

Spago Studio

Authors

Andrea Grassi
Roberto Scariello



Index

VERSIONS HISTORY	3
DOCUMENT GOAL.....	3
REFERENCES	3
1 WHAT ABOUT SPAGO STUDIO.....	4
1.1 WHAT ARE THE BENEFITS OF USING STUDIO ?	4
1.2 HOW DOES IT WORKS ?	4
1.3 WHAT CAN BE ARCHIVED WITH STUDIO?.....	5
2 STUDIO BUILD PROCESS	5
2.1 STUDIO DATA MODEL	6
2.2 TEMPLATES	7
2.3 TEMPLATES GROUPS	8
2.4 FIELDS TYPE	9
2.5 ENVIRONMENT	10
2.6 DICTIONARY	11
2.7 LOG FIELDS TYPE	12
2.8 LOG FIELD	13
2.9 TABLES.....	14
2.10 FIELDS TABLE	16
2.11 RELEATION TABLES	16
2.12 JOIN FIELDS.....	18
2.13 DECODING FIELDS	18
3 CLASSIC MODELS SAMPLE APPLICATION	19
3.1 DEMO PURPOSE	19
3.2 HOW TO BUILD “PRODUCTLINES” CRUD FUNCTION	19
3.2.1 <i>Step 1: Import table definition</i>	20
3.2.2 <i>Step 2: Generate java code and configuration files.....</i>	21
3.2.3 <i>Step 3: Add new generated files to the application.....</i>	21
3.2.4 <i>Step 4: Add a decorator to a column into the list module.....</i>	22
3.3 HOW TO BUILD “PRODUCTS + ORDERDETAILS” CRUD FUNCTION	24
3.3.1 <i>Step 1: Generate Products CRUD function</i>	24
3.3.2 <i>Step 2: Generate OrderDetail CRUD function</i>	24
3.3.3 <i>Step 3: Add relation from OrderDetail to Products.....</i>	25
3.4 HOW TO BUILD A LOOKUP FROM ORDER-DETAIL TO PRODUCTS	28
3.4.1 <i>Step 3: Add Lookup from ordetails (detail module) to order (list module).....</i>	28
3.5 SOFT DELETE AND AUDITING DEMO FUNCTION	30
3.6 AN EXAMPLE OF MASTER/DETAIL PAGE	31

Versions History

Version/Release n° :	1.0	Data Version/Release :	July, 14th 2008
Description:	First release (English version)		

Document Goal

The goal of this tutorial is to provide you with a simple introduction about Spago Studio basic concept, use and personalization. We will use the template *Studio.war* released in *studio-sample.zip*

References

You can find more information about Spago framework in the following documents (now in Italian, as soon as possible in English version) at:

http://spago.eng.it/docs_it/documentation/index.html :

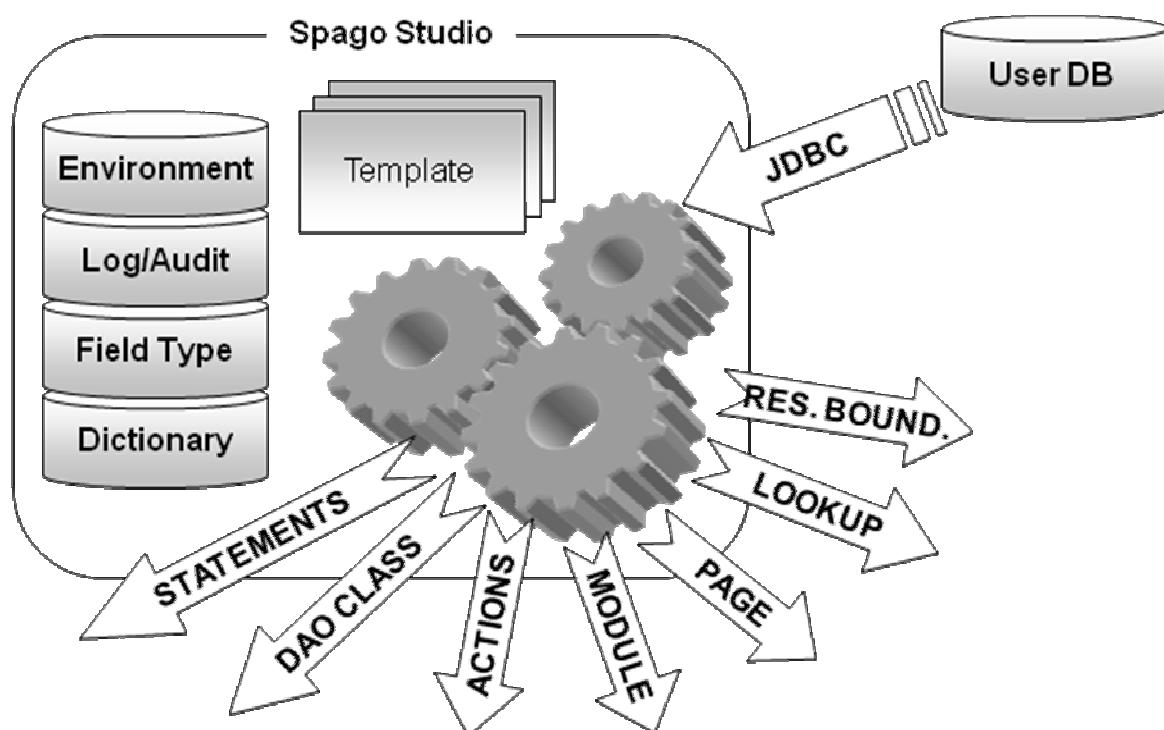
- [1] *Spago Overview*
- [2] *Spago User Guide.*

1 What about Spago Studio

Spago Studio application is a platform for building web based applications with many “CRUD” function to handling and feeding complex database structure.

It's a Web Application (built with Spago) with many wizards to automate Spago components editing and creation.

It can be used to rapidly prototype and develop data entry and reporting applications that work as a service over the internet, or a local network.



1.1 WHAT ARE THE BENEFITS OF USING STUDIO ?

Using Studio, it is very easy to adapt to changes of database tables. With a single click in Studio, you just generate a new “CRUD Function” based on your new table definitions.

1.2 How does it works ?

Using JDBC connection, Studio read database metadata such column name, column type or column length and use this information to generate SQL statements and data access object (DAO) java classes.

Studio use own legacy data to lead generation process behavior; column type for example, is referenced into Field Type table data were Studio finds many characteristics like decoration (formatting rule), validator, widget, ...

Success key for Studio generation process is a well formed source data base with correct use of column name and column type (think for example at using of Varchar instead of Char). For the best generation, the same column name should be used in all table for to reference to the same information.

1.3 WHAT CAN BE ARCHIVED WITH STUDIO?

- It is a time saver and simplify application development: for many simple function, user don't have to write any java code.
- It generates high performance and simple java code and sql statements. Studio import table definitions directly from the database so that you can start rewriting your application immediately with 50% of the work done automatically.
- It generates human-readable e customizable source codes because all starts from template.
- The generated codes and statements are expected to be bug-free.

2 Studio build process

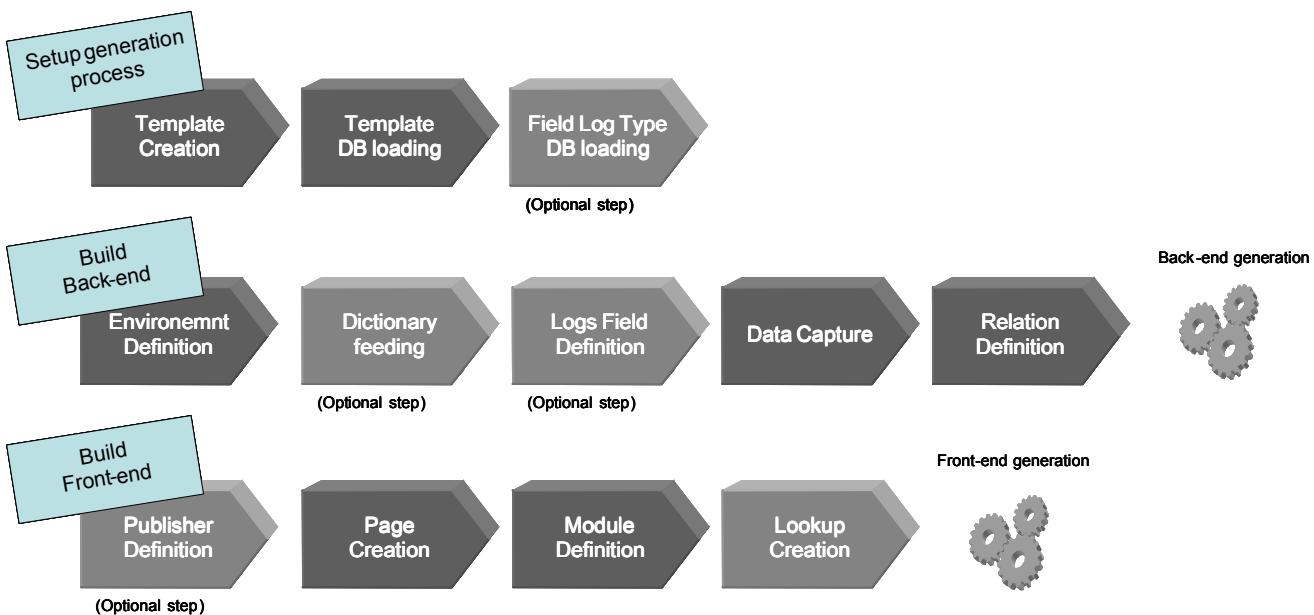
The build process start by personalization of template that will be use in generation process. The next step is the feeding of the dictionary table and other Studio basically legacy data.

