue;
import de.binaris.educationorganizer.rest.dto.VenueDTO;

/**
 * The Venues REST Service Endpoint
 */
@Stateless
@Path("forge/venues")
public class VenueEndpoint
{
  @PersistenceContext(unitName = "educationorganizer")
  private EntityManager em;

  @POST
  @Consumes("application/json")
  public Response create(VenueDTO dto) {
   Venue entity = dto.fromDTO(null, em);
   em.persist(entity);
   return Response.created(UriBuilder.fromResource(VenueEndpoint.class).
                              path(String.valueOf(entity.getId())).build()).build();
}

@DELETE
@Path("/{id:[0-9][0-9]*}")
public Response deleteById(@PathParam("id") Long id)
{
   Venue entity = em.find(Venue.class, id);
   if (entity == null) {
      return Response.status(Status.NOT_FOUND).build();
   }
   em.remove(entity);
   return Response.noContent().build();
}

@GET
@Path("/{id:[0-9][0-9]*}")
@Produces("application/json")
public Response findById(@PathParam("id") Long id)
{
   TypedQuery<Venue> findByIdQuery = em.createQuery("
                    SELECT DISTINCT v FROM Venue v LEFT JOIN
                              FETCH v.sections LEFT JOIN FETCH v.mediaItem
                              WHERE v.id = :entityId ORDER BY v.id", Venue.class);
   findByIdQuery.setParameter("entityId", id);
   Venue entity;
   try {
      entity = findByIdQuery.getSingleResult();
   }
   catch (NoResultException nre) {
      entity = null;
   }
   if (entity == null) {
      return Response.status(Status.NOT_FOUND).build();
   }
   VenueDTO dto = new VenueDTO(entity);
   return Response.ok(dto).build();
}

@GET
@Produces("application/json")
public List<VenueDTO> listAll(@QueryParam("start") Integer startPosition,
                                    @QueryParam("max") Integer maxResult)
{
   TypedQuery<Venue> findAllQuery = em.createQuery(
                              "SELECT DISTINCT v FROM Venue v
                                    LEFT JOIN FETCH v.sections
                                    LEFT JOIN FETCH v.mediaItem
                                    ORDER BY v.id", Venue.class);
   if (startPosition != null) {
      findAllQuery.setFirstResult(startPosition);
   }
   if (maxResult != null) {
      findAllQuery.setMaxResults(maxResult);
   }
   final List<Venue> searchResults = findAllQuery.getResultList();
   final List<VenueDTO> results = new ArrayList<VenueDTO>();

   for (Venue searchResult : searchResults) {
      VenueDTO dto = new VenueDTO(searchResult);
      results.add(dto);
   }
   return results;
}

@PUT
@Path("/{id:[0-9][0-9]*}")
@Consumes("application/json")
public Response update(@PathParam("id") Long id, VenueDTO dto)
{
   TypedQuery<Venue> findByIdQuery = em.createQuery(
                              "SELECT DISTINCT v FROM Venue v
                                    LEFT JOIN FETCH v.sections
                                    LEFT JOIN FETCH v.mediaItem
```

```
                                        WHERE v.id = :entityId ORDER BY v.id", Venue.class);
    findByIdQuery.setParameter("entityId", id);
    Venue entity;
    try {
      entity = findByIdQuery.getSingleResult();
    }
    catch (NoResultException nre) {
      entity = null;
    }
    entity = dto.fromDTO(entity, em);
    try {
      entity = em.merge(entity);
    }
    catch (OptimisticLockException e) {
      return Response.status(Response.Status.CONFLICT).entity(e.getEntity()).build();
    }
    return Response.noContent().build();
  }
}
```

Unter Verwendung der folgenden DTOs (Data Transfer Objects):

1.) VenueDTO:

```
package de.binaris.educationorganizer.rest.dto;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import javax.persistence.EntityManager;
import javax.xml.bind.annotation.XmlRootElement;

import de.binaris.educationorganizer.model.Section;
import de.binaris.educationorganizer.model.Venue;

@XmlRootElement
public class VenueDTO implements Serializable {

	private static final long serialVersionUID = 7507622024291054277L;

	private Long id;
	private String name;
	private AddressDTO address;
	private NestedMediaItemDTO mediaItem;
	private String description;
	private Set<NestedSectionDTO> sections = new HashSet<NestedSectionDTO>();
	private int capacity;

	public VenueDTO() {
	}

	public VenueDTO(final Venue entity) {
		if (entity != null) {
			this.id = entity.getId();
			this.name = entity.getName();
			this.address = new AddressDTO(entity.getAddress());
			this.mediaItem = new NestedMediaItemDTO(entity.getMediaItem());
			this.description = entity.getDescription();
			Iterator<Section> iterSections = entity.getSections().iterator();
			while (iterSections.hasNext()) {
				Section element = iterSections.next();
				this.sections.add(new NestedSectionDTO(element));
			}
			this.capacity = entity.getCapacity();
		}
	}

	public Venue fromDTO(Venue entity, EntityManager em) {
		if (entity == null) {
			entity = new Venue();
		}
		entity.setName(this.name);
		if (this.address != null) {
			entity.setAddress(this.address.fromDTO(entity.getAddress(), em));
		}
		if (this.mediaItem != null) {
			entity.setMediaItem(this.mediaItem.fromDTO(entity.getMediaItem(),
					em));
		}
		entity.setDescription(this.description);
		Iterator<Section> iterSections = entity.getSections().iterator();
		while (iterSections.hasNext()) {
			boolean found = false;
			Section section = iterSections.next();
			Iterator<NestedSectionDTO> iterDtoSections = this.getSections()
					.iterator();
			while (iterDtoSections.hasNext()) {
				NestedSectionDTO dtoSection = iterDtoSections.next();
				if (dtoSection.getId().equals(section.getId())) {
					found = true;
					break;
				}
			}
			if (found == false) {
				iterSections.remove();
				em.remove(section);
			}
		}
		Iterator<NestedSectionDTO> iterDtoSections = this.getSections()
				.iterator();
		while (iterDtoSections.hasNext()) {
			boolean found = false;
			NestedSectionDTO dtoSection = iterDtoSections.next();
			iterSections = entity.getSections().iterator();
			while (iterSections.hasNext()) {
				Section section = iterSections.next();
				if (dtoSection.getId().equals(section.getId())) {
					found = true;
					break;
				}
			}
			if (found == false) {
				Iterator<Section> resultIter =
					em.createQuery("SELECT DISTINCT s FROM Section s",
							Section.class).getResultList().iterator();
				while (resultIter.hasNext()) {
					Section result = resultIter.next();
					if (result.getId().equals(dtoSection.getId())) {
						entity.getSections().add(result);
						break;
					}
				}
			}
		}
		entity.setCapacity(this.capacity);
		entity = em.merge(entity);
		return entity;
	}

	public Long getId() {
		return this.id;
	}

	public void setId(final Long id) {
		this.id = id;
	}

	public String getName() {
		return this.name;
	}

	public void setName(final String name) {
		this.name = name;
	}

	public AddressDTO getAddress() {
		return this.address;
	}

	public void setAddress(final AddressDTO address) {
		this.address = address;
	}
	public NestedMediaItemDTO getMediaItem() {
		return this.mediaItem;
	}

	public void setMediaItem(final NestedMediaItemDTO mediaItem) {
		this.mediaItem = mediaItem;
	}

	public String getDescription() {
		return this.description;
	}

	public void setDescription(final String description) {
		this.description = description;
	}

	public Set<NestedSectionDTO> getSections() {
		return this.sections;
	}

	public void setSections(final Set<NestedSectionDTO> sections) {
		this.sections = sections;
	}

	public int getCapacity() {
		return this.capacity;
	}

	public void setCapacity(final int capacity) {
		this.capacity = capacity;
	}
}
```

2.) NestedVenueDTO (welches wiederum das AddressDTO verwendet) :

```
package de.binaris.educationorganizer.rest.dto;

import java.io.Serializable;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import de.binaris.educationorganizer.model.Venue;

public class NestedVenueDTO implements Serializable {

	private static final long serialVersionUID = 7022670393595713049L;

	private Long id;
	private String name;
	private AddressDTO address;
	private String description;
	private int capacity;

	public NestedVenueDTO() {
	}

	public NestedVenueDTO(final Venue entity) {
		if (entity != null) {
			this.id = entity.getId();
			this.name = entity.getName();
			this.address = new AddressDTO(entity.getAddress());
			this.description = entity.getDescription();
			this.capacity = entity.getCapacity();
		}
	}

	public Venue fromDTO(Venue entity, EntityManager em) {
		if (entity == null) {
			entity = new Venue();
		}
		if (this.id != null) {
			TypedQuery<Venue> findByIdQuery = em.createQuery(
					"SELECT DISTINCT v FROM Venue v WHERE v.id = :entityId",
					Venue.class);
			findByIdQuery.setParameter("entityId", this.id);
```

