Julians Developer Page

2010-12-11 22:41

I just refactored the FriggController to offer beforeAction, beforeInitialization and likewise after*-Methods which *may* be overriden by a Weavlet (such as Scetris). Changes were made in r1467, have a look at trunk/src/de/fu/scetris/web/Scetris.java@1467 to get a quick impression.

The reason I implemented it was a request from André for being able of doing something always before an action is executed. before- and afterAction will get the module-object as well as an action reflector for the action to be invoked. You can now do things like that:

```
class Scetris ... {
  protected void beforeAction(Module<RelationManager> module, ActionReflector
    module.put("variable", "that is always available");
  }
}
```

2010-12-11 23:42

I've added methods to Session and Module which ease handling of sessions from within modules. Module has gotten a getSession()-method and Session has gotten some methods for getting/setting custom data. Sessions rely internally on HttpSession-Objects, as specified by the Java Servlet Specification (so I did not have to do too much).

The new methods for Session are getValue(String name), getValue(String name, Class type), getValue(String name, <Type> fallback), ... Changes were committed in r1468. It is now possible to write stuff like the following within a modules action:

```
Module ... {
  Action ... { // simple counter for a page
    getSession().setValue("numberOfVisits", getSession().getValue("numberOfVi
  }
}
```

2010-12-12 00:26

The declaration of Modules has been greatly simplified, as well as the declaration of actions, the final and stable version has just landed in r1469.

- @ModuleInfo is no longer needed, has been removed from weave and I deleted all occurences
- @Action does not need to specify a name any longer. If no name is specified, the name of the method is taken as fallback.
- Actions do no longer need to have a first parameter with no annotations of type String[]. However, if such a parameter *is* present if will be usable as before.

Thus the minimal complete declaration of a Module (within Scetris) now looks like this:

```
/**
 * A minimal module
 */
public class ShowCase extends FriggModule<RelationManager> {

        public ShowCase(final Scetris $parent) {
                super($parent);
        }

        @Action(template = "name-of-template.xsl")
        public void _default() {
                // ... stuff
        }
}
```

An action may be written like this:

```
public void edit() { }
```

2010-12-12 00:37

I just found out that you **do not need to rebuild** the project every time to newly deploy it in tomcat. **If suffices to do an ant reinstall**, since Eclipse automatically build the project and supports AspectJs incremental building options. Have fun testing! Saves me **20 to 30 seconds** each time I want to test something.

2010-12-12 01:44

I just release the API for Forms. I played around a long time considering what would be best, I've written several modules and discarded all of them. However, here is the final API (I'm currently implementing it, will be landing tomorrow, since it's full of bugs and I really don't want to put THAT into SVN).

The new API at a glance:

- @Commit is de facto deprecated, since we now have Forms which feature their own commit-Method
- Several interfaces, namely Module.Validator and Module.Schema have been added
- Dozens of new Annotation types (all members of the Module.Schema Interface) are there

To give you a first impression:

```
Module ... {
  class MyForm implements Schema {
    @Field @Min(1) @Max(10)
    int priority;

    @Field @Only("abcdef0123456789") @Min(1)
    String hex;

    @Field("building")
    @Converter(RetrieveBuilding.class)
    Building building;

    public void commit() throws DatabaseException {
      // ...
    }
  }

  @Action myComplexModule(MyForm form) {
    put("form1", form);
  }
}
```

I will discuss the API in detail tomorrow, when there is something to play with :-)

2010-12-12 19:01

I relaxed a bit from pimping the Form-Validation (showing good progress) and implemented a query-cache (r1474). Some databases do this natively (mysql for example), some jdbc-driver enable result-pooling (the jdbc driver for postgres does not). So I did this by hand.

However, the queryCache does not have a timelimit and since a servlet is persistent it should not be used from within a weavlet (this feature may come in the future). It is mainly there to accelerate the scheduler. To use it, use RelationManager.enableQueryCache(). To disable it later on use RelationManager.disableQueryCache(). To clear it in between use RelationManager.enableQueryCache(true) (this leaves it enabled, but clears the cache).

How does it work? Well, in GenericRelationManager is a method executeQuery which is used throughout the ORM instead of PreparedStatement.executeQuery. If the query cache is enabled it will lookup the result from the cache instead of issuing a query to the database. However, the cache won't get updated, since it relies on a binary comparison of the queries, i.e. it is not smart. That means that in contrast to a query cache like mysql offers natively you will get outdated results if updating something and than querying the database for it.

I think the scheduler writes data back only when it’s stopped or paused, so this won’t be an issue for us. However, it is important to disable the cache than!

