Fornax-Platform : 5.4. SmartClient (CSC)

This page last changed on Jan 21, 2012 by patrik_nordwall.

Introduction

SmartClient is GUI client for Sculptor business tier. It's NOT using guidesign file and is fully driven by reflection and annotations read from classes generated for Sculptor BusinessTier. For now it's doing only basic things but it will improve in time.

All code is now copyrighted to my employeer (Factory4Solutions a.s.) which allow me to work at this project. If you will find some commercial usage for this project don't hesitate to contact us even with some success story or anything else where we can speed up progress of your adoption. Code is distributed under Apache 2.0 licence (primary Sculptor licence). My name is Pavel Tavoda and now I'm responsible for developing SmartClient GUI for Sculptor. If you have something to say, you can Gmail me at my account "pavel.tavoda".

Technology

Whole smartclient frontend is based on smartclient library from Isomorphic Software and respective Google Web Toolkit (GWT) binding called SmartGWT developed by Sanjiv Jivan's. Both libraries are LGPL however you can order support and additional components from Isomorphic Software.

Screenshots

All screenshots bellow are from client which is running from default sculptor generated business tier. No tweaks or any kind of customization. Everything you see is automatically available.

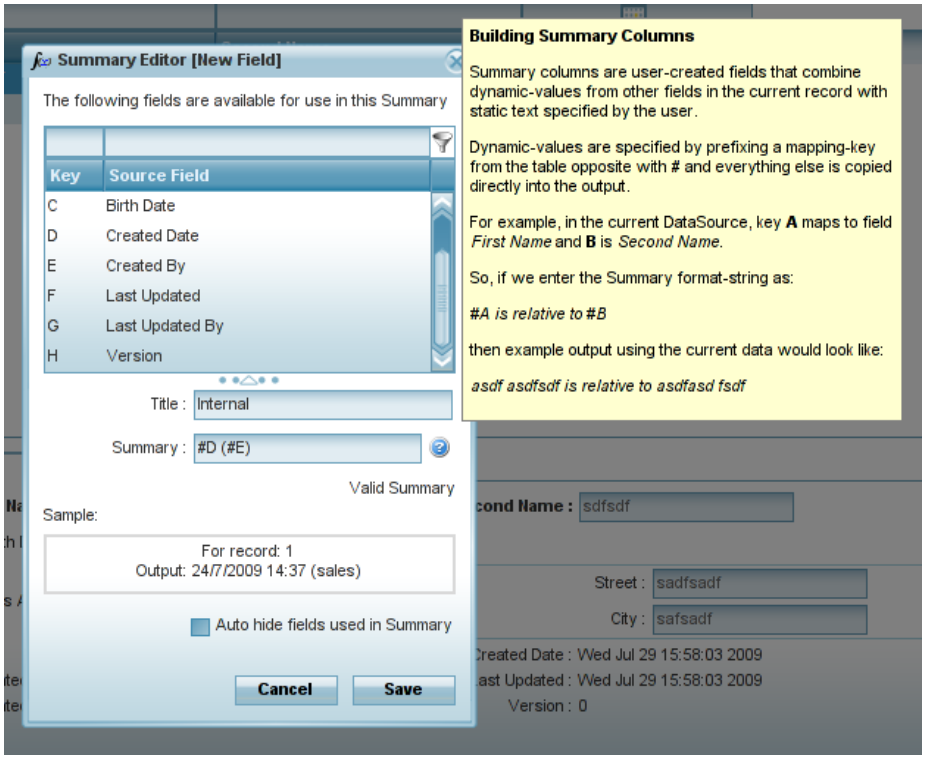
Full 1

Full 2

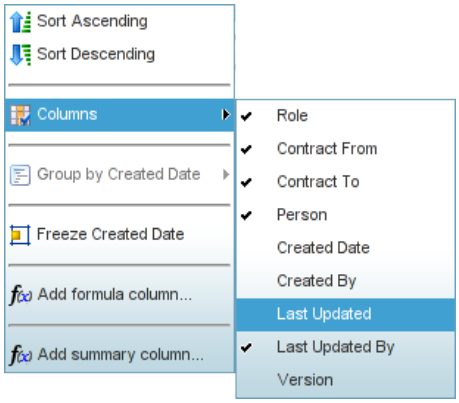
Full 3

Full 4

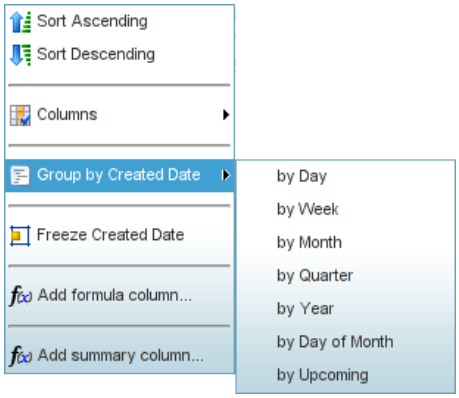
Adding summary column



Column selection



Column group operations



Features

Already working features

- CRUD - simple applications with findAll, findById, save, delete (scaffold operations) are supported
- Normal fields, Assocations (1:N, N:M), Aggregations (1:1, 1:N), Basic type support (not for key columns)
- Some additional carefully designed service operations should be called too
- Tree support - via parent attribute (screenshot: Full 1)
- 2 layouts, 5 skins (screenshot: Full 1-3)
- Flexible layout change (screenshot: Full 3)
- Automatic client side filtering, sorting when all data are fetched (SmartClient feature)
- Server side paging (paging between browser and server, and also paging between server and database when available read Advanced Tutorial - Paging section)
- Status support - if you have field named "status" in your entity it's automatically readonly and you can process it only on server side
- Dynamic rule enforcing - every call to every service is going through rule engine (Drools) where you can dynamically enforce additional rules. From simple validation to more complex rules like: "if somebody add new employee to system, than start task for IT department to arrange all required tools for him".
- COMET support - you can push actions from server. Whole infrastructure on client and server is ready. Server side implementation isn't optimal. However it should work on every server. For now you can for example show message box for user 'admin' when somebody change informations about company (of course when he have correct privileges). This functionality is available as normal Service. You can fire event also from rule engine.