```
				try {
					entity = findByIdQuery.getSingleResult();
				} catch (javax.persistence.NoResultException nre) {
					entity = null;
				}
				return entity;
			}
			entity.setName(this.name);
			if (this.address != null) {
				entity.setAddress(this.address.fromDTO(entity.getAddress(), em));
			}
			entity.setDescription(this.description);
			entity.setCapacity(this.capacity);
			entity = em.merge(entity);
			return entity;
		}

		public Long getId() {
			return this.id;
		}

		public void setId(final Long id) {
			this.id = id;
		}

		public String getName() {
			return this.name;
		}

		public void setName(final String name) {
			this.name = name;
		}

		public AddressDTO getAddress() {
			return this.address;
		}

		public void setAddress(final AddressDTO address) {
			this.address = address;
		}

		public String getDescription() {
			return this.description;
		}

		public void setDescription(final String description) {
			this.description = description;
		}

		public int getCapacity() {
			return this.capacity;
		}

		public void setCapacity(final int capacity) {
			this.capacity = capacity;
		}
}
```

3.) AddressDTO:

```
package de.binaris.educationorganizer.rest.dto;

import java.io.Serializable;

import javax.persistence.EntityManager;

import de.binaris.educationorganizer.model.Address;

public class AddressDTO implements Serializable {

		private static final long serialVersionUID = 7840652372873114375L;

		private String street;
		private String city;
		private String country;

		public AddressDTO() {
	}

	public AddressDTO(final Address entity) {
		if (entity != null) {
			this.street = entity.getStreet();
			this.city = entity.getCity();
			this.country = entity.getCountry();
		}
	}

	public Address fromDTO(Address entity, EntityManager em) {
		if (entity == null) {
			entity = new Address();
		}
		entity.setStreet(this.street);
		entity.setCity(this.city);
		entity.setCountry(this.country);
		return entity;
	}

	public String getStreet() {
		return this.street;
	}

	public void setStreet(final String street) {
		this.street = street;
	}

	public String getCity() {
		return this.city;
	}

	public void setCity(final String city) {
		this.city = city;
	}

	public String getCountry() {
		return this.country;
	}

	public void setCountry(final String country) {
		this.country = country;
	}
}
```

4.) NestedSectionDTO:

```
package de.binaris.educationorganizer.rest.dto;

import java.io.Serializable;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import de.binaris.educationorganizer.model.Section;

public class NestedSectionDTO implements Serializable {

	private static final long serialVersionUID = 7551319672738307511L;

	private Long id;
	private String name;
	private String description;
	private int numberOfRows;
	private int rowCapacity;
	private int capacity;

	public NestedSectionDTO() {
	}

	public NestedSectionDTO(final Section entity) {
		if (entity != null) {
			this.id = entity.getId();
			this.name = entity.getName();
			this.description = entity.getDescription();
			this.numberOfRows = entity.getNumberOfRows();
			this.rowCapacity = entity.getRowCapacity();
```

```
                        this.capacity = entity.getCapacity();
                }
        }

        public Section fromDTO(Section entity, EntityManager em) {
                if (entity == null) {
                        entity = new Section();
                }
                if (this.id != null) {
                        TypedQuery<Section> findByIdQuery = em.createQuery(
                                        "SELECT DISTINCT s FROM Section s WHERE s.id = :entityId",
                                        Section.class);
                        findByIdQuery.setParameter("entityId", this.id);
                        try {
                                entity = findByIdQuery.getSingleResult();
                        } catch (javax.persistence.NoResultException nre) {
                                entity = null;
                        }
                        return entity;
                }
                entity.setName(this.name);
                entity.setDescription(this.description);
                entity.setNumberOfRows(this.numberOfRows);
                entity.setRowCapacity(this.rowCapacity);
                entity = em.merge(entity);
                return entity;
        }

        public Long getId() {
                return this.id;
        }

        public void setId(final Long id) {
                this.id = id;
        }

        public String getName() {
                return this.name;
        }

        public void setName(final String name) {
                this.name = name;
        }

        public String getDescription() {
                return this.description;
        }

        public void setDescription(final String description) {
                this.description = description;
        }

        public int getNumberOfRows() {
                return this.numberOfRows;
        }

        public void setNumberOfRows(final int numberOfRows) {
                this.numberOfRows = numberOfRows;
        }

        public int getRowCapacity() {
                return this.rowCapacity;
        }

        public void setRowCapacity(final int rowCapacity) {
                this.rowCapacity = rowCapacity;
        }

        public int getCapacity() {
                return this.capacity;
        }

        public void setCapacity(final int capacity) {
                this.capacity = capacity;
        }
}
```

5.) NestedMediaItemDTO:

```
package de.binaris.educationorganizer.rest.dto;

import java.io.Serializable;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import de.binaris.educationorganizer.model.MediaItem;
import de.binaris.educationorganizer.model.MediaType;

public class NestedMediaItemDTO implements Serializable {

	private static final long serialVersionUID = 7626149832232284688L;

	private Long id;
	private MediaType mediaType;
	private String url;

	public NestedMediaItemDTO() {
	}

	public NestedMediaItemDTO(final MediaItem entity) {
		if (entity != null) {
			this.id = entity.getId();
			this.mediaType = entity.getMediaType();
			this.url = entity.getUrl();
		}
	}

	public MediaItem fromDTO(MediaItem entity, EntityManager em) {
		if (entity == null) {
			entity = new MediaItem();
		}
		if (this.id != null) {
			TypedQuery<MediaItem> findByIdQuery = em
				.createQuery(
				"SELECT DISTINCT m FROM MediaItem m WHERE m.id = :entityId",
				MediaItem.class);
			findByIdQuery.setParameter("entityId", this.id);
			try {
				entity = findByIdQuery.getSingleResult();
			} catch (javax.persistence.NoResultException nre) {
				entity = null;
			}
			return entity;
		}
		entity.setMediaType(this.mediaType);
		entity.setUrl(this.url);
		entity = em.merge(entity);
		return entity;
	}

	public Long getId() {
		return this.id;
	}

	public void setId(final Long id) {
		this.id = id;
	}

	public MediaType getMediaType() {
		return this.mediaType;
	}

	public void setMediaType(final MediaType mediaType) {
		this.mediaType = mediaType;
	}

	public String getUrl() {
		return this.url;
	}

	public void setUrl(final String url) {
		this.url = url;
	}
}
```

Hier noch eine **generischere Service Facade zum Listen** aller Venues über die folgenden REST HttpRequests:

GET /subclassname
GET /subclassname/:id
GET /subclassname/count

Der VenueService:

```
package de.binaris.educationorganizer.rest;

import javax.ejb.Stateless;
import javax.ws.rs.Path;

import de.binaris.educationorganizer.model.Venue;

/**
 * <p>
 *    A JAX-RS endpoint for handling {@link Venue}s. Inherits the actual
 *    methods from {@link BaseEntityService}.
 *
 *    This is a stateless service, declared as an EJB for transaction demarcation
 * </p>
 */
@Path("/venues")
@Stateless
public class VenueService extends BaseEntityService<Venue> {

   public VenueService() {
      super(Venue.class);
   }
}
```

Mit der Basisklasse **BaseEntityService**, von der beliebige Service Fassaden ableiten können:

```
package de.binaris.educationorganizer.rest;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.MultivaluedMap;
import javax.ws.rs.core.UriInfo;

/**
 * <p>
 *   A number of RESTful services implement GET operations
 *   on a particular type of entity. To follow the DRY principle,
 *   the generic operations are implemented in the <code>BaseEntityService</code>
 *   class, and the other services can inherit from here.
 *
 *   Subclasses will declare a base path using the JAX-RS {@link Path} annotation,
 *   for example: will support the following methods:
 *
 * <pre>
 * <code>
 *   GET /subclassname
 *   GET /subclassname/:id
 *   GET /subclassname/count
 * </code>
 * </pre>
 *
 *   Subclasses may specify various criteria for filtering entities
 *  when retrieving a list of them, by supporting custom query parameters.
 *  Pagination is supported by default through the query parameters <code>first</code>
 *  and <code>maxResults</code>.
 *
 *  The class is abstract because it is not intended to be used directly,
 *  but subclassed by actual JAX-RS endpoints.
 * </p>
 */
public abstract class BaseEntityService<T> {