After the previous setup phase user can start generation of back-end DAO object (java class, sql-statements, commons and specific data handle rules) than follow the generation of front-end function (pages, modules and lookup).

There are 3 groups of Studio application functions:

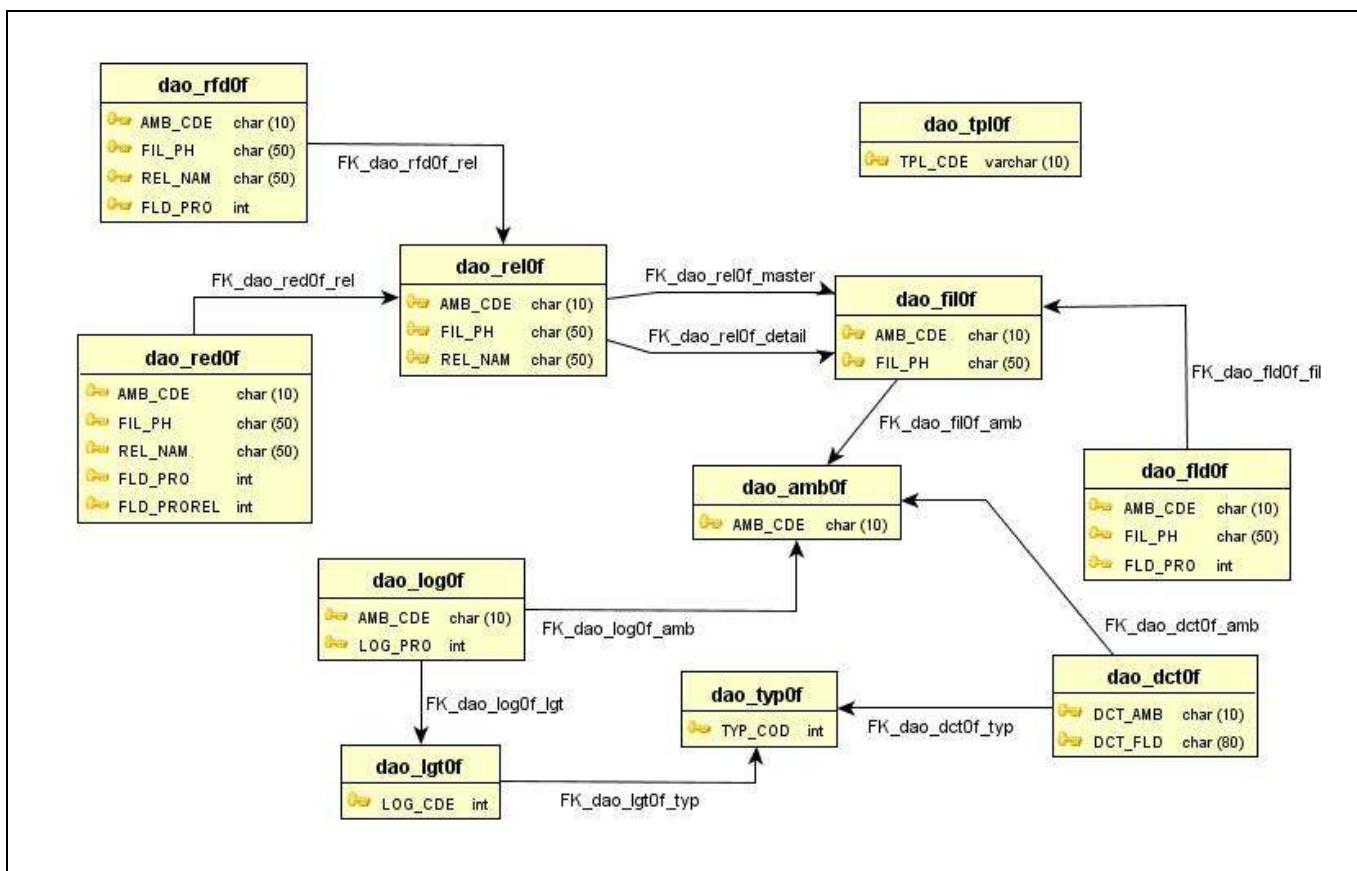
1. Domain management function to feed Studio legacy data required by generation process
2. Dao function for backend management
3. Front-end generation function

The following schema show the process generation flow.



2.1 STUDIO DATA MODEL

The following picture show table and relation of Studio legacy data model.



2.2 TEMPLATES

Templates are simple text fragment use from Studio to generate code or xml configuration files.

They contains static text and variable text (marked by the symbol “<? ... ?>”) that is change from Studio during generation process.

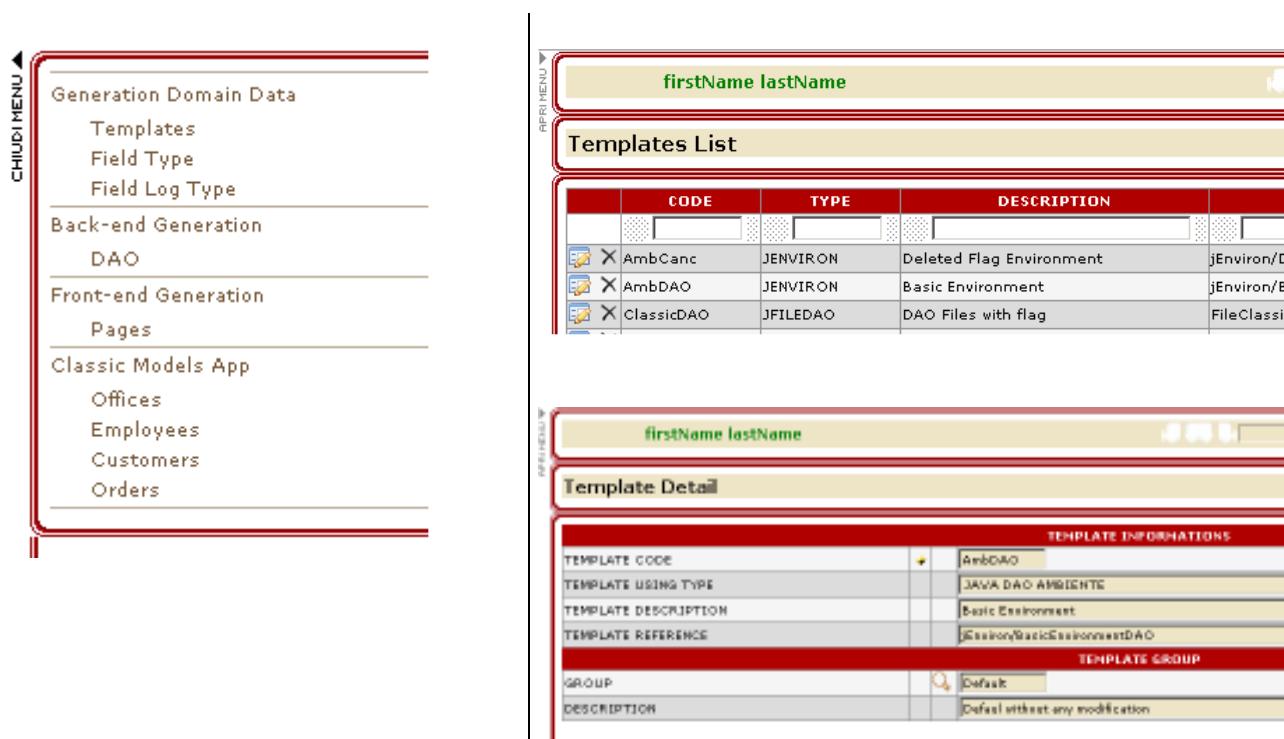
Use of templates permits many personalization of the generation phase without any java code change.

