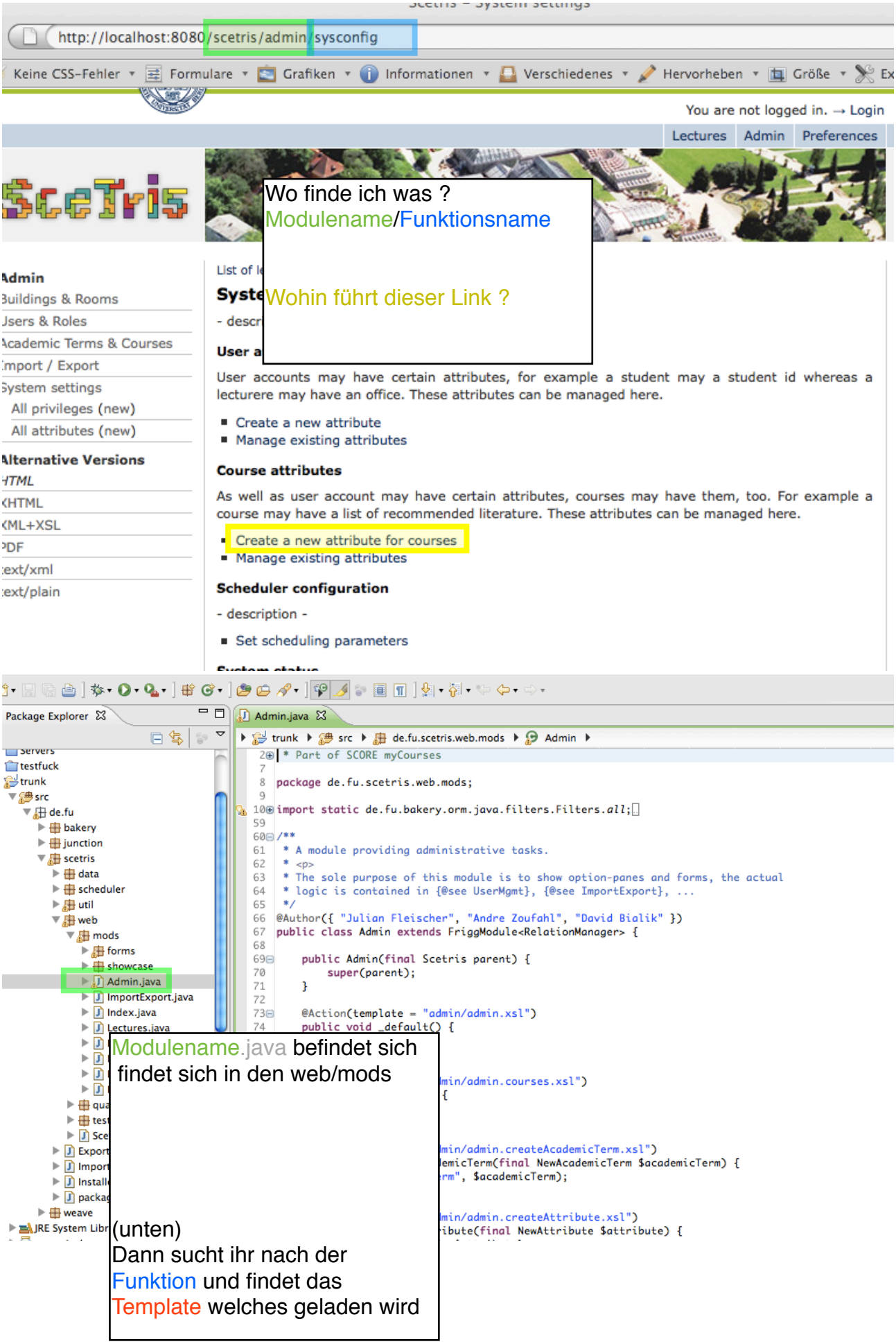
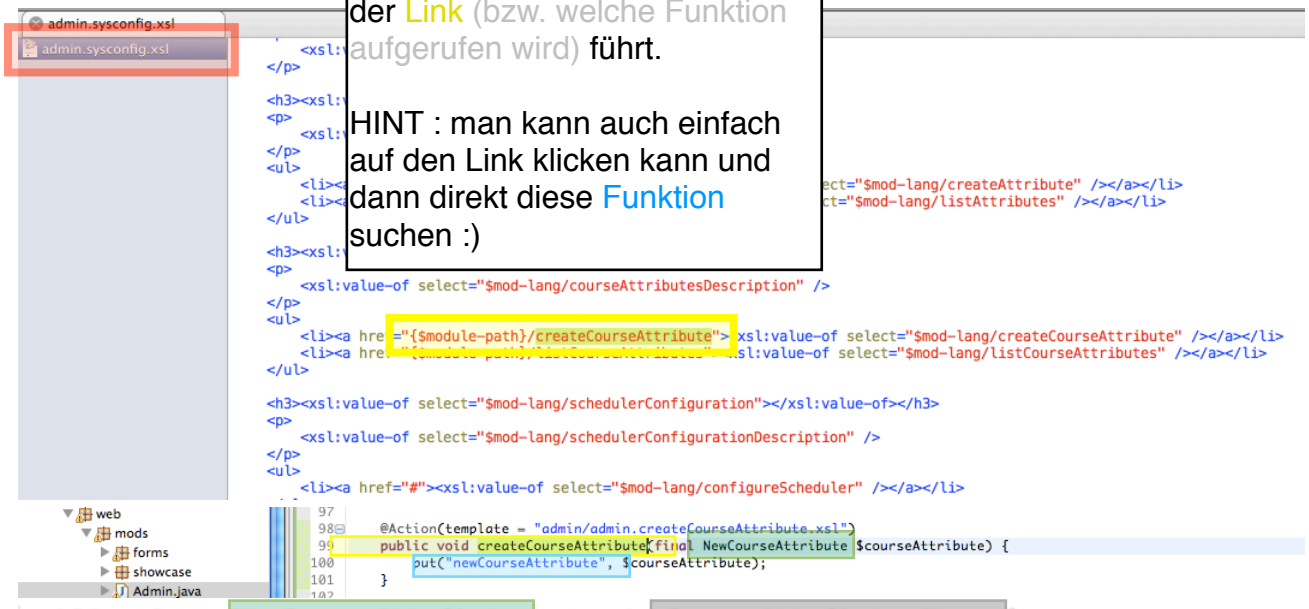