- Plugable server side GUI decorators - you can change service supported operations, for example add "Report" button to specific screens
- Setting list grid on GUI - columns order and width, calculated columns and storing predefined list setup for every user or as "default" for everybody (screenshots: Adding summary column, Column selection, Column group operations)

- Security - Spring security is used, you can customize security by Spring XML or annotations. Spring-security is now integrated with sculptor service context. Users are stored in DB and you can edit them by smarclient GUI.
- Client side debuging console (SmartClient feature, just add 'showConsole=true' to URL parameters and reload page)
- Predefined services required for proper client behaviour (Property storage, GUI datasource, Client command - COMET support, Service repository, ...) for example on screenshot 'Full 4' you can see GUI datasource, whole GUI is driven by this data  .
- Server side filtering
- Data paging between server and database
- Status field support - customizable GUI behavior depending on status
- Customization of many GUI aspects through properties
- Automatic iteration support for one entity service methods - for example delete method should be placed bellow list, you will select multiple rows and after pressing "delete" method delete will be called for every row automatically

Prepared features (near future)

- Support for service methods with more parameters - after pressing button, dialog will come where you can enter additional method parameters and after pressing OK, method will be invoked on server (even complex parameters with DropDown combo boxes for choosing relations or aggregations)
- Custom hook methods for various GUI actions or events
- Editing directly in list (no detail required) for simple structure
- New action in detail will ask for subclasses if some exists

Long run features

- Integration with ServiceMix4 ESB (Eclipse Swordfish, FUSE ESB) out of box
- Drools workflow integration with process instance persistence to DB

How to create smartclient for my project

Setup standart business tier project

1. Start following command which create standalone business tier project called 'sctest'

mvn archetype:generate -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-sculptor-archetype-standalone -DarchetypeVersion=2.1.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public

Fill in groupId and artifactId:

```
Define value for groupId: : org.sctest
Define value for artifactId: : sctest
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  org.sctest: :
```

2. Go to sctest directory. All other steps have to be done here

3. Add following to pom.xml to dependencies section:

```
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-core</artifactId>
            <version>2.0.2</version>
        </dependency>
        <dependency>
```

```
            <groupId>org.fornax.cartridges</groupId>
            <artifactId>fornax-cartridges-sculptor-smartclient-bt</artifactId>
            <version>${sculptor.version}</version>
        </dependency>
```