All template are stored into text files (one file for one template) and they are putted in the same folder “template” located into configuration path.

Studio reference and categorize all template file by the Template Reference Table.

dao_tplOf		Template Reference Table			
Fields					
Field	Type	description		Null	Key
TPL_CDE	varchar(10)	Template Code		NO	PRI
TPL_DES	varchar(50)	Template Description		NO	
TPL_REF	varchar(80)	Template Resource Reference		NO	
TPL_TPY	varchar(10)	Template Type for grouping sames template type (use to load list-box)		NO	
TPL_GRP	varchar(10)	Optional template group (used by tables and modules/validations)		YES	

User should be editing this table by the following specific Studio function.



The screenshot shows the Spago Studio interface with two main windows open:

- Template List:** This window displays a list of templates. The columns are CODE, TYPE, and DESCRIPTION. The data is as follows:

CODE	TYPE	DESCRIPTION
AmbCanc	JENVIRON	Deleted Flag Environment
AmbDAO	JENVIRON	Basic Environment
ClassicDAO	JFILEDAO	DAO Files with flag
- Template Detail:** This window provides detailed information for a selected template. The template details are:

TEMPLATE INFORMATION	
TEMPLATE CODE	AmbDAO
TEMPLATE USING TYPE	JENVIRON
TEMPLATE DESCRIPTION	Basic Environment
TEMPLATE REFERENCE	FileClassic
GROUP	Default
DESCRIPTION	Default without any modification

The following text is a simple xml-field-type template file for handling static “Yes/No” list-box field widget.

```
<FIELD id="<?ID?>" name="<?Name?>" size="<?FieldSize?>" maxlength="<?FieldMaxLength?>"  
widget="COMBO" onkeypress="<?FieldOnKeyPress?>" onkeyup="<?FieldOnKeyUp?>"  
decorator="<?FieldDecorator?>" template="<?template?>" >  
    <INSERT is_readonly="<?FieldInsertReadOnly?>" is_visible="<?FieldInsertVisible?>" >  
        <OPTION VALUE="Y" DISPLAY="Yes" />  
        <OPTION VALUE="N" DISPLAY="No" />  
    </INSERT>  
    <UPDATE is_readonly="<?FieldUpdateReadOnly?>" is_visible="<?FieldUpdateVisible?>" >  
        <OPTION VALUE="Y" DISPLAY="Yes" />  
        <OPTION VALUE="N" DISPLAY="No" />  
    </UPDATE>  
</FIELD>
```

Studio implements the following templates type:

Template Type	Description and use
Java DAO Environemnt	Scrap of Java code for Java Environment (super-class of DAO file)
Java DAO File	Scrap of Java code for DAO file (a table with his relation)
SQL	Scrap of SQL statements
XML Graph	Scrap of graph xml-configuration
XML Module	Scrap of module xml-configuration
XML FieldSet	Scrap of fieldset xml-configuration
XML Page	Scrap of page xml-configuration
XML Action	Scrap of action xml-configuration
XML Column	Scrap of column xml-configuration
XML Field Widget	Scrap of field widget xml-configuration
XML Field Validator	Scrap of validator xml-configuration
XML Field Publisher	Scrap of publisher xml-configuration
XML Lookup	Scrap of lookup xml-configuration
XML Lookup Field	Scrap of lookup-field xml-configuration
XML Tab	Scrap of tab xml-configuration
XML CRUD	Scrap of CRUD specialized function (one shot generation)

See next paragraphs for detail information about template using.

2.3 TEMPLATES GROUPS

Templates groups makes possible to link different templates according to their logical function.

In this way you can join a module template to it's relative validation template. Making this connection permits to you to reduce issues caused by selecting an incorrect template for a specific need.

Templates groups are optional, if you don't select a specific group for a template this would be present as a wildcard when you try to select one of that type.

Each template could reference a template group inserted into the Template Group Table.

dao_tpgOf
Template Groups Table
Fields

Field	Type	Description	Null	Key
TPL_GRU	varchar(10)	Template Group Code	NO	PRI
TPL_DEGS	varchar(50)	Template Group Description	NO	

You can access, create and edit Template Groups directly from Templates detail with the lookup function.

The screenshot shows two stacked windows of Spago Studio. The top window is titled "Template Groups List". It has a header bar with "firstName lastName" and various icons. Below is a table with columns "GROUP" and "DESCRIPTION". The table contains six rows, each with a small icon and a name: "Auditing", "Default", "DeleteSoft", "DetailMod", "ListMod", and "SearchMod". Each row also has a "Delete" icon. The bottom window is titled "Template Groups Detail". It has a header bar with "firstName lastName" and icons. Below is a table with columns "GROUP" and "DESCRIPTION". The "GROUP" column has a dropdown arrow icon. The "DESCRIPTION" column has a large yellow rectangular highlight.

2.4 FIELDS TYPE

Fields Type information are used by Studio for basic generation (more complex generation use dictionary table). In the basic generation mode, Studio for each SQL-Type looks for related field-type and builds field with this field characteristic.

Every SQL-Type has at least one correlation to field type.

During the generation process in basic mode, Studio choose always the first field-type found for each SQL-Type.

dao_typOf
Field Type Table

Fields

Field	Type	Description	Null	Key
TYP_COD	int(11)	Type code reference	NO	PRI
TYP.Des	char(50)	Type Description	YES	
TYP_CLA	char(80)	Name of decorator (see Spago decorators.xml configuration file)	YES	
TYP_VLD	char(80)	Name of validator (see Spago fieldvalidators.xml configuration file)	YES	
TYP_WDG	char(80)	Widget template name	YES	
TYP_SQL	int(11)	Associated SQL Type	YES	
TYP_INL	int(11)	Max length (internal on DB) think for example to the date numb. 20080710	YES	
TYP_OUL	int(11)	Max length (external on Field) for ex. A date with sep: 2008/07/10	YES	
TYP_VIN	char(80)	Name of validator for Insert	YES	
TYP_VUP	char(80)	Name of validator for Update	YES	
FLD_VIS	char(1)	"Y" if field is visible	YES	
TYP_INS	char(1)	"Y" if this field is use in Insert	YES	
TYP_UPD	char(1)	"Y" if this field is use in Update	YES	
TYP_DEF	Char(1)	"Y" if this is default to use for a SQL Type	YES	

For the best use, this table should contains user defined types (fiscalCode, cap, amount, ...). Defined user field type will be reference by the dictionary.

2.5 ENVIRONMENT

Environment is a tables container.

An environment collect a group of comparable tables with many commons characteristics.

dao_ambOf

Environment Table

Fields

Field	Type	Description	Null	Key
AMB_CDE	char(10)	Environment Code	NO	PRI
AMB_Des	char(50)	Environment Description	NO	
AMB_PACK	char(50)	Environment Java Class Package	NO	
AMB_POOL	char(25)	Name of connection pool (see data_access.xml Spago conf. file)	NO	
TPL_CDE	char(10)	Generation Template for environment DAO super class	YES	
ENV_TYP	Int	0 for a DAO Environment, 1 for a PCML Environment	YES	
TPL_GRU	varchar(10)	Code for the templates group	YES	

For each environment defined, there is a DAO java super-class with the commons methods for all extended DAO java classes.

API MENU ▶

firstName lastName

Environment list

	CODE	DESCRIPTION	TEMPLATE
A X	CLASSICLOG	Classic Models Environments with log fields	AmbCanc
A X	DAOCLASSIC	Environment Classic Models	AmbDAO
A X	NEWTEST		AmbDAO

ResourceBundle generation
 Java Environment class generation
 DAO x this Environment
 Dictionary x this Environment
 Log Field x this Environment

The following picture show the needed information for the environment definitions.

API MENU ▶

firstName lastName

Environment detail

ENVIRONMENT INFORMATIONS	
ENVIRONMENT CODE	DAOCLASSIC
DESCRIPTION	Environment Classic Models
TEMPLATE	BASIC ENVIRONMENT
PACKAGE	it.eng.common.dao
CONNECTION POOL	CLASSICMODELSPOOL
TEMPLATE GROUP	DEFUALT WITHOUT ANY MODIFICATION

Environment Template creation is a prerequisite for the environment definitions.

