Ramecera™

Screwdriver application

ReferenceManual-3.3.1

October, 2020

Index

Author: Massimo Maglioni – October, 2020

1 Preface

1.1 Graphic conventions

`markup` : the name of a markup, always followed by sign +

```
ELE line
```
: listings of ELE files

`constant width` : used for program listings as well as within paragraps to refer to program elements such as variable or method names

classification: used for listings of object types

1.2 Tags used in this document

`<nationalCode>` example: **https://www.berlin.de/** national code is **"de"**
`<myCompany>` example: **https://www.ibm.com/us-en/** my company is **"ibm"**
`<myApp>` example: **https://www.microsoft.com/de-de/windows/** my app is **"windows"**
`<rail>` the name of every rail is simply formed by the 4-character string "**`rail`**" followed by a number (start value=0) example: **`rail0, rail1`**, ecc.

2 Markup classification

- Markups starting with letter `e` are typical for Tiles.
- Markups starting with letter `h` are typical for Hibernate.
- Markups starting with letter `m` are typical of Maven.
- Markups starting with letter `u` are typical of Spring Security.
- Markups starting with letter `f` relate to a JSP form-type.
- Markups starting with letter `w` relate to Web-flow.
- Markups starting with letter `b` relate to menus.
- Markups starting with letter `q` relate to Quartz Scheduler.
- Markups starting with letter `s` relate to Spring.
- Markups starting with letter `x` concern the Authentication Provider, therefore assume the Spring Security environment; markups starting with letters `xl` concern, in addition to the Authentication Provider, the LDAP system.
- Markups starting with letter `o` configure graphically the **`index.jsp`** page, that is, the initial JSP.

When there is a `v` as the second letter, the markup indicates the software version, in one or more qualifiers.

This scheme has various exceptions, because certain tags have names chosen only because they are easier to remember.

2.1 The file MarkupsList-n.n.n.pdf

In this file there is a list of markups, each with its particular configuration; the configurations are reported in the Java source to the class it/ramecera/screwdriver/Hammer-.java.
This list, and consequently Hammer class, has to be modified only if the developer wants implement a better version of Screwdriver.
The 8 properties of each markup are: ***name***, ***type***, ***frequency***, ***imply***, ***mandatory wagons***, ***optional wagons***, ***domain***, ***previous***.

type and ***frequency*** are 2 numeric codes; the possible values are descripted at the end of the markups list.
imply: in this column, separated by commas, we find the markups which, as a consequence, must be present; and those which (preceded by character **!**) must not be present; if the markup indicated as "imply" begins with the +, then it indicates what is the "locomotor" in its own line; therefore it says that we necessarily deal with a "wagon" markup. After indicating the required markup, a value may be present: it means that it implies the existence of that markup followed by that value (example: **`xusers+jdbc`**). The presupposed markups are all locomotors, in the same line or in other position of the ELE file.

2.2 The multipurpose markups

`primary` - it captures an include file (i.e. a file in the **`include`** directory, extension **`.eli`**; see User Guide) containing the main appropriate markups for that application; example: **`primary+JBoss EAP 7.1_s4_h5_jpa2`** calls the file **`box/include/primary_JBoss EAP 7.1_uv_s4_h5_jpa2.eli`** .

`appse` - name and version of the application server or servlet container; usually part of the include `primary`.

`jeev` - Java Enterprise Edition version; usually part of the include `primary`.

`depen` - group of pre-established dependencies, to be declared in **`pom.xml`**; for now it can be depen+0, depen+1, depen+2 or depen+3.

`pompr` - properties declared, via the <properties> tag, inside pom.xml; they are added to the existing ones (for beginners: the <properties> tag is used to avoid having to repeat a version declaration several times within the POM, and to change it acting directly on the property).

`company` - the company's internet domain; ex .: it.ramecera.

`filte` - present if web.xml filters are to be declared.