4. OPTIONAL - Start 'mvn eclipse:eclipse' and import project to Eclipse

5. Add following to src/main/resources/generator/sculptor-generator.properties

```
scaffold.operations=findById,findAll,findByCondition,save,delete
findAll.paging=true
generate.parameterName=true
generate.spring.dataSourceSupport=true
generate.module.gui=false
```

6. Replace your src/main/resources/templates/SpecialCases.xpt with following:

```
«IMPORT sculptormetamodel»
«EXTENSION extensions::helper»
«EXTENSION extensions::dbhelper»
«EXTENSION extensions::properties»

«AROUND templates::jpa::JPA::persistenceUnitAnnotatedClasses(String unitName) FOR Application»
«targetDef.proceed()-»
   <!-- smartclient classes -->
   <class>org.fornax.cartridges.sculptor.smartclient.domain.AuditLog</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.ClientCommand</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.FileUpload</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.GuiDataSource</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.GuiField</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.ListSettings</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.Property</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.PropertyDefinition</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.Roles</class>
   <class>org.fornax.cartridges.sculptor.smartclient.domain.Users</class>
«ENDAROUND»
```

7. Define some entities in src/main/resources/model.btdesign, best is to start with scaffold operations. For example:

```
import "classpath:/model-smartclient.btdesign"

Application MyApp {
   basePackage=org.sctest // If you used different package name in step 1 change this accordingly

      Module base {
            Entity Person {
                  scaffold

                  String firstName;
                  String secondName;
                  String phone;
                  Date birthDate
            }

            Entity Company {
                  scaffold
```

```
                String companyName;
                - @CompanyType companyType;
                - Set<@Person> employees;
        }

        enum CompanyType {
                GOV, CORP, INT
        }
    }
}
```

8. Build project

```
mvn -Dmaven.test.skip=true clean install
```

Setup SmartClient project

9. Go to your project folder (one dir up) and create smartclient web project by invoking following command:

```
mvn archetype:generate -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-
sculptor-archetype-smartclient \
-DarchetypeVersion=2.1.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/
repositories/public
```

Fill in groupId and artifactId (value for artifactId has to be same as business tier project name with '~~something' suffix, important is ~~'):

```
Define value for groupId: : org.sctest
Define value for artifactId: : sctest-smartclient
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  org.sctest: :
```

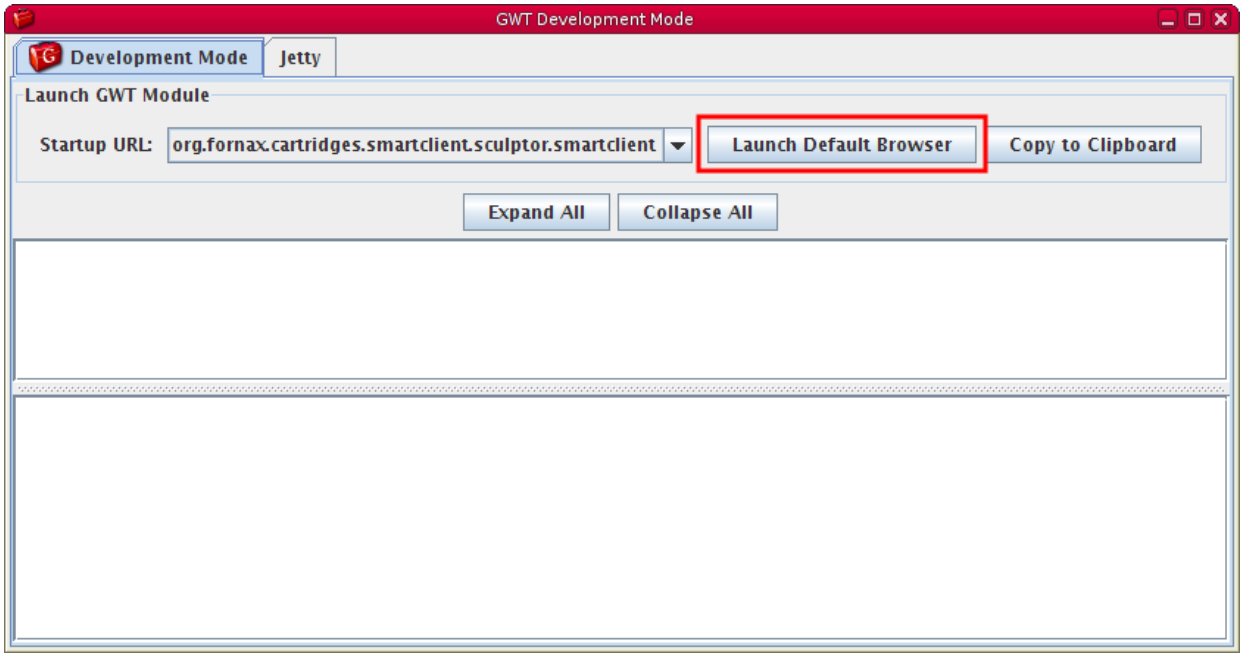
10. Go to sctest-smartclient folder and build project (WAIT till everything complete)

```
mvn -Dmaven.test.skip=true clean install
```