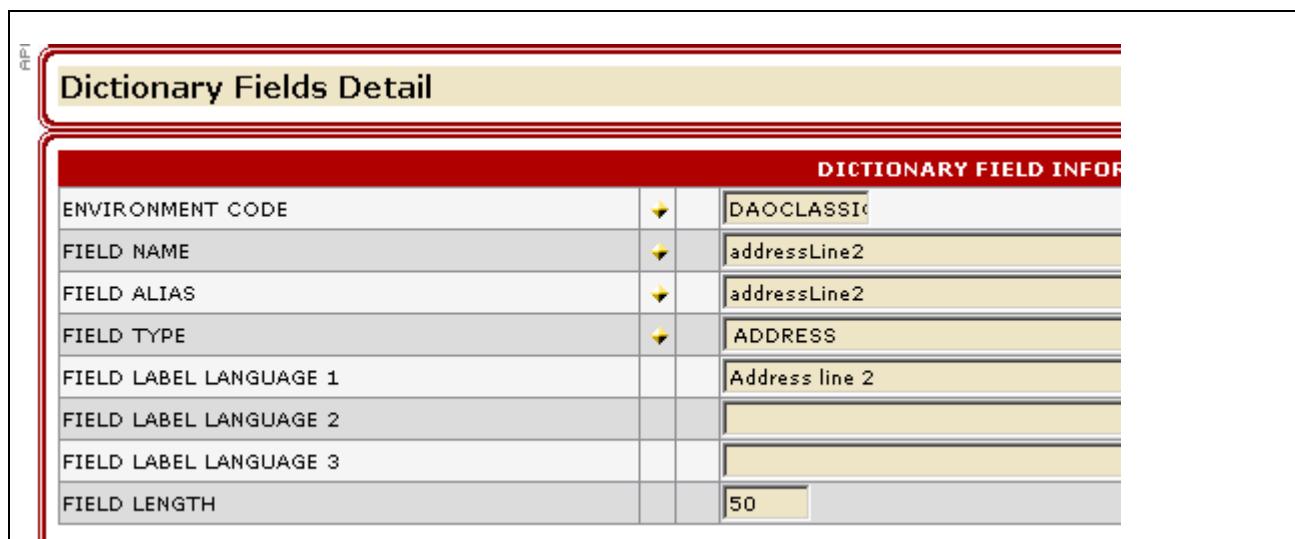
2.6 DICTIONARY

The data dictionary contains definitions for fields as used in generation process.
For each field name, this table define a specific data type (Field Type dao_typ0f).

Also this table contains some override characteristic of the referenced FieldType.

dao_dct0f		Dictionary Table
Fields		
Field	Type	Description
DCT_AMB	char(10)	Environment code for this dictionary
DCT_FLD	char(80)	Field name reference
DCT_WEB	char(80)	Field name in module and statements (alias name)
DCT_TYP	Int(11)	Referenced type in dao_typ0f
DCT_L01	char(80)	First language defined label
DCT_L02	char(80)	Second language defined label
DCT_L03	char(80)	Third language defined label
DCT_LEN	Int(11)	Input length (override of TYP_OUL)
DCT_TYD	char(1)	<i>Reserved for Future Use</i>
DCT_DLN	Int(11)	<i>Reserved for Future Use</i>

There is a different dictionary for each created environment.



DICTIONARY FIELD INFOR							
ENVIRONMENT CODE	▼	DAOCLASSIC					
FIELD NAME	▼	addressLine2					
FIELD ALIAS	▼	addressLine2					
FIELD TYPE	▼	ADDRESS					
FIELD LABEL LANGUAGE 1	▼	Address line 2					
FIELD LABEL LANGUAGE 2	▼						
FIELD LABEL LANGUAGE 3	▼						
FIELD LENGTH	▼	50					

2.7 LOG FIELDS TYPE

It is sometimes necessary to keep track of what changes were made to the database, and by whom. This is known as Audit Logging or an Audit Trail; also many application require a delete logic using for example flag or extinction date.

All this behaviours are managed by the environment for the group of table defined.

This table contains reference type for the log field; any type specify a sql-type by the related Field Type and a java code.

Example of use:

Log Fields Type Detail

LOG FIELD TYPE INFORMATION		
LOG FIELD DESCRIPTION		Deleted flag
LOG FIELD JAVA CODE		getCurrentSqlDate()
LOG FIELD DATA TYPE		DATE

Generated java code for this log field type:

```
addParameter(_dataConnection.createDataField("", Types.DOUBLE,DateTime.getCurrentDate()))
```

Table field description:

dao_lgt0f		Log Field Type Table				
Fields						
Field	Type	Description			Null	Key
LOG_CDE	Int(11)	Log Field Type Code			NO	PRI
LOG_DES	char(50)	Log Field type Description			YES	
LOG_COD	char(100)	Java code			YES	
LOG_DTP	Int(11)	Field Type (use for reference sql-type)			YES	

2.8 LOG FIELD

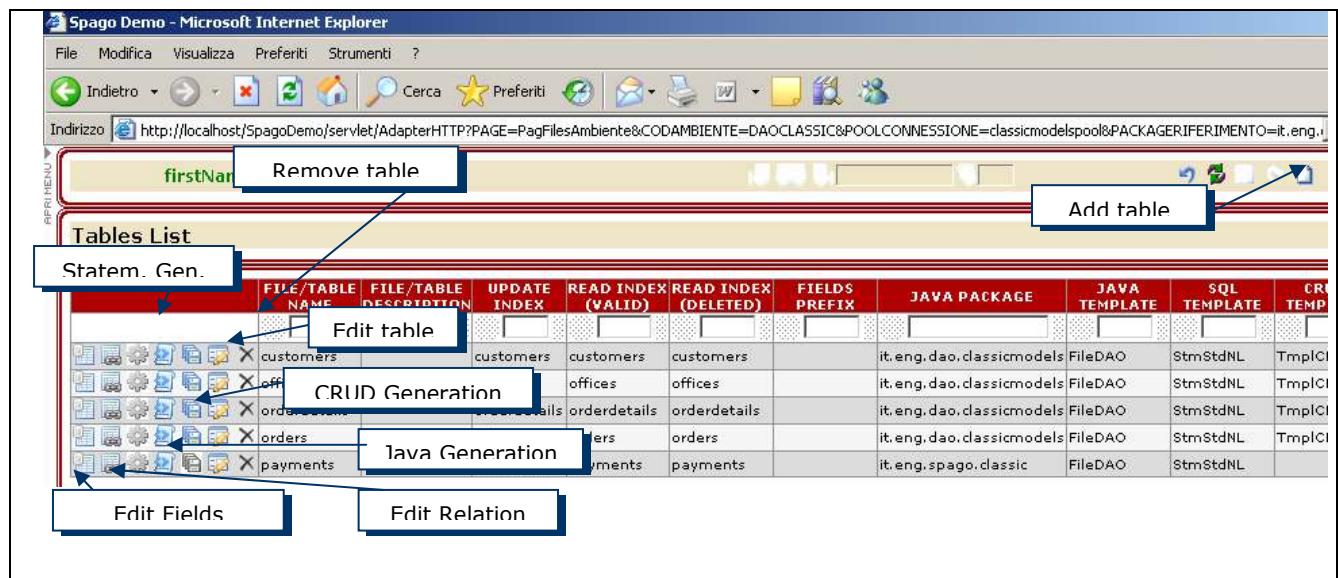
This table contains the log fields for a specific environment (group of table). Please pay attention to the "Log Field Name": it is not a key field because the same field can be reference more times in the same environment.

dao_log0f		Log Field Table				
Fields						
Field	Type	Description			Null	Key
AMB_CDE	char(10)	Environment code for this Log Field			NO	PRI
LOG_PRO	int(11)	Progressive code for order			NO	PRI
LOG DES	char(50)	Description			NO	
LOG_FLG	char(1)	Use for this field (always, only insert, only update, ...)			NO	
LOG_TIP	int(11)	Reference Log Field Type			NO	
LOG_FLD	char(50)	Log Field Name			NO	
GRP_CDE	char(10)	Group code for PCML Groups			YES	

2.9 TABLES

This is table that contains tables definition.

Tables are environment components (see previous environment paragraph); the user can add, remove or modify tables from an environment by the Studio function.



FILE	TABLE NAME	DESCRIPTION	UPDATE INDEX	READ INDEX (VALID)	READ INDEX (DELETED)	FIELDS PREFIX	JAVA PACKAGE	JAVA TEMPLATE	SQL TEMPLATE	CR TEMP
	customers	customers	customers	customers			it.eng.dao.classicmodels	FileDAO	StmStdNL	TmplCI
	offices	offices	offices	offices			it.eng.dao.classicmodels	FileDAO	StmStdNL	TmplCI
	orderdetails	orderdetails	orderdetails	orderdetails			it.eng.dao.classicmodels	FileDAO	StmStdNL	TmplCI
	orders	orders	orders	orders			it.eng.dao.classicmodels	FileDAO	StmStdNL	TmplCI
	payments	payments	payments	payments			it.eng.spago.classic	FileDAO	StmStdNL	

Table definition process start with a lookup into the application database to find all needed jdbc metadata like field name, type, length, keys, ...

The following picture show the detail view for the table creation.