`antisamy` - present if the Antisamy framework is to be used; dragging from the **`include/java`** directory, add the classes

- **`CrossSiteScriptingFilter.java`**
- **`JsessionIdRemoveFilter.java`**
- **`HttpRequestWrapper.java`**

- **AntiSamyListener.java**
- **XssSanitizer.java**
- **policy.xml**

in various directories; adds a listener in **web.xml**.

log4j - Apache version log4j; it inserts dependencies in pom.xml

captc - presence of a captcha; create a new servlet in **web.xml**, and go to form the dependencies in pom.xml

language - 2-character abbreviation for the language; it is inserted in i18n properties for Spring.

jsp - path and name of the JSP, the path starts from **webapp/WEB-INF/views**; only one intermediate directory is allowed: for example **webapp/WEB-INF/views/tiles/normal.jsp**
will be indicated with
jsp+tiles/normal.jsp
the JSP name must begin with a lowercase letter, if uppercase it is automatically converted; even if they are in different directories, for an easier search in the file system, the overall set of JSPs must not have duplicates; see User Guide, "Processing JSPs".

i18n - descriptions internationalization functionality; the default language is the one indicated by the defaulan+ markup.

webmime - presence of mime-mapping parameters in **web.xml**

weberror - presence of error-page parameters in **web.xml**

charset - Unicode character set; the default is UTF-8

indexjsp - presence of a home page with name index.jsp; the class **<nationalCode>/<myCompany>/<myapp>/web/controller/DoorControllerBelong**
is automatically created.

matrix - presence of matrix variables

pretprin - presence of the prettyPrint option:
class = "org.springframework.web.servlet.view.json.MappingJackson2JsonView">
<property name = "prettyPrint" value = "true" />

pragma - if I want to check the cache in the previous JSP (corresponds to the immediately preceding JSP markup)

requmap - indication of the request mapping, that is the final part of the URL, which generates a method which in turn produces the name of the JSP to be called; the methods created are placed in classes of type @Controller, which consequently go to the path **<nationalCode>/<myCompany>/<myapp>/web/controller**

colorform - color of the area around the login button

`userpassform` - userid and password (separated by commas) in the trivial case of userid and password carved in the code

2.3 Markups generating a menu (type b)

`bmenu` - wagon of `jsp`, indicates the presence of a menu at the head of the JSP

`bdefmenu` - defines a menu

`blabel` - defines each of the menu items (or buttons)

`bfixedtop` - wagon by `bdefmenu`, if present, add the navbar-fixed-top clause to the menu

`burl` – wagon of `blabel`, the URL of the menu item

2.4 Markups connected to Spring (type s)

`sv` - Spring version, no markup if spring is not used; it inserts dependencies in pom.xml

`sreso` - in **`DispatcherServlet.xml`**, it writes one or more **`<mvc:resources`** tag.
If it has no argument, it is a simple resource, of the type **`/resources/xxxxyyyy/**`**; if it has an argument, we have a tag for each `sreso` markup; its mapping is the location + 2 asterisks; the starting position is **`src/main/webapp`**

`srobo` - in the DispatcherServlet: presence of the typical Spring robots.txt file

`sieco` - in **`mvc:view-controller`**, it represents both the path and the view-name

`sdefauto` - default-autowire

`sinterce` - name of the class subjected to Spring interceptor; they can be none, one or more than one; each markup, if `i18n+` is also present, creates a call in **`Dispatch-erServlet.xml`** to **`org.springframework.web.servlet.i18n.LocaleChangeInterceptor`**

2.5 Markups of the Maven group (type m)

`mavmo` - version of the Maven model; it is present in the first tags of the pom.xml file

`mavgr` - group id of Maven; it is present in the first tags of the pom.xml file

`mavco` - version of java for the Maven compiler; it is present in the properties of the pom.xml file

`mv` - version of Maven; it is present in the first tags of the pom.xml file

`mavbu` - boolean, indicates if I want the <build> tag, obviously with all its contents, in the pom.xml; if I want it, I must also specify the 3 markups `mavcl`, `mavcp`, `mavwa`