   @Inject
   private EntityManager entityManager;

   private Class<T> entityClass;

   public BaseEntityService() {}

   public BaseEntityService(Class<T> entityClass) {
      this.entityClass = entityClass;
   }

   public EntityManager getEntityManager() {
      return entityManager;
   }

   /**
    * <p>
    *   A method for retrieving all entities of a given type.
    *   Supports the query parameters <code>first</code>
    *   and <code>maxResults</code> for pagination.
    * </p>
    *
    * @param uriInfo application and request context information
    *                               (see {@see UriInfo} class information for more details)
    * @return
    */
   @GET
   @Produces(MediaType.APPLICATION_JSON)
   public List<T> getAll(@Context UriInfo uriInfo) {
      return getAll(uriInfo.getQueryParameters());
   }

   public List<T> getAll(MultivaluedMap<String, String> queryParameters) {
      final CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
      final CriteriaQuery<T> criteriaQuery = criteriaBuilder.createQuery(entityClass);
      Root<T> root = criteriaQuery.from(entityClass);
      Predicate[] predicates = extractPredicates(queryParameters, criteriaBuilder, root);
      criteriaQuery.select(criteriaQuery.getSelection()).where(predicates);
      criteriaQuery.orderBy(criteriaBuilder.asc(root.get("id")));
      TypedQuery<T> query = entityManager.createQuery(criteriaQuery);
      if (queryParameters.containsKey("first")) {
            Integer firstRecord = Integer.parseInt(queryParameters.getFirst("first"))-1;
            query.setFirstResult(firstRecord);
      }
      if (queryParameters.containsKey("maxResults")) {
            Integer maxResults = Integer.parseInt(queryParameters.getFirst("maxResults"));
            query.setMaxResults(maxResults);
      }
                        return query.getResultList();
   }

   /**
    * <p>
    *   A method for counting all entities of a given type
    * </p>
    *
    * @param uriInfo application and request context information
    *                               (see {@see UriInfo} class information for more details)
    * @return
    */
   @GET
   @Path("/count")
   @Produces(MediaType.APPLICATION_JSON)
   public Map<String, Long> getCount(@Context UriInfo uriInfo) {
      CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
      CriteriaQuery<Long> criteriaQuery = criteriaBuilder.createQuery(Long.class);
      Root<T> root = criteriaQuery.from(entityClass);
```

```
        criteriaQuery.select(criteriaBuilder.count(root));
        Predicate[] predicates = extractPredicates(uriInfo.getQueryParameters(), criteriaBuilder, root);
        criteriaQuery.where(predicates);
        Map<String, Long> result = new HashMap<String, Long>();
        result.put("count", entityManager.createQuery(criteriaQuery).getSingleResult());
        return result;
    }

    /**
     * <p>
     *    Subclasses may choose to expand the set of supported
     *    query parameters (for adding more filtering
     *    criteria on search and count) by overriding this method.
     * </p>
     * @param queryParameters - the HTTP query parameters received by the endpoint
     * @param criteriaBuilder - @{link CriteriaBuilder} used by the invoker
     * @param root  @{link Root} used by the invoker
     * @return a list of {@link Predicate}s that will added as query parameters
     */
    protected Predicate[] extractPredicates(MultivaluedMap<String, String> queryParameters,
                                            CriteriaBuilder criteriaBuilder, Root<T> root) {
        return new Predicate[]{};
    }

    /**
     * <p>
     *    A method for retrieving individual entity instances.
     * </p>
     * @param id entity id
     * @return
     */
    @GET
    @Path("/{id:[0-9][0-9]*}")
    @Produces(MediaType.APPLICATION_JSON)
    public T getSingleInstance(@PathParam("id") Long id) {
        final CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
        final CriteriaQuery<T> criteriaQuery = criteriaBuilder.createQuery(entityClass);
        Root<T> root = criteriaQuery.from(entityClass);
        Predicate condition = criteriaBuilder.equal(root.get("id"), id);
        criteriaQuery.select(criteriaBuilder.createQuery(entityClass).getSelection()).where(condition);
        return entityManager.createQuery(criteriaQuery).getSingleResult();
    }
}
```

c) Das Service Backend:

Gerade die beispielhaft beschriebene Venue Entity hat also einige Abhängigkeiten zu anderen Entities, deren Attribute bzw. Funktionalitäten sie über die folgenden relational aggregierten Entities bzw. Embeddables integriert und verwendet:

- Address (Embedded)
- MediaItem
- Section

Die Beispiel-Applikation verwendet weiterhin die folgenden Entities, die hier auch aufgeführt werden sollen:

- MediaType
- Event
- EventCategory

Die Relationen und Constraints (HibernateValidation) zeigen sich auch in den entsprechenden JPA 2.1 Entities und deren Annotationen. Hier beispielhaft die von der Applikation ‘educationorganizer‘ verwendeten Entities:

1. Venue:

```
package de.binaris.educationorganizer.model;

import static javax.persistence.CascadeType.MERGE;
import static javax.persistence.FetchType.EAGER;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.SequenceGenerator;

import org.hibernate.validator.constraints.NotEmpty;

/**
 * <p>
 * Represents a single venue
 * </p>
 */
@Entity
public class Venue implements Serializable {

    private static final long serialVersionUID = 7518459985117709991L;

    /**
     * The ID of the Venue.
     */
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    /**
     * <p>
     * The name of the venue.
     *
     * The name of the event forms it's natural identity and cannot be shared between venues.
     *
     * The name must not be null and must be one or more characters, since
     * the Bean Validation constraint <code>@NotEmpty</code> enforces this.
     * </p>
     */
    @Column(unique = true)
    @NotEmpty
    private String name;

    /**
     * The address of the venue
     */
    private Address address = new Address();

    /**
     * A description of the venue
     */
    private String description;

    /**
     * <p>
     * A set of sections in the venue
     *
     * The <code>@OneToMany<code> JPA mapping establishes this relationship.
     * EAGER fetching means the related objects will be fetched immediately
     * the moment a venue is fetched.
     * This relationship is bi-directional (a section knows which venue it is part of),
     * and the <code>mappedBy</code>
     * attribute establishes this. We cascade MERGE persistence operations
     * to the set of sections, so, for example if a venue is updated,
     * it's sections will also be updated.
     * </p>
     */
    @OneToMany(cascade = MERGE, fetch = EAGER, mappedBy = "venue")
    private Set<Section> sections = new HashSet<Section>();

    /**
     * The capacity of the venue
     */
    private int capacity;

    /**
     * An optional media item entity points to the venue too.
     * The <code>@ManyToOne</code> establishes the relationship.
     */
    @ManyToOne
    private MediaItem mediaItem;
```

```
/* getters and setters */

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Address getAddress() {
    return address;
}

public void setAddress(Address address) {
    this.address = address;
}

public MediaItem getMediaItem() {
    return mediaItem;
}

public void setMediaItem(MediaItem description) {
    this.mediaItem = description;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public Set<Section> getSections() {
    return sections;
}

public void setSections(Set<Section> sections) {
    this.sections = sections;
}

public int getCapacity() {
    return capacity;
}

public void setCapacity(int capacity) {
    this.capacity = capacity;
}

/* toString(), equals() and hashCode() for Venue, using the natural identity of the object */

@Override
public boolean equals(Object object) {
    if (!(object instanceof Venue)) {
        return false;
    }
    Venue castOther = (Venue) object;
    return id != null ? id.equals(castOther.getId()) : false;
}

@Override
public int hashCode() {
    return id != null ? id.hashCode() : System.identityHashCode(this);
}

@Override
public String toString() {
    return name;
    }
}
```

2. Section:

```
package de.binaris.educationorganizer.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.NotEmpty;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

/**
 * <p>
 * A section is a specific area within a venue layout.
 * A venue layout may consist of multiple sections.
*
* The name and venue_id form the natural id of this entity,
 * and therefore must be unique. JPA requires us to use the class level
 * <code>@Table</code> constraint.
 * </p>
 */
@Entity
@Table(uniqueConstraints=@UniqueConstraint(columnNames={"name", "venue_id"}))
@JsonIgnoreProperties("venue")
public class Section implements Serializable {

    private static final long serialVersionUID = 5014811055691416915L;