Spago Demo - Microsoft Internet Explorer

File Modifica Visualizza Preferiti Strumenti ?

Indietro ➤ 🔍 Cerca ⚡ Preferiti 🌐

✉️ 📄 🗂️ 📁 🎯 🌐

Press lookup icon to find the request table

firstName lastName	
Table Detail	
FILE/TABLE DEFINITION ENVIRONMENT REFERENCE CODE: DAOCLASSIC	
FILE INFORMATIONS FILE/TABLE NAME: classicmodelspool	
CONNECTION POOL:	classicmodelspool
UPDATE INDEX/VIEW:	
READ INDEX/VIEW (VALID)	
READ INDEX/VIEW (DELETED)	
FILE DETAILS FIELD PREFIX: <input type="text"/> JAVA DAO PACKAGE: <input type="text"/> JAVA TEMPLATE: JAVA FILES TEMPLATE SQL TEMPLATE: DAO STATEMENTS WITHOUT FLAG CRUD TEMPLATE: <input type="text"/> FILE/TABLE DESCRIPTION: <input type="text"/>	

File/Table name is the name of the table (choose from lookup)

Connection pool is a read-only field derived from the selected environment.

“Update Index/View”, “Read Index/View (Valid)” and “(Deleted)” are customizable views name referenced into statement and java template.

A common use of "Update Index/View" in a "logic delete environment", is for testing record existing during insert/update otherwise it's contains the table name.

An example of “Read Index/View (Valid)” it is to populate a list box from a domain table.

Table and field description:

Fields		Tables Table		
Field	char(10)	Description	NO	PRI
AMB_CDE	char(10)	Environment code for this Table	NO	PRI
FIL_PH	char(50)	Table name or for file name in OS400 environment	NO	PRI
FIL_UPD	char(50)	View Name for Update statements or Table name	NO	
FIL_LSV	char(50)	View Name for "only valid inquiry" or Table name	NO	
FIL_LSA	char(50)	View Name for "only cancelled inquiry" or Table name	NO	
FIL_PRE	char(50)	Optional Field Prefix (When all field of the same table has the same prefix)	YES	
FIL_PKG	char(80)	Java package name	YES	
FIL_TPL	char(10)	Java Template	YES	
FIL_STM	char(80)	Sql Statement Template	YES	
FIL_DSC	char(80)	Free description	YES	
FIL_CRU	char(80)	CRUD Template (one shot generation)	YES	



2.10 FIELDS TABLE

This table contains the table fields

Fields		Table Fields Table											
Field	Type	Description										Null	Key
AMB_CDE	char(10)	Environment code										NO	PRI
FIL_PH	char(50)	Reference Table										NO	PRI
FLD_PRO	int(11)	Incremental (numeric key for this field)										NO	PRI
FLD.Des	char(50)	Free field description										NO	
FLD_FLD	char(50)	Field Name										NO	
FLD_WEB	char(50)	Field name in module and statements (alias name)										NO	
FLD_WIDGET	char(25)	Widget template name										NO	
FLD_LENGTH	int(11)	Length in the DB										NO	
FLD_ILEN	int(11)	Max length (internal on DB) think for example to the date numb. 20080710										NO	
FLD_OLEN	int(11)	Max length (external on Field) for ex. A date with sep: 2008/07/10										NO	
FLD_KEY	char(1)	"Y" if this field is part of the table key										YES	
FLD_NUL	char(1)	"Y" if this field is nullable										YES	
FLD_DEC	int(11)	Decimals for numeric type										YES	
FLD_TYP	int(11)	Referenced field type										YES	
FLD_MND	char(1)	"Y" if this field is mandatory										YES	
FLD_INS	char(1)	"Y" if this field is use in Insert										YES	
FLD_UPD	char(1)	"Y" if this field is use in Update										YES	

This table is automatically filled by the import process (during save of the table) and contains copy of the data from dictionary table and from field type table.

A change to dictionary or to field type doesn't automatically update a previous saved table.

This Studio behaviour allow easy management of special case without changing dictionary or field type.

PROGRESSIVE CODE	DESCRIPTION	FIELD NAME	WEB NAME	TEMPLATE WIDGET	LENGTH	INTERNAL LENGTH	EXTERNAL LENGTH	KEY	NULL	DECIMALS	TYPE	MANDATORY IN		
X 10	Office code	officeCode	officeCode	Text	10	50	50	Yes	No	0	50	TEXT	Yes	Ye
X 20	City	city	city	Text	50	50	50	No	Yes	0	50	TEXT	No	Ye
X 30	Phone #	phone	phone	Text	50	20	20	No	Yes	0	140	Phone Number	No	Ye
X 40	Address line 1	addressLine1	addressLine1	Text	50	50	50	No	Yes	0	130	Address	No	Ye
X 50	Address line 2	addressLine2	addressLine2	Text	50	50	50	No	Yes	0	130	Address	No	Ye
X 60	State	state	state	Text	50	50	50	No	Yes	0	50	TEXT	No	Ye

2.11 RELEATION TABLES

Table relation definition must be done after the "import process" (lookup of table, key definition, field generation using types, dictionary and metadata).

A relation caption, present in each row of table list function, lead the user into the relation definition process.

Tables List

Caption for table relation	FILE/TABLE DESCRIPTION	UPDATE INDEX	READ INDEX (VALID)	READ INDEX (DELETED)	FIELDS PREFIX	JAVA PA
	customers	Customers table	customers	customers	customers	it.eng.dao.cl
	employees	Employees table	employees	employees	employees	it.eng.dao.cl
	offices	Offices table	offices	offices	offices	it.eng.dao.cl
	orderdetails	Order details table	orderdetails	orderdetails	orderdetails	it.eng.dao.cl

Relation Detail

RELATED FILE INFORMATIONS

RELATION NAME	SEEK FILE INFORMATIONS
OrderDetail	<input type="text"/> orderdetails <input type="text"/> Order details table <input type="text"/> DAOCLASSIC <input type="text"/> INNER JOIN <input type="text"/> LEFT JOIN <input type="text"/> RIGHT JOIN <input type="text"/> OUTER JOIN

Relations List

RELATION NAME	RELATED AMBIENT CODE	RELATED PHYSIC FILE	RELATION TYPE	...
orderheader	DAOCLASSIC	Environment Classic Models	orders	Inner join

It's possible make a relation with any defined table from all environment; a lookup function lead the user to find the table. SQL generator built in Studio support all relations type.

dao_rel0f

Releation Tables Table

Fields

Field	Type	Description	Null	Key
AMB_CDE	char(10)	Environment code (source)	NO	PRI
FIL_PH	char(50)	Reference Table (source)	NO	PRI
REL_NAM	char(50)	Relation name	NO	PRI
AMB_CDEREL	char(10)	Environment code (destination)	NO	FK
FIL_PHREL	char(50)	Reference Table (destination)	NO	FK
REL_TYP	int(11)	Left Join, Right Join or Outer Join	YES	
SUBREL	char(1)	'Y' if this relation has other sub-relation	YES	

2.12 JOIN FIELDS

For any defined relation, the user must indicate all the related fields.
It's possible relate any field of the first table with any key field of the second table.

The screenshot shows the Spago Studio interface. At the top, there is a menu bar with various icons. Below it, a title bar says "NON AUTENTICATO". The main area has two main sections:

- Relations List:** A table with columns: RELATION NAME, RELATED AMBIENT CODE, RELATED PHYSIC FILE, and RELATION TYPE. One row is visible with values: "orderhe", "DAOCLASSIC", "Environment Classic Models orders", and "Inner join". A red arrow points from the "RELATION NAME" column to the "FIELD" column in the "Relation Fields List" panel below.
- Relation Fields List:** A table with columns: FIELD and RELATED FIELD. One row is visible with values: "Order number" and "10".

dao_red0f

Log Field Table

Fields

Field	Type	Description	Null	Key
AMB_CDE	char(10)	Environment code	NO	PRI
FIL_PH	char(50)	Table Name	NO	PRI
REL_NAM	char(50)	Relation Name	NO	PRI
FLD_PRO	int(11)	Progressive field code	NO	PRI
FLD_PROREL	int(11)	Progressive related field code	NO	PRI

2.13 DECODING FIELDS

The decoding fields permits to add other table fields to the generated inquiry statements. Common use of this function is to take a description from an other table using the code.

dao_rfd0f

Log Field Table

Fields

Field	Type	Description	Null	Key
AMB_CDE	char(10)	Environment code	NO	PRI
FIL_PH	char(50)	Table name	NO	PRI
REL_NAM	char(50)	Relation name	NO	PRI
FLD_PRO	int(11)	Progressive field code	NO	PRI
FLD_USE	char(1)	Progressive related field code	NO	

3 Classic Models Sample Application

The Classic Models Inc. sample application is build on example database of the Eclipse BIRT (Business Intelligence Reporting Tools) project. Its main goal is to be obvious and simple, yet able to support the Spago Studio development.