http://localhost:8080/scetris/admin/sysconfig
You are not logged in. → Login
Lectures
Admin
Preferences
Wo finde ich was ?
Modulename/Funktionsname
Wohin führt dieser Link ?
Admin
Buildings & Rooms
Users & Roles
Academic Terms & Courses
Import / Export
System settings
All privileges (new)
All attributes (new)
Alternative Versions
HTML
XHTML
XML+XSL
PDF
text/xml
text/plain
User a
User accounts may have certain attributes, for example a student may a student id whereas a lecturere may have an office. These attributes can be managed here.
Create a new attribute
Manage existing attributes
Course attributes
As well as user account may have certain attributes, courses may have them, too. For example a course may have a list of recommended literature. These attributes can be managed here.
Create a new attribute for courses
Manage existing attributes
Scheduler configuration
- description -
Set scheduling parameters
Package Explorer
Admin.java
trunk
src
de.fu.scetris.web.mods
Admin
testfuck
de.fu
bakery
junction
scetris
data
scheduler
util
web
mods
forms
showcase
Admin.java
ImportExport.java
Index.java
Lectures.java
weave
JRE System Libr
* Part of SCORE myCourses
package de.fu.scetris.web.mods;
import static de.fu.bakery.orm.java.filters.Filters.all;
/**
* A module providing administrative tasks.
* <p>
* The sole purpose of this module is to show option-panes and forms, the actual
* logic is contained in {@see UserMgmt}, {@see ImportExport}, ...
*/
@Author({ "Julian Fleischer", "Andre Zoufahl", "David Bialik" })
public class Admin extends FriggModule<RelationManager> {
public Admin(final Scetris parent) {
super(parent);
}
@Action(template = "admin/admin.xsl")
public void _default() {
Modulename.java befindet sich findet sich in den web/mods
(unten)
Dann sucht ihr nach der Funktion und findet das Template welches geladen wird

```
@Action(template = "admin/admin.sysconfig.xsl")
public void sysconfig() {

}
```

Im Template seht ihr dann wohin der Link (bzw. welche Funktion aufgerufen wird) führt.