    /**
     * The ID of the Section.
     */
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    /**
     * <p>
     * The short name of the section, may be a code such as A12, G7, etc.
     * The <code>@NotEmpty<code> Bean Validation constraint means
     * that the section name must contain at least 1 character.
     * </p>
     */
    @NotEmpty
    private String name;

    /**
     * <p>
     * The description of the section, such as 'class room 1', etc.
     * The <code>@NotEmpty<code> Bean Validation constraint means
     * that the section description must contain at least 1 character.
     * </p>
     */
    @NotEmpty
    private String description;

    /**
     * <p>
     * The venue to which this section belongs. The <code>@ManyToOne<code>
     * JPA mapping establishes the relationship. The <code>@NotNull</code>
     * Bean Validation constraint means that the venue must be specified.
     * </p>
     */
    @ManyToOne
    @NotNull
    private Venue venue;
```

```
/**
 * The number of rows that make up the section.
 */
private int numberOfRows;

/**
 * The number of seats in a row.
 */
private int rowCapacity;

/* Boilerplate getters and setters */

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public int getNumberOfRows() {
    return numberOfRows;
}

public void setNumberOfRows(int numberOfRows) {
    this.numberOfRows = numberOfRows;
}

public int getRowCapacity() {
    return rowCapacity;
}

public void setRowCapacity(int rowCapacity) {
    this.rowCapacity = rowCapacity;
}

public int getCapacity() {
    return this.rowCapacity * this.numberOfRows;
}

public Venue getVenue() {
    return venue;
}

public void setVenue(Venue venue) {
    this.venue = venue;
}

/* toString(), equals() and hashCode() for Section, using the natural identity of the object */

@Override
public boolean equals(Object object) {
    if (!(object instanceof Section)) {
        return false;
    }
    Section castOther = (Section) object;
    return id != null ? id.equals(castOther.getId()) : false;
}

@Override
```

```
    public int hashCode() {
        return id != null ? id.hashCode() : System.identityHashCode(this);
    }

    @Override
    public String toString() {
        return name;
    }
}
```

3. Address:

```
package de.binaris.educationorganizer.model;

import java.io.Serializable;

import javax.persistence.Embeddable;
/**
 * <p>
 * A reusable representation of an address.
 *
 * Addresses are used in many places in an application,
 * so to observe the DRY principle, we model Address as an embeddable
 * entity. An embeddable entity appears as a child in the object model,
 * but no relationship is established in the RDBMS.
 * </p>
 */
@SuppressWarnings("serial")
@Embeddable
public class Address implements Serializable {

    /* Declaration of fields */
    private String street;
    private String city;
    private String country;

    /* Declaration of boilerplate getters and setters */

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    /* toString(), equals() and hashCode() for Address, using the natural identity of the object */

    @Override
    public boolean equals(Object o) {
        if (this == o)
            return true;
        if (o == null || getClass() != o.getClass())
            return false;

        Address address = (Address) o;

        if (city != null ? !city.equals(address.city) : address.city != null)
            return false;
        if (country != null ? !country.equals(address.country) : address.country != null)
            return false;
        if (street != null ? !street.equals(address.street) : address.street != null)
            return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = street != null ? street.hashCode() : 0;
        result = 31 * result + (city != null ? city.hashCode() : 0);
        result = 31 * result + (country != null ? country.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return street + ", " + city + ", " + country;
    }
}
```

4.MediaItem:

```
package de.binaris.educationorganizer.model;

import static javax.persistence.EnumType.STRING;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;

import org.hibernate.validator.constraints.URL;

/**
 * <p>
 * A reference to a media object such as images, audio recordings, texts,
 * that can be used in the application.
 * A media item contains the type of the media, which is required to render it
 * accordingly, as well as the URL at which the media is available.
 * </p>
 */
@Entity
public class MediaItem implements Serializable {

    private static final long serialVersionUID = 2323239584722232506L;
    /**
     * The ID of MediaItem.
     */
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    /**
     * <p>
     * The type of the media, required to render the media item correctly.
     *
     * The media type is a <em>closed set</em> - as each different
     * type of media requires support coded into the view layers,
     * it cannot be expanded upon without rebuilding the application.
     * It is therefore represented by an enumeration. We instruct
     * JPA to store the enum value using it's String representation,
     * so that we can later reorder the enum members,
     * without changing the data. This does mean we cannot change
     * the names of media items once the app is put into production.
     * </p>
     */
    @Enumerated(STRING)
    private MediaType mediaType;

    /**
     * <p>
     * The URL from which the media item can be sourced
     *
     * The url of the media item forms it's natural id
     * and cannot be shared between event categories
     *
     * The <code>@URL<code> Bean Validation constraint ensures
     * the URL is a valid URL.
     * </p>
     */
    @Column(unique = true)
    @URL
    private String url;

    /* Boilerplate getters and setters */

    public Long getId() {
        return id;
    }

    public MediaType getMediaType() {
        return mediaType;
    }

    public void setMediaType(MediaType mediaType) {
        this.mediaType = mediaType;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    /* toString(), equals() and hashCode() for MediaItem, using the natural identity of the object */

    @Override
    public String toString() {
        return "[" + mediaType.getDescription() + "] " + url;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof MediaItem)) {
            return false;
        }
        MediaItem castOther = (MediaItem) object;
        return id != null ? id.equals(castOther.getId()) : false;
    }

    @Override
    public int hashCode() {
        return id != null ? id.hashCode() : System.identityHashCode(this);
    }
}
```

5. MediaType:

```
package de.binaris.educationorganizer.model;

/**
 * <p>
 * The {@link MediaType} describes the types of media this application
 * can handle and render.
 *
 * The media type is a <em>closed set</em> - as each different type of
 * media requires support coded into the view layers,
 * it cannot be expanded upon without rebuilding the application.
 * It is therefore represented by an enumeration. When used, you
 * should instruct JPA to store the enum value using it's String representation,
 * so that we can later reorder the enum members,
 * without changing the data. This does mean we cannot change
 * the names of media items once the app is put into production.
 * To do this add <code>@Enumerated(STRING)</code> to the field declaration.
 *
 * The {@link MediaType} describes whether or not the
 * type of media can be cached locally and used offline.
 * </p>
 */
public enum MediaType {

    /**
     * The types of media the application can currently handle.
     */
    IMAGE("Image", true),
    TEXT("Text", true),
    PODCAST("Podcast", true),
    AUDIO("Audio", true),
    STREAM("Stream", true);

    /**
     * A human readable description of the media type.
     */
    private final String description;

    /**
     * A boolean flag indicating whether the media type can be cached.
     */
    private final boolean cacheable;

    /* Boilerplate constructor and getters */

    private MediaType(String description, boolean cacheable) {
        this.description = description;
        this.cacheable = cacheable;
    }

    public String getDescription() {
        return description;
    }

    public boolean isCacheable() {
        return cacheable;
    }
}
```

6. Event:

```
package de.binaris.educationorganizer.model;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.SequenceGenerator;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

/**
 * <p>
 * Represents an event, which may happen at different venues.
 *
 * Event's principal members are it's relationship to
 * {@link EventCategory} - specifying the type of event it is and
 * {@link MediaItem} - providing the ability to add media (such as a picture)
 * to the event for display.
 * It also contains additional info about the event,
 * such as it's name and a description.
 * </p>
 */
@Entity
public class Event implements Serializable {

     private static final long serialVersionUID = 2534178488726198123L;

    /**
     * The ID of Event.
     */
     @Id
     @GeneratedValue(strategy = GenerationType.AUTO)
     private Long id;

    /**
     * <p>
     * The name of the event.
     *
     * The name of the event forms it's natural identity and
     * cannot be shared between events.
     * Two constraints are applied using Bean Validation
     *
     * <ol>
     * <li><code>@NotNull</code> &mdash; the name must not be null.</li>
     * <li><code>@Size</code> &mdash; the name must be at least 5 characters
     * and no more than 250 characters, which is a simple length constraint.</li>
     * </ol>
     */
    @Column(unique = true)
    @NotNull
    @Size(min = 5, max = 250, message = "An event's name must contain between 5 and 250 characters")
    private String name;