The database represents a fictitious company: Classic Models Inc. which buys collectable model cars, trains, trucks, buses, trains and ships directly from manufacturers and sells them to distributors across the globe.

Classics database it was change by adding columns “expireDate” and “lastUser” on table employees to show logical delete function and audit capability of the DAO infrastructure.



3.1 DEMO PURPOSE

Spago Demo Application has many “ready to play” feature such as:

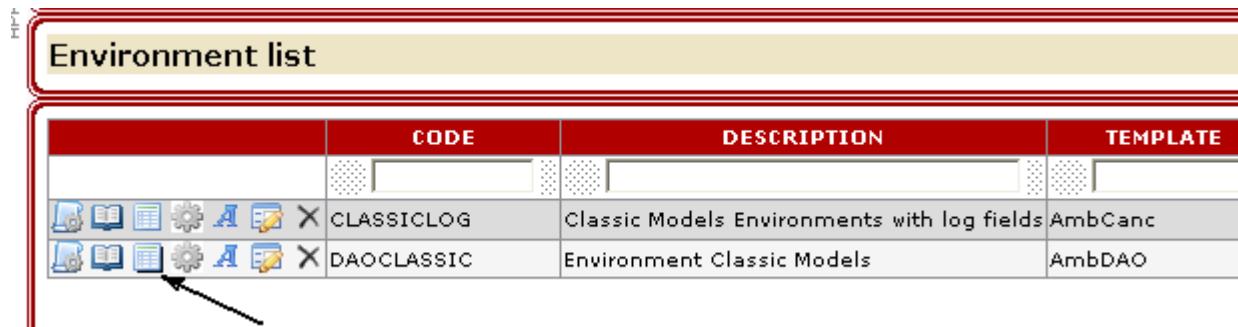
- Simple list page with sort, filter, ...
- Simple detail page with mandatory and data validation
- A lookup
- Logical delete
- An example of Master / Detail page

3.2 HOW TO BUILD “PRODUCTLINES” CRUD FUNCTION

This section shows how to use Spago Studio through the implementation of simple “ProductLines” CRUD (Create /Read/Update/Delete) function.

3.2.1 STEP 1: IMPORT TABLE DEFINITION

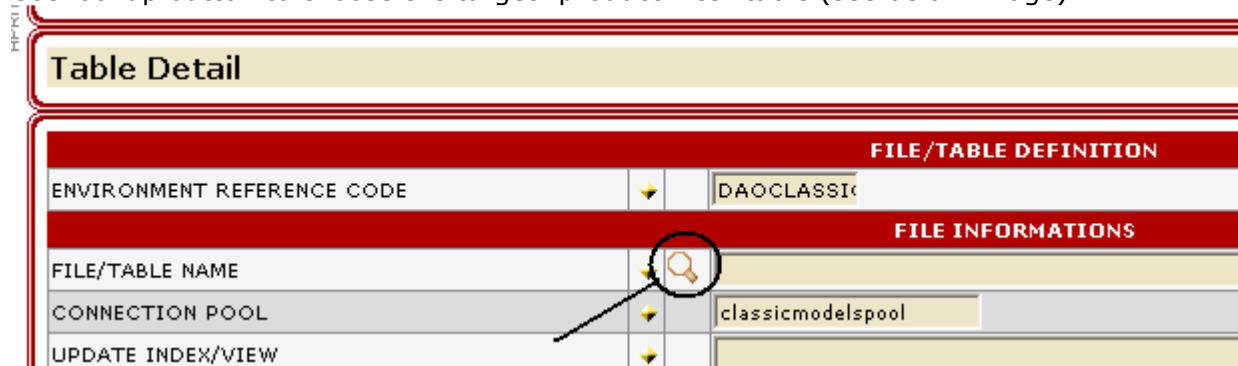
- Open DAO function from application menu
- Click on "Environment files" caption of the "DAOCLASSIC" environment (see below img)



- Add new table (see below image)



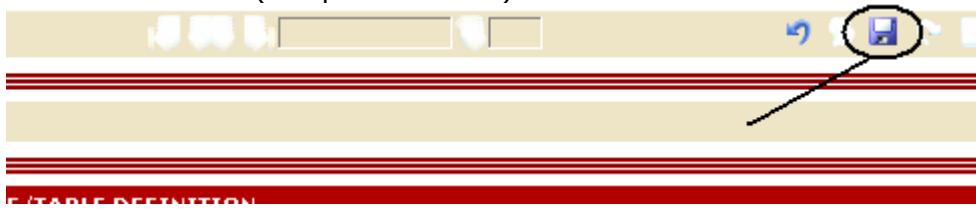
- Use lookup button to choose the target "productlines" table (see below image)



- Insert package, choose CRUD Template and, if desired, a relative path (see picture below)

FILE/TABLE DEFINITION		
ENVIRONMENT REFERENCE CODE	DAOCCLASSIC	
FILE INFORMATIONS		
FILE/TABLE NAME	productlines	
CONNECTION POOL	classicmodelspool	
UPDATE INDEX/VIEW	productlines	
READ INDEX/VIEW (VALID)	productlines	
READ INDEX/VIEW (DELETED)	productlines	
FILE DETAILS		
FIELD PREFIX		
JAVA DAO PACKAGE	it.eng.dao.classicmodels	
JAVA TEMPLATE	JAVA FILES TEMPLATE	
SQL TEMPLATE	DAO STATEMENTS WITHOUT FLAG	
CRUD TEMPLATE	TEMPLATE CRUD STANDARD	
FILE/TABLE DESCRIPTION		
XMLPATH	productlines	

- Press save button (see picture below)



3.2.2 STEP 2: GENERATE JAVA CODE AND CONFIGURATION FILES

- Use "Java Dao Class" Caption, "XML Statement" Caption and "Crud" Caption for generate the Customer CRUD function

Tables List				
	FILE/TABLE NAME	FILE/TABLE DESCRIPTION	UPDATE INDEX	DELETE INDEX
	productlines		productlines pro	

3.2.3 STEP 3: ADD NEW GENERATED FILES TO THE APPLICATION

- Modify the master.xml of classic models (home\andrea\workspace\SpagoDemoWeb\WebContent\WEB-INF\conf\classicmodels\master.xml) to add the new configuration files.

Add the below line:

```
<CONFIGURATOR path="/WEB-INF/conf/classicmodels/productlines/master_productlines.xml" />
```

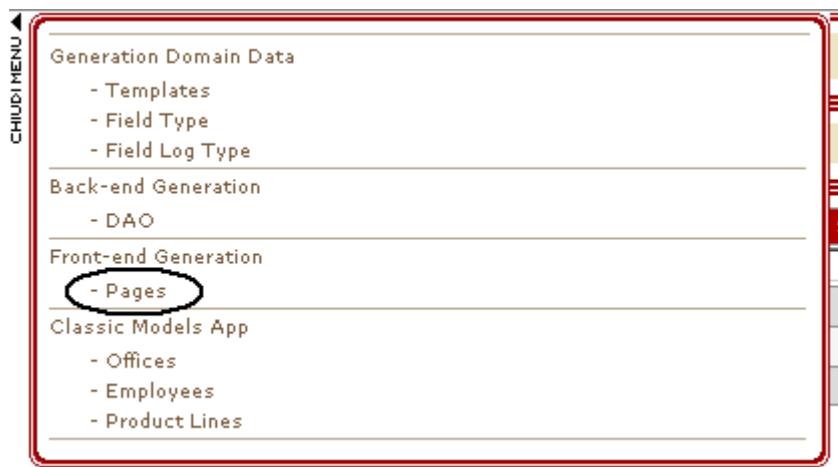
- Modify the menu.xml file for add a new link to productlines page:

```
<MENU name="Classic Models App" alt="Classic Models Test Application">
<SUBMENU name="Offices" alt="Offices" url="/SpagoDemo/servlet/AdapterHTTP?PAGE=Pagooffices" />
<SUBMENU name="Employees" alt="Employees"
url="/SpagoDemo/servlet/AdapterHTTP?PAGE=PagoEmployees" />
<SUBMENU name="Product Lines" alt="Product Lines"
url="/SpagoDemo/servlet/AdapterHTTP?PAGE=pagproductlines" />
</MENU>
```

- Refresh workspace on eclipse and restart the application
- Enjoy width your new Customer CRUD function and try it on

