Howto: Create Forms

We are using two kinds of forms in scetris: Handcrafted ones and semi-handcraftes ones. This article will try to illustrate the both of them.

Generell principles

Forms are objects and as such they are derived from a class. It is possible to extend forms, create constructors, etc. Note however, that a public constructor with no arguments must exist (i.e. none, if you do not specify any). Most issues, like converting a String value a user has entered, into a java-object are handled automatically. If, however, you want some special behaviour, you may specify your own **Converter**. Additionally there are **Validators**. Validators check for a single value to be valid, i.e. a Converter checks the syntax and converts it into a java-object whereas a Validator checks additional semantics on a given field, such as no duplicate values.

Converters are taken from the junction-lib, whereas Validators are specific to weave.

Technical details

Forms do implement [score:/trunk/src/de/fu/weave/Form.java de.fu.weave.Form]. However, since the interface describes many methods which are likely to be the same across all forms, there exist AbstractForm implementations in the GenericModule-classes such as FriggModule. They cope with stuff like passing the RelationManager around, etc. So be sure to always implement AbstractForm, which should be already visible in the Module-Implementation you're extending.

The interfaces relevant for Forms are the following:

- de.fu.weave.Form
- de.fu.weave.Validator
- de.fu.junction.converter.StringConverter

Your first Form

In order to create a Form you need to have Module, we recommend reading HowtoModuleDevelopment. If you've got one, read on.

You declare a Form simply by implementing the Form interface or extending an abstract base class. We recommend the latter one. Each public member of such a Form may be annotated via @Field (which does not need to be imported, since it's declared within Form). So your first Form might look like this:

```
/*declaration of module class */ {
  public static class MyFirstForm extends AbstractForm {
    @Field("name")
    public String userName = "anonymous";
  }
}
```

To use that form you will have to put() it into the output-document within an action:

```
  @Action
  public void anyAction(MyFirstForm $form) {
    put("myFirstForm", $form);
  }
```

That was the part on the Java-side of life. Note, that the $form is being passed in as a parameter, just like @Arg-parameters are. You module still lacks information on how to display that stuff. Fortunately there exists a lego-template which automatically builds forms for us. So the accompanying part in your stylesheet may look like this:

```
  <xsl:call-template name="form-buider">
    <xsl:with-param name="form">myFirstForm</xsl:with-param>
  </xsl:with-param>
```