HINT : man kann auch einfach auf den Link klicken kann und dann direkt diese Funktion suchen :)




```
public class NewCourseAttribute extends CourseAttribute.Form {

    /**
     *
     */
    private static final long serialVersionUID = -708132292

    public CourseAttribute $createdCA;


    @Override
    public boolean commit() throws Exception {
        $createdCA = manager().newCourseAttribute(name)
                .setUniqueValue(uniqueValue)
                .setType(type)
                .setRequired(required)
                .create();
        System.out.println($createdCA);
        return true;
```

Nun wird es spannend:
Die Funktion kriegt ein Objekt und puttet dieses als xml-node

Wohin müsst ihr nun ?
In ../forms/ legt ihr eine entspr. Klasse an (wenn nicht schon vorhanden)
Dieses erbt bei nun von einer Entität die entspr. Formdaten

<-- der Code ist erstmal irrelevant
(tip: er wird ausgeführt, wenn das Form abgeschickt)

```
*/
@de.fu.bakery.orm.java.annotation.Entity(name = "CourseAttribute")
@de.fu.bakery.orm.java.annotation.Relation(name = "\"scetris\"."\"CourseAttribute\"", requiredSqlCols = {"name"})
@de.fu.weave.xml.annotation.XmlElement("CourseAttribute")
public class CourseAttribute extends de.fu.bakery.orm.java.Gene
    final RelationManager manager;
    boolean exists = false;

    public static final String Timekey = "timekey";
    public static final String Id = "id";
    public static final String Name = "name";
    public static final String Type = "type";
    public static final String UniqueValue = "unique_value";
    public static final String Required = "required";


    /**
     *
     */
    public abstract static class Form extends de.fu.scetris.dat
        public static final long serialVersionUID = 1415640L;

        public java.sql.Timestamp timekey;

        @de.fu.weave.Form.Pos(2)
        @de.fu.weave.Form.Field("id") @de.fu.weave.Form.Require
        @de.fu.weave.Form.FormControl(de.fu.weave.Form.Control.
        @de.fu.weave.Form.ActiveValidator("id$validator")
        @de.fu.weave.Form.ActiveConverter("id$converter")
        public int id;
```

In ../../data findet ihr die Entität und die entspr. Formdaten

Was hier steht, sind alles Felder die im Form dann vorkommen.
Z.B. ist die Id an Position 2. Versteckt und Required.
An sich ist das egal, da die ja autogeneriert sind und hier keine was ändern muss.

Was ihr dann noch machen müsst, ist im Template das was ihr puttet auch mittels Formbuilder sichtbar machen.

admin.createCourseAttribute.xsl
admin.createDepartment.xsl
admin.createUser.xsl

```
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns="http://www.w3.org/1999/xhtml"
xmlns:vars="http://technodrom.scravy.de/2010/data" xmlns:item="http://technodrom.scravy.de/2010/item">

<xsl:import href="common.xsl" />
<xsl:import href="../lego/form-builder.xsl" />

<xsl:template mode="content" match="vars:meta">
    <xsl:call-template name="form-builder">
        <xsl:with-param name="form">newCourseAttribute</xsl:with-param>
    </xsl:call-template>
</xsl:template>

<xsl:template mode="path" match="vars:meta">

</xsl:template>

</xsl:transform>
```

List of lectures

Name:

Type:

Unique Value:

Required:

Create new Courseattribute

Voila, Form fertig.
Wenn ihr das Form nun abschickt, wird der commit() code der Klasse ausgeführt (Hier: das CourseAttribute wird erzeugt)

Nun noch in I18n die Sprachdatei erweitern um entspr. Labels !
(Labels die nicht belegt sind, werden einfach so ausgegeben)

admin.createCourseAttribute.xsl
admin.createDepartment.xsl
admin.createUser.xsl
admin.en.xml

```
<!-- ENTITY : COURSEATTRIBUTE -->
<newCourseAttribute.name>Name</newCourseAttribute.name>
<newCourseAttribute.type>Type</newCourseAttribute.type>
<newCourseAttribute.uniqueValue>Unique Value</newCourseAttribute.uniqueValue>
<newCourseAttribute.required>Required</newCourseAttribute.required>
<newCourseAttribute.button.submit>Create new Courseattribute</newCourseAttribute.button.submit>
```