3.2.4 STEP 4: ADD A DECORATOR TO A COLUMN INTO THE LIST MODULE

- Open main menu and select "Pages" item



- Select modules caption on pagproductlines row

	NAME	SCOPE
X	pagEmployees	REQUEST
X	pagoffices	request
X	pagproductlines	request

- Select caption Columns Selection on modproductlinesList row

	NAME	SCOPE
	modproductlinesList	it.eng.sp
	modproductlinesDetail	it.eng.sp

- Click on Edit caption on textDescription Row

	NAME	LABEL	TYPE	COLSPAN
	productLine		FIELD	
	textDescription		FIELD	
	htmlDescription		FIELD	
	image		FIELD	

- Select CROP70 decorator from dropdown combo

MODULE	
MODULE NAME	modproductlinesList
NAME	textDescription
LABEL	
TYPE	FIELD
COLSPAN	
DECORATOR	CROP70
NOFILTER	
READ ONLY	
WIDTH	
SORT ACTION	sortByTextDescription
TEMPLATE	COLUMN TEMPLATE

- Click over the save button



- Then push the back button until you're back in modules list



- Select the Build XML caption to show the XML for the list module

	NAME	
	modproductlinesList	it.eng.spago
	modproductlinesDetail	it.eng.spago

- Edit the showed path to include productlines directory before the module name

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Creation date and time :26/11/2008 15:35:49 -->