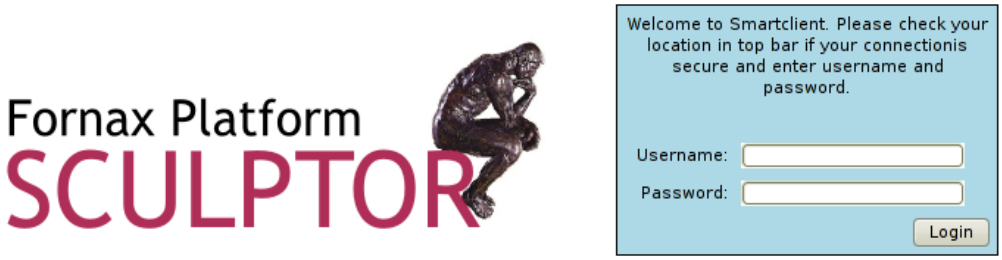
11. Start smartclient

```
mvn gwt:run
```

When you see following launcher, click on "Launch Default Broswer" button.



12. New browser window should appear with login page:



IF NOT, open FireFox and paste following URL to location bar and press Enter:

```
http://127.0.0.1:8888/smartclient/Main.html?gwt.codesvr=127.0.0.1:9997
```

13. Login as admin/admin.

14. When you see following error press OK and paste again previous URL to location bar and press Enter.

15. Start page like this should appear:

Importing Smartclient project to Eclipse

16. Install GWT support to Eclipse.

17. Start 'mvn eclipse:eclipse' in smartclient project. In our sample, inside directory sctest-smartclient.

18. Import project to eclipse: File -> Import -> Existing Project Into Workspace

19. Right click directory "src_launcher" inside project and choose -> Build Path -> Use as Source Folder

If you did optional step 4 you can change dependency to business tier project (sctest). Instead of library, you make dependency directly to project for easier debugging (you don't have to run maven after each change in busines tier project).

20. (OPTIONAL) Remove dependency on business tier JAR project: Right click project (sctest-smartclient) -> Build Path -> Configure Build Path -> Java Build Path -> Libraries. Find sctest-1.0-SNAPSHOT.jar and click "Remove" button.

21. (OPTIONAL) Add dependency directly on business tier project: In same dialog select "Projects" tab. Click "Add" button. Find "sctest" project and click "OK". Close dialog with "OK" button on bottom right corner.

22. Make project GWT compatible: Right click on 'sctest-smartclient' project -> Properties -> Google -> Web Toolkit -> select "Use Google Web Toolkit" checkbox

23. Compile with GWT: Right click "sctest-smartclient" project -> Google -> GWT Compile

24. Right click 'smartclient.launch' -> Run As -> smartclient. Next time you can use directly Run -> Run History -> smartclient or change parameters through Run -> Run Configurations ... . Or you can start debugging with Run -> Debug History -> smartclient.

25. Login page should appear in your default browser. If not, start browser and use following URL:

```
http://localhost:8888/smartclient
```

or if you want to debug frontend code (on-the-fly recompilation - details in GWT manual)

```
http://127.0.0.1:8888/smartclient/Main.html?gwt.codesvr=127.0.0.1:9997
```

Don't forget to copy/paste to browser location bar after login AGAIN.

26. Login as admin/admin

Possible tweaks

Changing database

By default HSQLDB is used. This is in-memory database and all data are lost after you stop server. Because many things are dynamic in SmartClient they are stored to database. For this SmartClient need some supporting services. They are defined in smartclient-bt project which is included in your model.btdesign file in business tier project.