Here is a small piece of code sketching the idea:

```
manager().enableQueryCache();
// do scheduling
manager().disableQueryCache();
// write back
```

In principle it shouldn’t matter whether you first disable the query cache and than write back or the other way around, since queries which result in UPDATE or CREATE statements are handled using executeUpdate() or execute(), rather than executeQuery(). However, I guess with the above version you are on the safe side.

The cache is write-through, which should already be clear, since it only caches reads from the database.

2010-12-13 08:04

I created the wiki-page HOWTOs which contains links to Howtos. They should provide minimal documentation on how to develop, how to setup, how to do this and that. The first howto is HowtoSetup . I recommend Wikipages for HOWTOs to Start with "Howto" and than be camel cased with a short and concise name in order to find them easily and link them easily (so that wiki: within rectangular brackets is not needed). To "overcomplicate" things, here is a regular expression in PCRE format for valid Howto-Wikipages:

```
Howto([A-Z][a-z]+)+
```

2010-12-13 10:34

I spent the last two hours reorganizing stuff (as well as having a shower and eating pizza), in order to have a clean codebase for integrating form validations (Changes: r1476, r1477, r1478, r1479).The following directory structure is now (more-or-less) *official*:

```
web/
  xsl/
    common/ -- scetris common stuff lives here
      global.xsl
      forms.xsl
    lego/ -- lego stuff lives here: *no* scetris stuff
      bricks.xsl
      form-builder.xsl
    <modname-as-mounted>/
       <modname-as-mounted>.xsl
       <modname-as-mounted>.<action>.xsl
```

I have only moved the modules my, shocase and imex so far. The other will follow later on. I did not port them, since I would also have to change import-paths. However, the *new* bricks.xsl etc. are subject to change and already cleand up internally, so this would break things if I only moved them.

A word on the files:

- **bricks.xsl** is what was lego.xsl, but without legacy templates (and I don't want them in there). It contains templates for form controls (input-text, input-number, ...)
- **form-builder.xsl** is new, since andres formbuilder was something like bricks is. The form-builder really builds forms and is only useful together with Module.put(String $name, Form $form)
- **global.xsl** is a copy of fuberlin/global.xsl with the corrects paths adjusted and lecacy stuff removed. This is the scetris-main-template with the look'n'feel of our universities corporate design
- **forms.xsl** contains little reusable forms. As of this writing this is only the login-form.

2010-12-14 22:17

Since I'm moving around nearly all files which belong to scetris (Modules + Templates) I'm just on about restructuring the whole menu-flow. I'll be using the left menu, which will be accessible via a new template - you know, these mode="script" or mode="content" templates which can be found in most empty templates.

2010-12-14 23:33

I've just decided that the jQuery UI Form Validation plugin I've been using so far isn't as flexible as I thought it was. However, I found one that is easy to integrate instead of the previous one. Here are some articles on the plugin:

- http://docs.jquery.com/Plugins/Validation
- http://www.webreference.com/programming/javascript/jquery/form_validation/
- http://bassistance.de/2007/07/04/about-client-side-form-validation-and-frameworks/

However, I'm weaving them into Lego, so this will go unnoticed for you ;-)

2010-12-15 02:42

Pooh, not only implementing it but also documenting it is a hard job :-) However, since I'm writing the User Management Module and *re*write the current modules along with enhancing and extending the form mechanisms it is growing into a nice framework which is /quite/ usefull. I'm currently very excited about the automatic generation of forms and the data-model.

Long story short, here the final Syntax:

```
public class MyModule extends FriggModule<...> {

  public class MyForm implements Form {

    @Field("first-name") @Pos(1.1) @Min(1)
    public String firstName;
```

```
    @Field("last-name") @Pos(1.2) @Min(1)
    public String lastName;

    @Field("pw") @Min(3) @Pos(2.2) @Max(8) @Repeat
    public Password password;

    @Field("mail") @Pos(2.1)
    public EMailAddress email;

    public void commit() throws Exception {
      // commit stuff here
    }
  }

  @Action public void newUser(MyForm form) {
    put("szimpla", form);
  }
}
```

This is an excerpt from the User-Management-Module. An XSL-T for that might look like this (just the part which is actually interesting is presented):

```
  <xsl:template mode="content" match="vars:meta">
        <xsl:apply-templates mode="form-builder" select="form:szimpla/* | doc
                <xsl:sort select="number(@pos)" />
        </xsl:apply-templates>
  </xsl:template>

  <form:szimpla>
        <form:additional pos="1.0">
                <h2>Username and password</h2>
        </form:additional>
        <form:additional pos="2.0">
                <h2>Login credentials</h2>
        </form:additional>
  </form:szimpla>
```