    /**
     * <p>
     * A description of the event.
     *
     * Two constraints are applied using Bean Validation
     * <ol>
     * <li><code>@NotNull</code> &mdash; the description must not be null.</li>
     * <li><code>@Size</code> &mdash; the name must be at least 20 characters
     * and no more than 1000 characters, which is a simple length constraint,
     * and requires some description - a simple example of a business constraint.</li>
     * </ol>
     */
    @NotNull
    @Size(min = 20, max = 500,
            message = "An event's description must contain between 20 and 500 characters")
    private String description;

    /**
     * <p>
     * A media item, such as an image,
     * which can be used to entice a browser to book a ticket.
     *
     * Media items can be shared between events, so this
     * is modeled as a <code>@ManyToOne</code> relationship.
     * Adding a media item is optional, and the view layer
     * will adapt by not showing any item link, if none is provided.
     * </p>
     */
    @ManyToOne
    private MediaItem mediaItem;

    /**
     * <p>
     * The category of the event
     *
     * Event categories are used to ease searching of available of events,
     * and hence this is modeled as a <code>@ManyToOne</code> relationship.
     * The Bean Validation constraint <code>@NotNull</code>
     * indicates that the event category must be specified.
     * </p>
     */
    @ManyToOne
    @NotNull
    private EventCategory category;

    /* Boilerplate getters and setters */

    public Long getId() {
       return id;
    }

    public void setId(Long id) {
       this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public MediaItem getMediaItem() {
        return mediaItem;
    }

    public void setMediaItem(MediaItem picture) {
        this.mediaItem = picture;
    }

    public EventCategory getCategory() {
        return category;
    }

    public void setCategory(EventCategory category) {
        this.category = category;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    /* toString(), equals() and hashCode() for Event, using the natural identity of the object */

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Event)) {
            return false;
        }
        Event castOther = (Event) object;
        return id != null ? id.equals(castOther.getId()) : false;
    }

    @Override
    public int hashCode() {
        return id != null ? id.hashCode() : System.identityHashCode(this);
    }

    @Override
    public String toString() {
        return name;
    }
}
```

7. EventCategory:

```
package de.binaris.educationorganizer.model;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;

import org.hibernate.validator.constraints.NotEmpty;

/**
 * <p>
 * The Category of an event.
 *
 * {@link EventCategory} is a simple entity,
 * used to easier filtering of information by users.
 * </p>
 */
@Entity
```

```
public class EventCategory implements Serializable {

     private static final long serialVersionUID = 5942034636094201600L;

    /**
     * The ID of EventCategory.
     */
     @Id
     @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    /**
     * <p>
     * A description of the event category.
     *
     * The description of an event category forms it's natural id
     * and cannot be shared between event categories.
     * The <code>@NotEmpty<code> Bean Validation constraint means
     * that the event category descripton must contain least 1 character
     * and cannot be null.
     * </p>
     */
    @Column(unique=true)
    @NotEmpty
    private String description;

    /* Getters and setters */

    public Long getId() {
        return id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    /* toString(), equals() and hashCode() for EventCategory, using the natural identity of the object */

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof EventCategory)) {
            return false;
        }
        EventCategory castOther = (EventCategory) object;
        return id != null ? id.equals(castOther.getId()) : false;
    }

    @Override
    public int hashCode() {
        return id != null ? id.hashCode() : System.identityHashCode(this);
    }

    @Override
    public String toString() {
        return description;
    }
}
```

Für die Verwendung von JPA 2.1 generiert Maven mittels der Dependency **hibernate-jpamodelgen** in der pom.xml bei Aufruf von **mvn install** im Verzeichnis

target/generated-sources/annotations/de/binaris/educationorganizer/model

die JPA 2 Meta-Modell Klassen, z. B. die abstrakte Klasse **EventCategory_.java**, wenn JPA 2 verwendet werden soll, dem **Deployment-Paket educationorganizer.war** zusammen mit den anderen hinzuzufügen ist:

```
package de.binaris.educationorganizer.model;

import javax.annotation.Generated;
import javax.persistence.metamodel.SingularAttribute;
import javax.persistence.metamodel.StaticMetamodel;

@Generated(value = "org.hibernate.jpamodelgen.JPAMetaModelEntityProcessor")
```

```
@StaticMetamodel(EventCategory.class)
public abstract class EventCategory_ {

        public static volatile SingularAttribute<EventCategory, Long> id;
        public static volatile SingularAttribute<EventCategory, String> description;

}
```

e) Die Datenbank:

Die persistence.xml aus dem /META-INF Unterverzeichnis zur Aktivierung von JPA 2 sieht für die Verwendung der HSQL Beispiel-Datenbank folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
   xmlns="http://java.sun.com/xml/ns/persistence"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
       http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
   <persistence-unit name="educationorganizer">
     <jta-data-source>java:jboss/datasources/EducationorganizerDatasource</jta-data-source>
     <properties>
       <property name="hibernate.hbm2ddl.auto" value="create-drop" />
       <property name="hibernate.show_sql" value="false" />
     </properties>
   </persistence-unit>
</persistence>
```

Die Verwendung von JPA 2 ist am folgenden <persistence …>-Element sehr gut zu erkennen. Denn ohne dass hier die version="2.0" eingetragen ist, fährt man gezwungenermaßen stets nur persistence 1.0, d.h. z.B. JPA 1.

```
<persistence version="2.0"
   xmlns="http://java.sun.com/xml/ns/persistence"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
       http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
```

Für eine produktive Applikation und Datenbank sollte die Datasource in der standalone.xml entsprechend konfiguriert werden, wie es z. B. in diesem Blog-Eintrag mit dem Titel "" hier für den JBoss 7.1 (Brontes) beschrieben wurde. Für eine Beispiel-Applikation soll die Verwendung der HSQL "in memory"-Datenbank genügen. Hierfür wird im /WEB-INF-Verzeichnis neben der beans.xml (CDI-Aktivierung) auch die folgende **educationorganizer-ds.xml** hinterlegt:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- unmanaged datasource for testing purposes only uses H2,
       an in memory database that ships with JBoss AS. -->
<datasources xmlns="http://www.jboss.org/ironjacamar/schema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.jboss.org/ironjacamar/schema
    http://docs.jboss.org/ironjacamar/schema/datasources_1_0.xsd">

    <!-- The datasource is bound into JNDI at this location. We reference
       this in META-INF/persistence.xml -->
    <datasource jndi-name="java:jboss/datasources/EducationorganizerDS"
       pool-name="grusskarten" enabled="true"
       use-java-context="true">
       <connection-url>
             jdbc:h2:mem:educationorganizer;DB_CLOSE_ON_EXIT=FALSE;DB_CLOSE_DELAY=-1
       </connection-url>
       <driver>h2</driver>
       <security>
          <user-name>admin</user-name>
          <password>banana_joe</password>
       </security>
    </datasource>
</datasources>
```

Bekanntlich sind die neuen MySQL-Datenbanken seit der Version 5.1 aber auch in der Lage sowohl Trigger und seit der Existenz von JDBC 4-fähigen Datenbank-Treibern (z. B. der Version 34) auch in der Lage Transaktionen zu verarbeiten. Auch Sequenzen kann eine aktuelle MySQL-Datenbank für jede Tabelle autoinkrementell ohne großen Aufwand verwenden.

Hierfür wird einfach zusätzlich je Entity-Tabelle (**"Table per Class" Design Pattern**) eine Sequenz-Tabelle angelegt, die an die Definition der 'hibernate_sequence'-Tabelle angelehnt ist. Diese 'hibernate_sequence'-Tabelle wird bei der Verwendung einer entsprechender **@Id** Annotation mit **@GeneratedValue(strategy = GenerationType.*AUTO*)** auf einer Enitity Klase zunächst beim Deployment der Beispiel-Applikation **'educationorganizer'** für alle erzeugten Entity-Tabellen gemeinsam angelegt, wenn in der persistence.xml die Property **<property name="hibernate.hbm2ddl.auto" value="create-drop" />** gesetzt wurde und natürlich der Persistence Provider in der persistence.xml hinzugefügt wurde, also ganz am Anfang die folgende Zeile: **<provider>org.hibernate.ejb.HibernatePersistence</provider>**.

Durch die Konfiguration von **create-drop** werden in der zuvor manuell angelegten Datenbank (**CREATE DATABASE IF NOT EXISTS `educationorganizer`**) beim Deployment der Beispiel-Applikation auch die Foreign Keys für die relational verbundenen Tabellen angelegt.

Mit nur einer **'hibernate_sequence'-Tabelle für alle Entities** würde die Id eines Datensatzes beim Hinzufügen eines neuen Datensatzes allerdings für alle Tabellen gleichzeitig inkrementiert, was zu dem Verhalten führt, dass zwischen den Einträgen zweier Datensätze derselben Tabelle ein unerwünschter Offset entsteht, weil inzwischen in einer anderen Tabelle ebenfalls ein Datensatz hinzugefügt wurde.

Deshalb wird die 'hibernate_sequence'-Tabelle wieder gelöscht und stattdessen aus der angelegten Datenbank 'educationorganizer'

1. die folgende import.sql mittels HeidiSQL erstellt (**mit je einer Sequenz-Tabelle je Entity**):

Die import.sql ist hier zum Download verfügbar.

2. Vor einem erneuten Build und Deployment des **educationorganizer.war** wird in der persistence.xml die Property **<property name="hibernate.hbm2ddl.auto" value="none" />** zurückgesetzt und die @**Id** Annotation mit der @**GeneratedValue** Annotation auf jeder davon betroffenen Entity Klasse um die entsprechende Sequenz-Definition der angelegten Sequenz-Tabellen erweitert:

Venue:

```
/**
 * The ID of Venue.
 */
@Id
@GeneratedValue(
        strategy = GenerationType.SEQUENCE,
        generator="my_entity_seq_gen_venue")
@SequenceGenerator(
        name = "my_entity_seq_gen_venue",
        sequenceName="sequence_venue",
        allocationSize=1)
private Long id;
```

Section:

```
/**
 * The ID of Section.
 */
@Id
@GeneratedValue(
        strategy = GenerationType.SEQUENCE,
        generator="my_entity_seq_gen_section")
@SequenceGenerator(
        name = "my_entity_seq_gen_section",
        sequenceName="sequence_section",
        allocationSize=1)
private Long id;
```

MediaItem:

```
/**
 * The ID of MediaItem.
 */
@Id
@GeneratedValue(
        strategy = GenerationType.SEQUENCE,
        generator="my_entity_seq_gen_media_item")
@SequenceGenerator(
        name = "my_entity_seq_gen_media_item",
        sequenceName="sequence_media_item",
        allocationSize=1)
private Long id;
```

Event:

```
/**
 * The ID of Event.
 */
@Id
@GeneratedValue(
        strategy = GenerationType.SEQUENCE,
        generator="my_entity_seq_gen_event")
@SequenceGenerator(
        name = "my_entity_seq_gen_event",
        sequenceName="sequence_event",
        allocationSize=1)
private Long id;
```

Event:

```
/**
 * The ID of EventCategory.
 */
@Id
@GeneratedValue(
        strategy = GenerationType.SEQUENCE,
        generator="my_entity_seq_gen_event_category")
@SequenceGenerator(
        name = "my_entity_seq_gen_event_category",
        sequenceName="sequence_event_category",
        allocationSize=1)
private Long id;
```

3. Der JBoss WildFly Application Server wird konfiguriert:

Diese Minimal-Konfiguration betrifft:

- die MySQL Datenbank/den Java MySQL-Datenbank-Treiber und die MySQL-Datasource der 'educationorganzier' Beispiel-Applikation
- die Hibernate Version und ihre interessierenden Features/Dependencies
- das Start-Skript des Application Servers WildFly 8.2 für den Start ohne zusätzliche Umgebungsvariablen

Eine fertig Version mit Beispiel-Konfiguration der Beispiel-Applikation 'educationorganizer' kann hier heruntergeladen, in ein beliebiges Verzeichnis ausgepackt und durch Starten von **standalone.bat (Windows) oder standalone.sh (Linux)** sofort gestartet werden. Anschließend ist die Beispiel-Applikation aufrufbar unter der folgenden Url:

http://localhost:8080/educationorganizer

Die Test-Frameworks und die Tests

a) Arquillian Tests für das Service-Backend
wurden in diesem folgendem Blog-Eintrag hier bereits erklärt.

b) QUnit-Tests für die REST-Services
wurden in diesem folgendem Blog-Eintrag hier bereits erklärt.

c) Selenium Tests für die Benutzerschnittstelle
wurden in diesem folgendem Blog-Eintrag hier bereits erklärt.

Bei dieser Gelegenheit darf auch auf das sehr effektive, interessante und erfolgreiche Seminar

"**TDD mit Java**" **hier** und **hier** von Binaris Informatik hingewiesen werden.

Wie Test-Suites, die mit dem Selenium IDE PlugIn durchgeführt wurden und automatisiert ausgeführt werden können, erstellt werden, wurde bereits in diesem Blog Eintrag hier beschrieben.

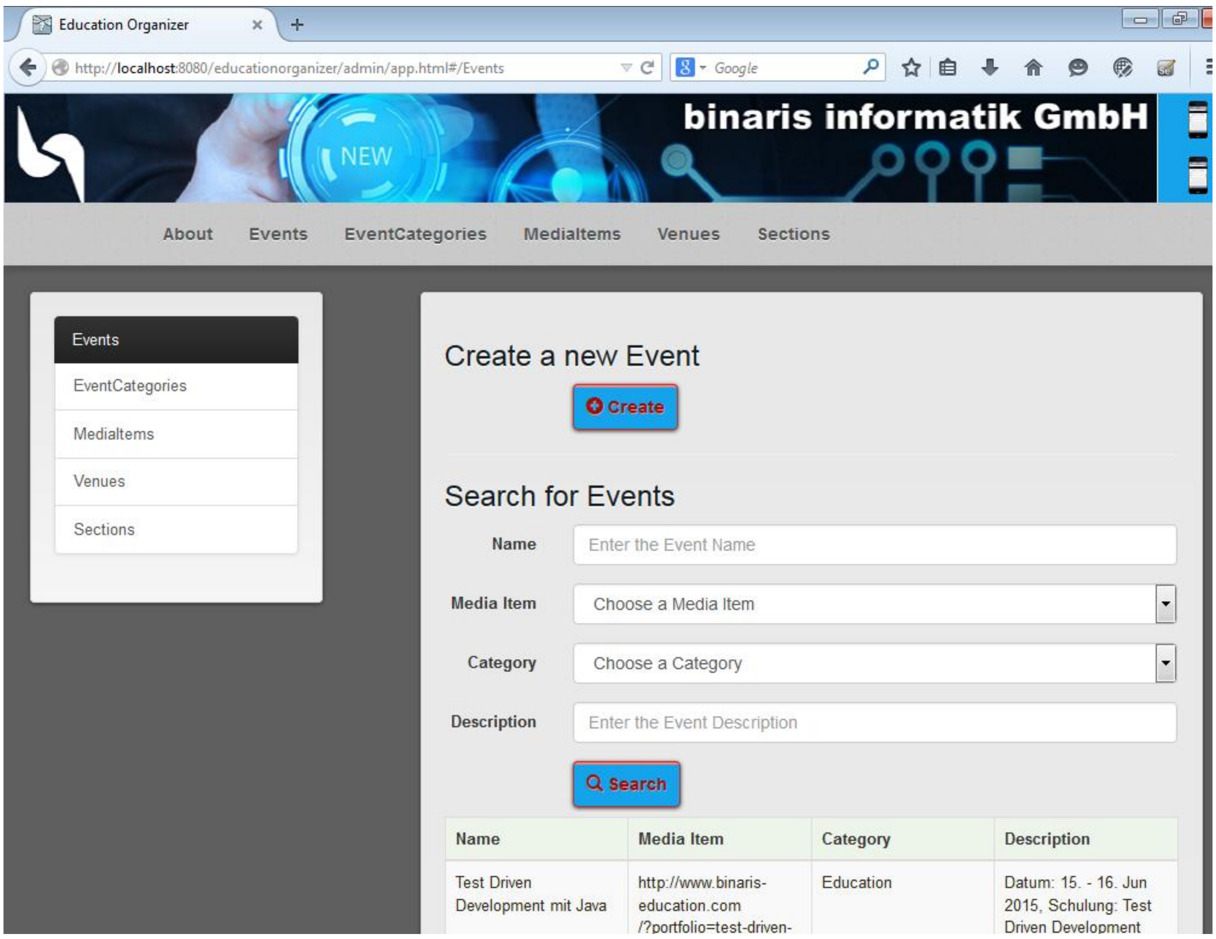
Hier nun auch das komplette Web-Archiv **educationorganizer.war** zum Download:
http://www.4shared.com/file/YxqB4JjRba/educationorganizer.html

Hier den **WildFly 8.2** als Zip-Archiv zum Download und Entpacken:
http://download.jboss.org/wildfly/8.2.0.Final/wildfly-8.2.0.Final.zip

Installation/Start der Beispiel-Applikation:
- Das JBoss Zip-Archiv entpacken
- JAVA_HOME, JBOSS_HOME entsprechend setzen und
  in der standalone.bat oder standalone.sh verwenden oder den vorkonfigurierten Server **hier** auspacken.
- educationorganizer.war ins Server-Unterverzeichnis /standalone/deployments speichern
- Neben educationorganizer.war eine leere Textdatei namens educationorganizer.war.dodeploy speichen.
- Server starten per standalone.bat oder standalone.sh, http://localhost:8080/educationorganizer/ aufrufen
- Die Selenium Tests erstellen per Selenium IDE PlugIn/Recorder erstellen und ausführen.

Hier noch ein weiteres interessantes Lernvideo von JBoss:
http://www.jboss.org/video/vimeo/92554627/

Nachher:



Create a new Event

Search for Events

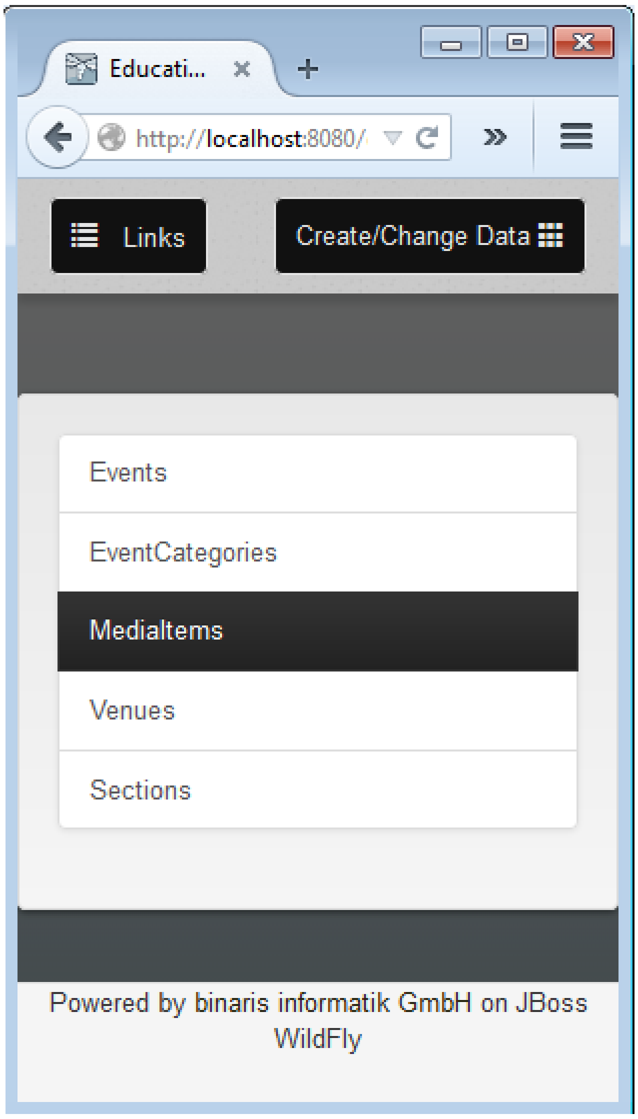
Name: Enter the Event Name

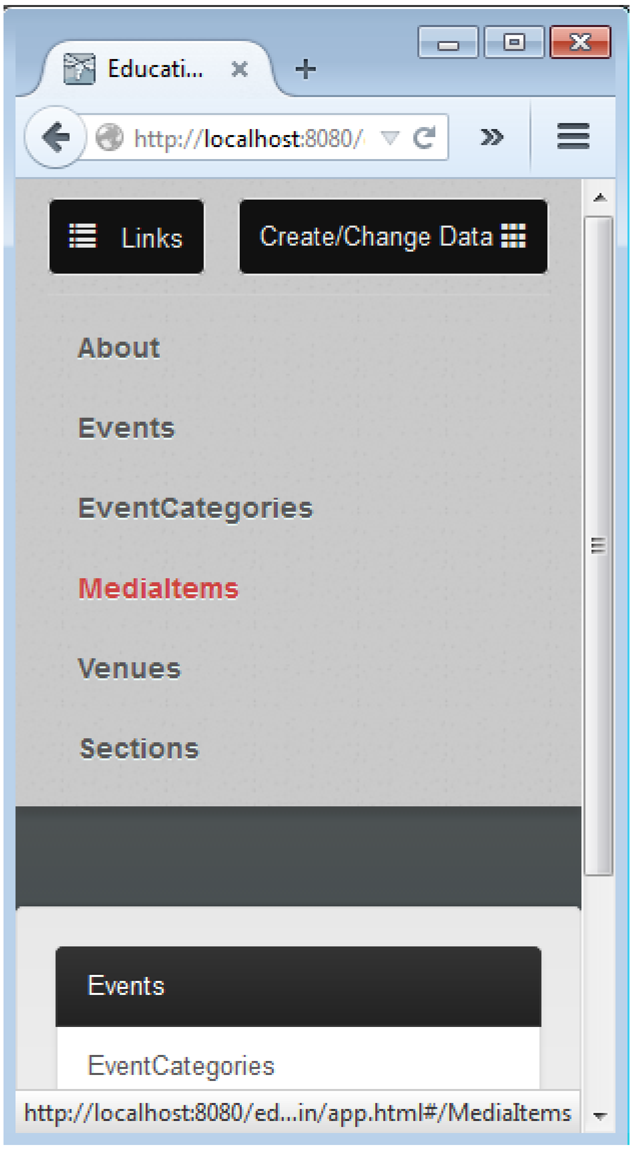
Media Item: Choose a Media Item

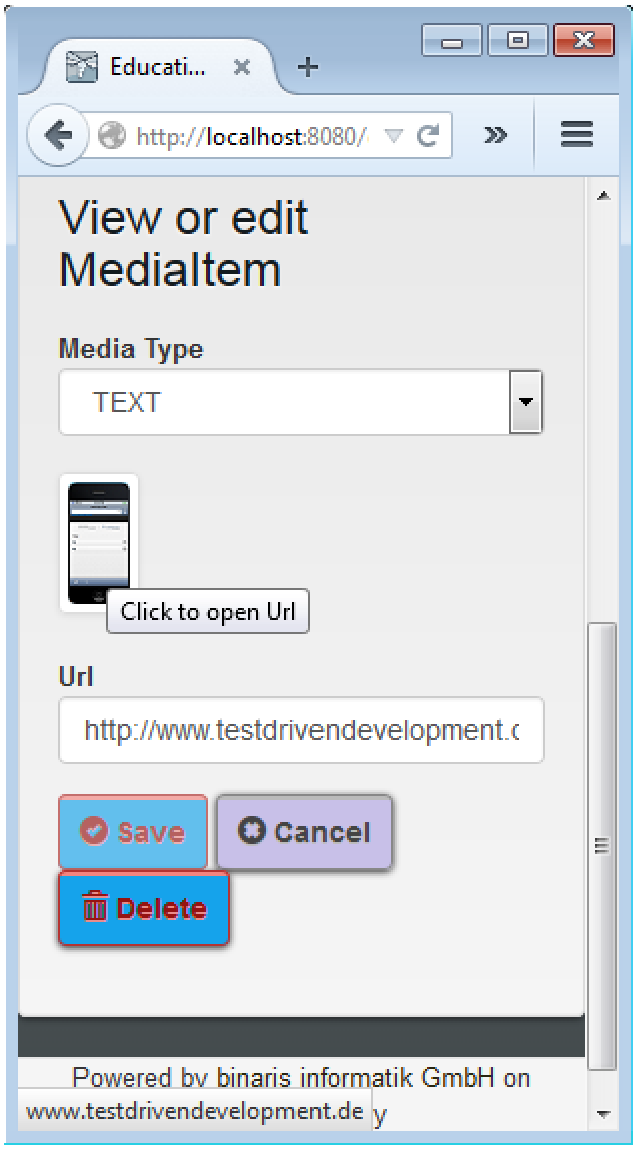
Category: Choose a Category

Description: Enter the Event Description

Name	Media Item	Category	Description
Test Driven Development mit Java	http://www.binaris-education.com /?portfolio=test-driven-	Education	Datum: 15. - 16. Jun 2015, Schulung: Test Driven Development


Educati...
http://localhost:8080/
Links
Create/Change Data
Events
EventCategories
MediaItems
Venues
Sections
Powered by binaris informatik GmbH on JBoss WildFly


Educati...
http://localhost:8080/
Links
Create/Change Data
About
Events
EventCategories
MediaItems
Venues
Sections
Events
EventCategories
http://localhost:8080/ed...in/app.html#/MediaItems



Educati...

http://localhost:8080/

View or edit MediaItem

Media Type

TEXT

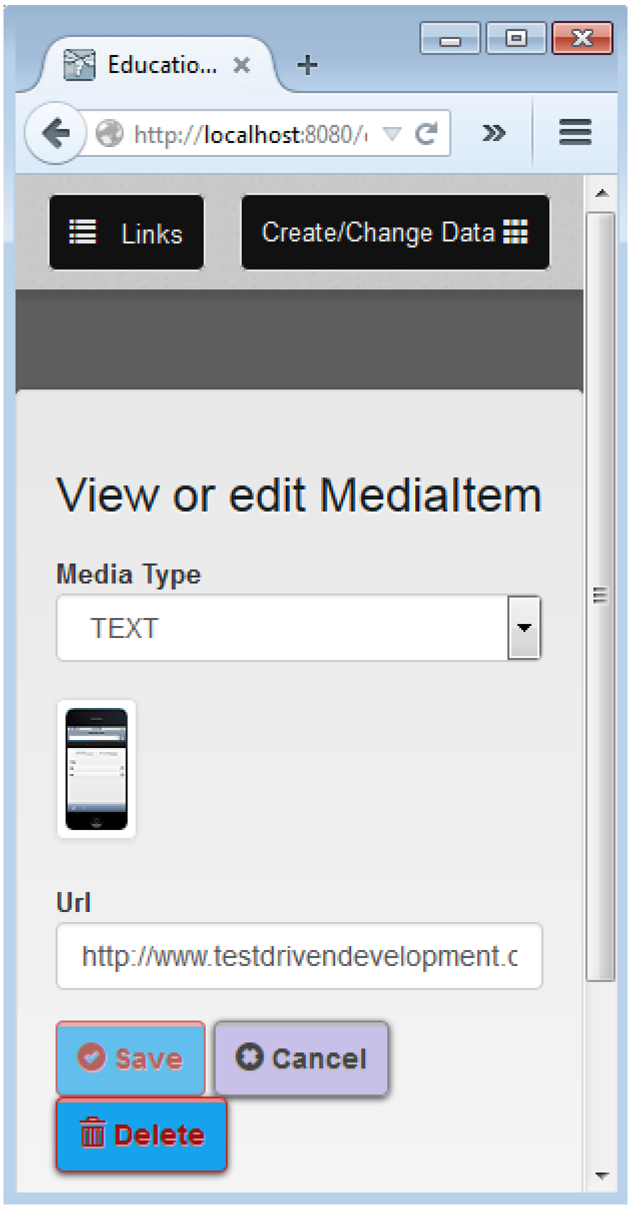
Click to open Url

Url

http://www.testdrivendevelopment.c

Powered by binaris informatik GmbH on

www.testdrivendevelopment.de y



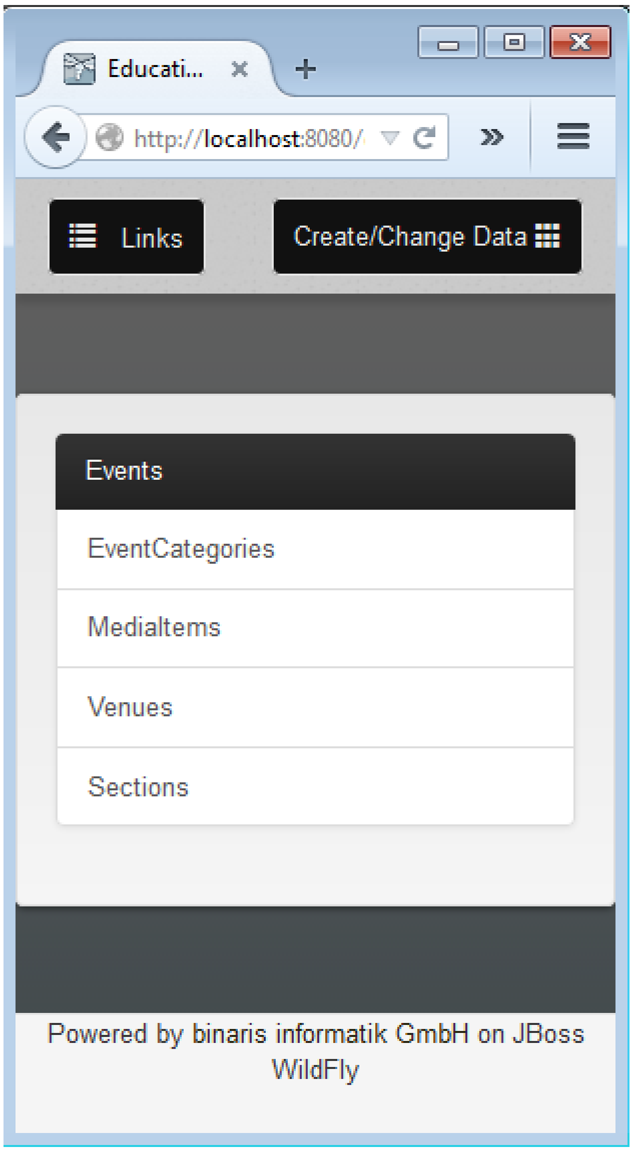
View or edit MediaItem

Media Type

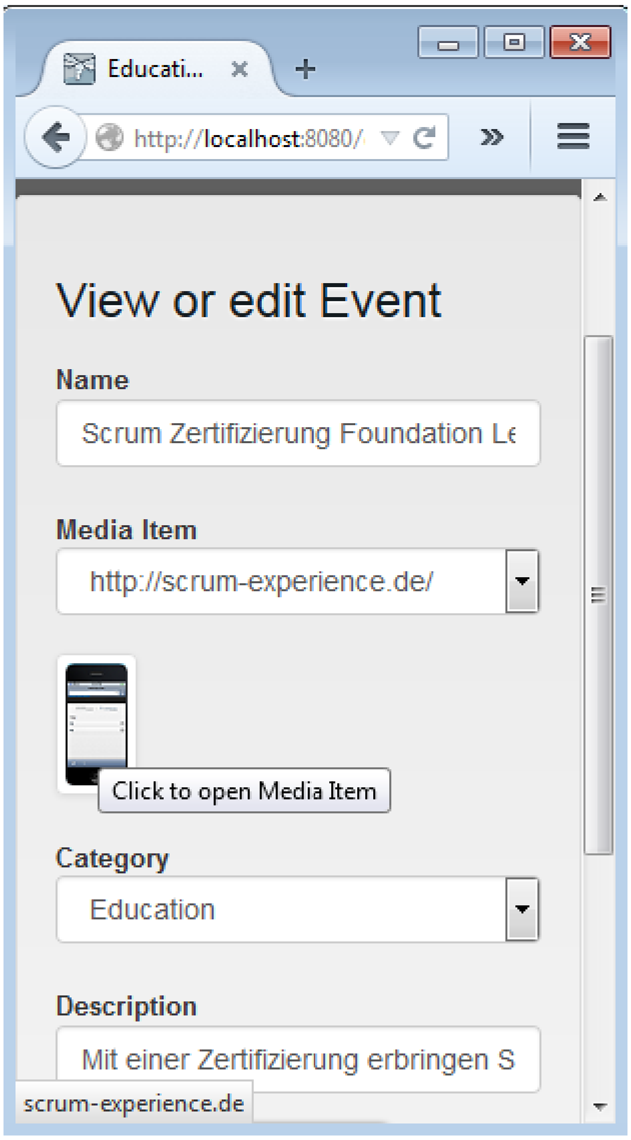
TEXT

Url

http://www.testdrivendevelopment.c



Powered by binaris informatik GmbH on JBoss WildFly



View or edit Event

Name
Scrum Zertifizierung Foundation Le

Media Item
http://scrum-experience.de/

Click to open Media Item

Category
Education

Description
Mit einer Zertifizierung erbringen S

scrum-experience.de