Security

Whole security setup is in war/WEB-INF/Security.xml. By default in-memory authentication provider is used with two users (user,admin) and two roles (ROLE_USER, ROLE_ADMIN). It's for demonstration purpose only. For more details about security setup look to Spring Security Manual. In Security XML you have predefined also database based authentication with SHA256 for password encoding with username salt. Also services are predefined. For inserting admin user to database you can use following insert statement:

```
INSERT INTO users
 (ID,USERNAME,PASSWORD,ENABLED,EXPIREAT,CREATEDDATE,CREATEDBY,LASTUPDATED,LASTUPDATEDBY,VERSION,SELECTED
 VALUES
 (1,'admin','a4a88c0872bf652bb9ed803ece5fd6e82354838a9bf59ab4babb1dab322154e1','1','2012-01-01',
 '2009-01-01 01:01:01','system', '2009-01-01 01:01:01','system',0,'en');

INSERT INTO ROLES (ID,ROLENAME,CREATEDDATE,CREATEDBY,LASTUPDATED,LASTUPDATEDBY,VERSION)
 VALUES (1,'ROLE_ADMIN', '2009-01-01 01:01:01','system', '2009-01-01 01:01:01','system',0);

INSERT INTO ROLE_USERS (ROLE,USERS) VALUES (1,1);
```

Tweak it for your database. Of course after you change database from default HSQLDB to your preferred one. Read in previous chapter.

Using the model in smartclient-bt project

If you need references from your model to the elements in fornax-cartridges-sculptor-smartclient-bt you need to do 2 things.
Add import in your model.btdesign

```
import "classpath:/model-smartclient.btdesign"
```

Define that generation of that module should not be done, in sculptor-generator.properties:

```
generate.module.gui=false
```

Changing GUI behavior

You can do it by 2 way. First is defining annotation on services and second by defining properties. For development purpose is best to put it to default.properties. They are loaded to database after each application start. However you can override property value in GUI (database) in PropertyService. Once you override it, new value is used instead of value from default.properties.

Changing field order in detail:

```
productService.fieldOrder=name, description,activeFrom, activeTo, owner, id, lastUpdatedBy:s
```

:s at the end of field mean that it will start at begging of new row

You can customize each field by setting jsonPostfix. You can define new properties or override existing properties for fields e.g.:

```
orderService.orderName.jsonPostfix=canEdit:false,required:false
```

Here is about 160 properties you can define which can change field behaviour at different levels. As datasource field, column in list and also as field in detail. For properties look to Smartclient documentation at http://www.smartclient.com/docs/7.0rc2/a/b/c/go.html and attributes of DataSourceField, ListGridField and FormItem. Some more interesting examples:

```
orderService.jsonPostfix= showFilterEditor:true, initialCriteria:{isSummaryTransaction:"Y"}
savingsService.clientAccount.jsonPostfix=pickListCriteria:{accountType:"TRG"}
detailService.textRef=ticker,name
```

You have to understand which property to define and change. You can also define properties which can change behavior in JavaScript. Be careful you can screw things up. Look for Smartclient documentation and showcase with sources at http://www.smartclient.com/index.jsp#_Welcome for real meaning of properties.

textRef is used for defining columns of drop down pick lists for N:1 and 1:1 relations. First field is also used for defining searching when you start to type inside combo box. Value is defined on related service e.g.:

```
orderDetailService.textRef=ticker,name
```

Tab where service is shown is by default deduced from package name. You can reorder service location in tabs by defining tab property e.g.:

```
auditLogService.tab=Services
clientCommandService.tab=Services
fileUploadService.tab=Services

clientService.tab=Client
clientAccountService.tab=Client
```

Internal structure

FAQ

Q: Why did you use smartclient library

A: smartclient library is most complete JavaScript library available today. Primary advantages in short: LGPL licence, nice pixel perfect skins, many components, clever data handling support build inside library (data sources with lazy loading), GWT binding (we should develop in Java). For all features look to SmartGWT Showcase.

Q: Why don't you generate code from design file

A: I like dynamic approach which AJAX approach and smartclient library allow. After every manual GUI customization you add nail to coffin of your inflexibility. Later when you decide to change your data model you have to trace all dependencies on client side. With dynamic approach whole GUI react on changes in your service tier. Development is more iterative, faster. Many customization should be done directly by user, no developer have to be involved (for example columns order and visibility in lists, calculated columns, ...)