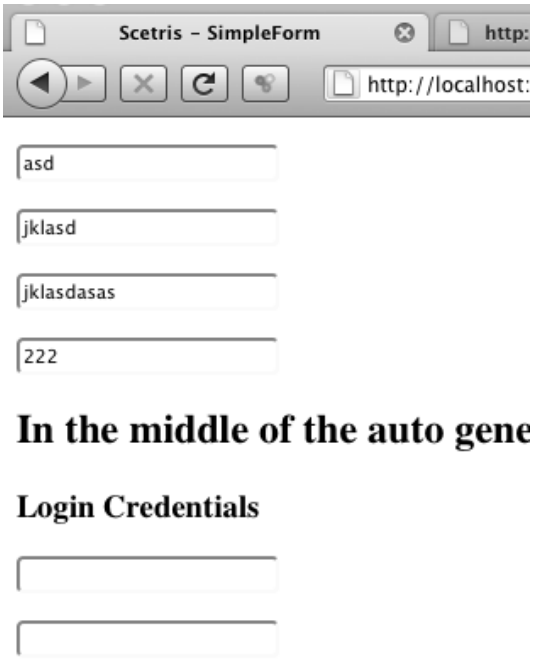
The last part is the interesting. You see, by applying the templates a form will be generated. The position of the form fields is determined by @Pos (thus their position in the Java-File does not matter). Thanks to @Pos you can include your own elements within the auto generated form. This is done by applying the template on its own structure (thats the document("")-part) and intermixing this node set with the auto generated form data (form:szimpla) which has been spit out by weave.

This is a sample how the XML-Output does look like (it's unfortunately not for the above example, I'm much to tired and lazy right now to setup a proper example, I guess you'll get the idea):

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <n0:meta action="" context-path="" first-page="my" first-page-path="/scetris/my" lang="de" logged-in="no" module="showca
  path="/scetris" style="weave.css" type="xml">
  - <n1:szimpla bla="bla">
      <n1:input-numeric name="aNumber" pos="4.0" step="1.0"/>
      <n1:input-text min="1.0" name="firstName" pos="1.0" step="0.1"/>
      <n1:input-text min="1.0" name="lastName" pos="1.1" step="0.1"/>
      <n1:input-text max="3.0" min="3.0" name="loginName" only="abcdefghijklmnopqrstuvwxyz" pos="2.0" step="0.1"/>
      <n1:input-text name="mail" pos="3.0" step="0.1"/>
      <n1:input-text name="pw" pos="2.1" step="0.1"/>
    </n1:szimpla>
  - <n0:user>
      <n2:user>de.fu.weave.AnonymousUser@82f9028</n2:user>
    </n0:user>
  - <n3:req method="get">
      <n3:arg name="~type">xml</n3:arg>
    </n3:req>
  </n0:meta>
```

That structure was generated by the put()-Method. It's rendering by lego looks like this:



It's not that spectacular, use your imagination. What you cannot see in the picture is, that for example the numeric field accepts only numbers etc. Also, when submitting the form, it will automatically be checked by jQuery and again by Weave! All resulting from the initial definition of a Form using Annotation. Ouu I love Annotations.

Let me summarize what is possible right now:

- Express a data model as java-class using java-classes as types
- Java-types are rendered as form-controls by lego
- They are converted and checked by weave as well as by jQuery
- Forms are totally auto generated, however, custom party can be mixed in with arbitrary pos-values (I recommend using doubles in the style of a table of contents)

Some types, like Range, EMailAddress and Password, are from junction; I had to create my own since Java does not feature them by default. Have a look at trunk/src/de/fu/junction/data@1492.

2010-12-17 14:18 14:20

Just made a rather large commit in r1501 (read the comment, it's quite qood documented). Work on!

2010-12-18 21:13

It is rather convenient to have a **watch** on catalina.out. I do the following (working directory is trunk):

```
watch -n 1 | tail -n 20 tomcat/logs/catalina.out
```

This will steadily (1sec) display the last 20 lines of catalina.out . watch is included with most linux-distributions, under OS X i had to install it using Mac Ports, cygwin may or may not have it installed per default.

I'm using this to do debugging (the old school way) - by inserting System.out.println where I want to see a value. The Line will display in catalina.out.

It is also worth noting that you do not need to restart tomcat if it fails to load an XSL-Template. If you symlinked web/xsl it will attempt to load the template until it is successfully loaded (this is standard behavior, since I wrote the template manager in a way that it automatically flushes and reloads templates if they contain errors or need too much memory).

current progress

- r1520 AbstractForm (large comment)

▶ **Attachments**