<MODULE class="it.eng.spago.dispatching.module.list.smart.impl.DefaultSmartListModule"
name="modproductlinesList" template="xmlCrud/TemplateListModuleCRUD">
<CONFIG title="List for productlines">
<ROW_PROVIDER class="it.eng.spago.paginator.smart.impl.DBRowHandler">

```

- Push the write button close the windows, refresh your workspace and republish the application, you will see the list as the image below.

PRODUCT LINE	TEXT DESCRIPTION	HTML DESCRIPTION	IMAGE
Classic Cars	Attention car enthusiasts: Make your wildest car ownership dreams come ...		
Motorcycles	Our motorcycles are state of the art replicas of classic as well as co ...		
Planes	Unique, diecast airplane and helicopter replicas suitable for collecti ...		
Ships	The perfect holiday or anniversary gift for executives, clients, frien ...		
Trains	Model trains are a rewarding hobby for enthusiasts of all ages. Whethe ...		
Trucks and Buses	The Truck and Bus models are realistic replicas of buses and specializ ...		
Vintage Cars	Our Vintage Car models realistically portray automobiles produced from ...		

3.3 HOW TO BUILD “PRODUCTS + ORDERDETAILS” CRUD FUNCTION

This section shows generation step’s of a particularly CRUD function.
Following this short tutorial you’ll learn how to manage decoding field for the related tables.

3.3.1 STEP 1: GENERATE PRODUCTS CRUD FUNCTION

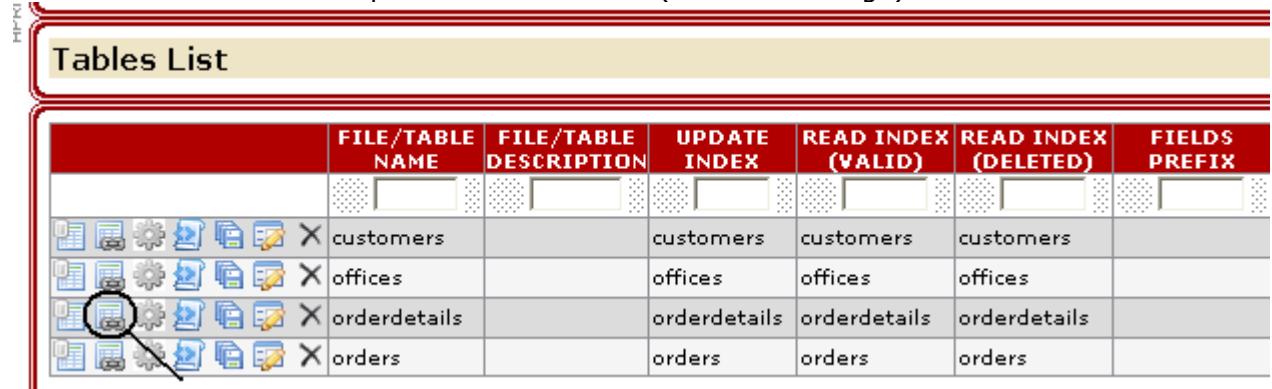
- See the previous paragraph of the ProductLines CRUD function.
NB: remember to modify master.xml and menu.xml

3.3.2 STEP 2: GENERATE ORDERDETAIL CRUD FUNCTION

- See the previous paragraph of the ProductLines CRUD function
NB: remember to modify master.xml and menu.xml

3.3.3 STEP 3: ADD RELATION FROM ORDERDETAIL TO PRODUCTS

- Press the "add relation" caption on orderdetails (see below image)

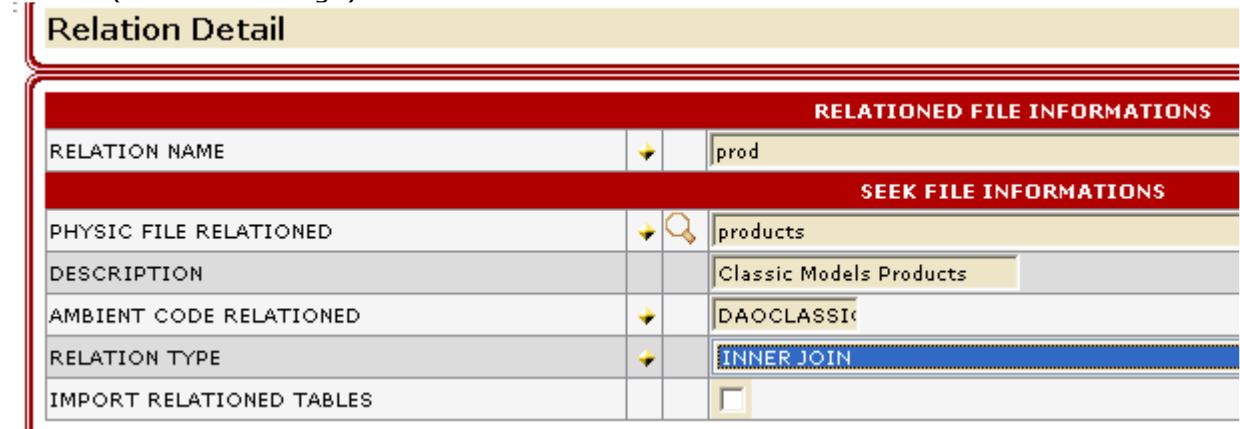


	FILE/TABLE NAME	FILE/TABLE DESCRIPTION	UPDATE INDEX	READ INDEX (VALID)	READ INDEX (DELETED)	FIELDS PREFIX
	customers		customers	customers	customers	
	offices		offices	offices	offices	
	orderdetails		orderdetails	orderdetails	orderdetails	
	orders		orders	orders	orders	

- Insert new relation using the new button (see below image)

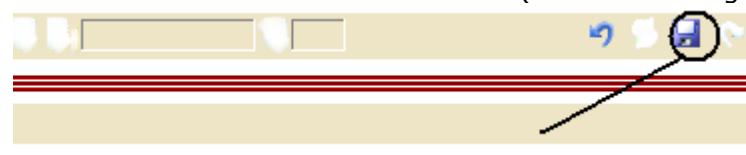


- Fill the required information and choose the "Phisic File Related" using the lookup button (see below image)

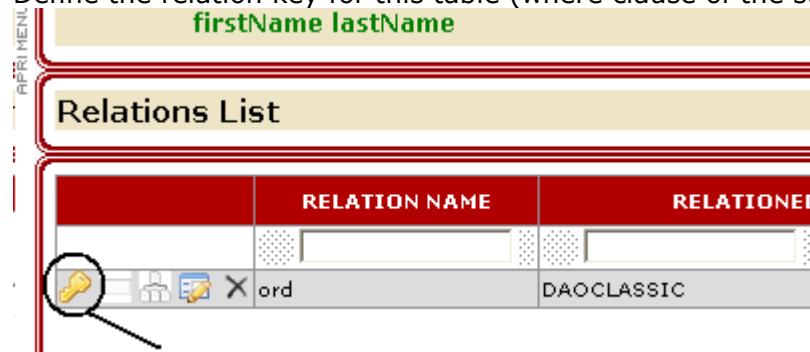


RELATED FILE INFORMATIONS	
RELATION NAME	prod
SEEK FILE INFORMATIONS	
PHYSIC FILE RELATED	products
DESCRIPTION	Classic Models Products
AMBIENT CODE RELATED	DAOCLASSIC
RELATION TYPE	INNER JOIN
IMPORT RELATED TABLES	

- Press save button for add this record (see below image)



- Define the relation key for this table (where clause of the statement)



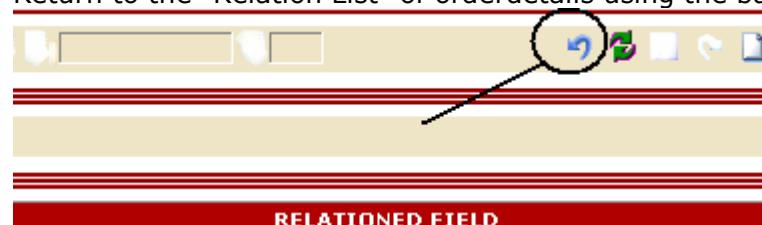
- Insert new "relation fields" using the new button (see below image)



- Set the key "Product number" of orderdetail width "Product number" of order then insert this record using "save" button (see below image)



- Return to the "Relation List" of orderdetails using the back button (see below image)



- Add the decoding field in this example status of order

Relations List

	RELATION NAME	RELATED TABLE
	prod	DAOCLASSIC

- Press the new button then insert the new record (see below image)

Decoding Fields Detail

PROGRESSIVE CODE	DECODE
20	productName
FIELD DESCRIPTION	

- Use "Java Dao Class" Caption, "XML Statement" Caption and "Crud" Caption for re-generate the Customer CRUD function (see below image)

Tables List

	FILE/TABLE NAME	FILE/TABLE DESCRIPTION	UPDA IND
	customers		custom
	offices		offices
	orderdetails		orderde
	orders		orders

- Refresh workspace on eclipse and restart the application.
You can see in the order-detail list the column product name, also in detail module (see below image)

Lista orderdetails

ORDER NUMBER	PRODUCT CODE	QUANTITY ORDERED	PRICE EACH	ORDER LINE NUMBER	PROD_PRODUCTNAME
10100	S18_1749	30	136,013	3	1917 Grand Touring Sedan
10100	S18_2248	50	55,090	2	1911 Ford Town Car
10100	S18_4409	22	75,460	4	1932 Alfa Romeo 8C2300 Spider Sport
10100	S24_3969	49	35,290	1	1936 Mercedes Benz 500k Roadster

3.4 HOW TO BUILD A LOOKUP FROM ORDER-DETAIL TO PRODUCTS

This section shows generation step's for insert a lookup to a product into the detail module of order-detail function (see below image).

HF1

Dettaglio orderdetails	
PRODUCT CODE	<input type="text"/>
PROD_PRODUCTNAME	<input type="text"/>
PRODUCT	
ORDER NUMBER	<input type="text"/>
QUANTITY ORDERED	<input type="text"/>
PRICE EACH	<input type="text"/>
ORDER LINE NUMBER	<input type="text"/>
ORDER DETAIL	

3.4.1 STEP 3: ADD LOOKUP FROM ORDETAILS (DETAIL MODULE) TO ORDER (LIST MODULE)

- Open the “Page” function from menu
- Press the “module” caption from orderdetail row (see below image)

HF1

Defined PAGES for current application		
	NAME	SO
	Pagoffices	request
	PagEmployees	REQUEST
	Pagorders	request
	Pagorderdetails	request

- Press the “field details” caption from ModcustomersDetail row (see below image)

HF1

Defined MODULES for current applica		
	NAME	
	ModcustomersList	it.eng.spago
	ModcustomersDetail	it.eng.spago

- Add new Fieldset using the new button and insert: Name, legend, choose position and width then choose the ProductsLookup (for insert a lookup for this fieldset); see below image

Defined FIELDSET for DETAIL MODULE

MODULE	
MODULE NAME	ModorderdetailsDetail
FIELDSET DEFINITION	
FIELDSET NAME	key1
FIELDSET LEGEND	ORDER
POSITION	LEFT POSITION
WIDTH	100%
LOOKUP	LOOKUPORDERS
TEMPLATE	TEMPLATE FIELDSET

- The last step, consist on move field from fieldset "key" to the new fieldset "key1". For to do that open the choosen filed into the fieldset "key" (follow the below images)

Defined FIELDSET for DETAIL MODULE

	NAME	POSITION
	key	left
	key1	LEFT

- Choose productCode and after prod_productName

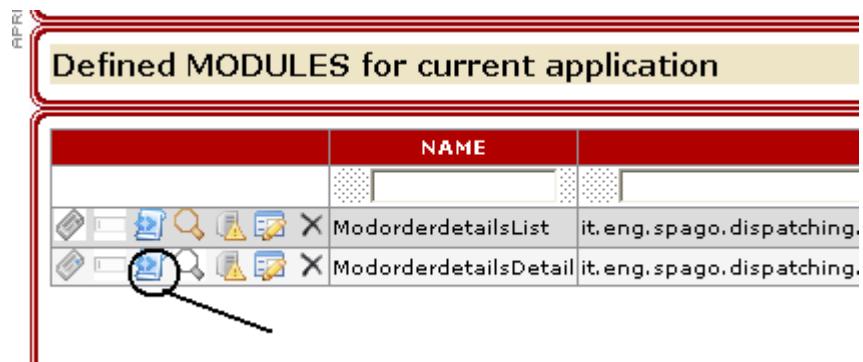
Defined FIELDS for DETAIL MODULE

ID	NAME	TEMPLATE WIDGET
1	productCode	TEXT
2	prod_productName	TEXT

Detail for FIELD modorderdetailsDetail FIELD (productCode)

FIELD IDENTIFICATION	
MODULE NAME	modorderdetailsDetail
FIELDSET NAME	key1
NEW FIELDSET	
FIELD NAME	KEY KEY1
FIELD LABEL	KEY1
FIELD TEMPLATE WIDGET	TEXT FIELDS TEMPLATE
SIZE	50
MAX LENGTH	50
JAVASCRIPT ON KEY PRESS EVENT	OnKeyPress
JAVASCRIPT ON KEY UP EVENT	OnKeyUp
DECORATOR CLASS	
LOOKUP FIELD	PRODUCTCODE

- Save modified xml for detail module using the specific caption (see below image), remember to edit the path to include “orderdetails” directory before the module name



- Refresh workspace on eclipse and restart the application
- Enjoy width your new lookup to Product list function and try it on

3.5 SOFT DELETE AND AUDITING DEMO FUNCTION

The built-in employee function show how to handle a logical delete (by setting a discharge field) and a simple auditing trace of the last user for the record modification.

This feature is implemented by the environment “CLASSICLOG”

Environment list

	CODE	DESCRIPTION	TEMPLATE
	CLASSICLOG	Classic Models Environments with log fields	AmbCanc
	DAOCLASSIC	Environment Classic Models	AmbDAO

Open log-field

You can see in this environment two different log-field (using the specific caption). The “Logic Delete Flag” is used only for the delete action; “Audit user filed” is used always (insert,update,select and delete). See below image.

Log Fields List

PROGRESSIVE CODE	DESCRIPTION	FIELD NAME	USED	FIELD TYPE
X 1	Logical Delete Flag	discharge	Delete	10 Deleted flag
X 2	Audit user field	lastUser	Allways	20 User

Also the “progressive code” specify the insert order of this field into the statement and java files.

3.6 AN EXAMPLE OF MASTER/DETAIL PAGE

The built-in employee function also shows you an example of Master/Detail page, if you select a detail caption the web page will be splitted in two half. The upper half page will show you the details data for the selected employee record, the lower half contains two tabs that will show you the customers related to the employee and his office.

You can see an example of it on the below image

HYPER MENU ▾

firstName lastName

Employee Detail

Employee	
EMPLOYEE NUMBER	1702
LAST NAME	Gerard
FIRST NAME	Martin
EXTENSION	x2312
E-MAIL	mgerard@classicmodelcars.com
JOB TITLE	Sales Rep

Chief		Offices	
REPORTS TO	1102	OFFICE CODE	3
CHIEF NAME	Gerard	RELOFFICES_CITY	NYC
CHIEF LASTNAME	Bondur		

Customers		Office Location								
Customer Number	Customer Name	Contact Last Name	Contact First Name	Phone	Address Line 1	Address Line 2	City	State	Postal Code	Country
216	Enaco Distributors	Saavedra	Eduardo	(93) 203 4555	Rambla de Cataluña, 23		Barcelona		08022	Spain
298	Vida Sport, Ltd	Holz	Mihael	0897-034555	Grenzacherweg 237		Genève		1203	Switzerland
344	CAF Imports	Fernandez	Jesus	+34 913 728 555	Merchants House	27-30 Merchant's Quay	Madrid		28023	Spain
376	Precious Collectables	Urs	Braun	0452-076555	Hauptstr. 29		Bern		3012	Switzerland
450	Cowida Auto Solutions Ltd	Commons	Martin	011 555 22 00	12 Avril 47		Madrid		28020	Spain

In these tabs you can add, edit or delete the relative data as on standard page.