`mavcl` - version of the Maven clean plugin

`mavcp` - version of the Maven compliler plugin; not to be confused with the `mavco` markup, which indicates the version of java

`mavwa` - version of the Maven war plugin

2.6 Markups of the Web Flow group (type w)

`wv` - version of the web flow framework; it inserts dependencies in pom.xml

`wparams` - parameters of the web flow framework; I can have more than one flow in the same application, although it becomes very complex

2.7 Markups of the Tiles group (type e)

`ev` - version of the Tiles framework; it inserts dependencies in pom.xml

`edefn` - name of the tiles definition: **`<definition name=`** in **`tiles.xml`**

2.8 Markups of the Spring Security group (type u)

`uv` - security version, absent if spring is not used; add a listener in `web.xml`; it inserts dependencies in **`pom.xml`**

`userpass` - if you want an independent JSP with userid and password, and all subsequent processing; possible values are ***simple***, ***domain*** and ***trivial***; ***simple*** when I access comparing userid and password on 2 ordinary columns of a DB table (markup `utabuser`); ***trivial*** when the userid and password are specified and readable only in the XML file that manages the Authentication Manager;

`utabuser` - DB schema and name of the DB table containing the user's profile; ex .: copper.actors; the userid coincides with the key of this table; the next `uwelcome` markup indicates the table column that must be displayed inside the welcome message; the next `upassword` markup indicates the table column that contains the password

`udomainfil` - if in authentication I want and must specify a network domain, therefore to be treated separately

`usignup` - if I want to build the registration functionality of a new user, integrated into the application; the menu item is simply added, but then the JSP must be added with all its management.

`uintp` - name of the internal properties file, specific for spring security, in location **`/src/main/resources`**.

2.9 Authentication Provider markups (type x)

`xprovider` - Authentication Provider methodology, element of Spring Security; possible values: user-service, class, jdbc, ldap.

`xgbac` - query in JdbcUserDetailManager to configure user groups, implies xprovider (GBAC = Group Based Access Control).

`xusers` - query in JdbcUserDetailManager to configure individual users, implies xprovider+jdbc.

`xauth` - query in JdbcUserDetailManager to configure individual permissions, implies `xprovider+jdbc`.

`xencoder` - type of hash + salt of the password, to be set on the DB and to be calculated on the password provided before the comparison; for now the accepted values are useridMD5 if **`xprovider+class`** and saltedSHA256 if **`xprovider+jdbc`**.

2.9.1 Markups for LDAP

`xlbase` - the user group is specified, through group-search-base.

`xlclass` - specify the class of users.

`xlpass` - specify the password treatment; can have password-compare, hash_ssha; if it has any other value, it is intended to be the password-attribute expected as the password.

`xlservice` - you want to use LdapUserDetailsService.

2.10 The markups of the Hibernate group (type h)

`hv` - version of the Hibernate framework; only the first 2 qualifiers of the version are indicated, for example 3.2; it inserts dependencies in `pom.xml`.

`hvalid` - version of the validation according to Hibernate; the presence of the `hv` markup is not necessary, because it also works autonomously; it inserts dependencies in **`pom.xml`**.

`hdbms` - the name of the Database Management System; it is used to establish the DBMS dialect registered in Hibernate.

`hdbschema` - the DB schema of the tables used; it is present as many times as there are schema DBs; helps to form **`web.xml`**, plus the 2 files <rail>-orm.xml and <rail>-service.xml name for each DB schema; if it isn't already, it is automatically converted to lowercase. It can drive the optional `hjarv` markup, which serves to indicate that

the corresponding JPA project is also of the Maven type, and therefore produces a jar with a version number.

`hjarv` - if present, indicates the version number of the jar produced by the JPA project corresponding to the schema DB, therefore also indicates that this is of the Maven type; if absent, indicates that the JPA project is not Maven.

`hjndi` - the name of the JNDI, i.e. the data source declared in the configuration of the application server; present as many times as there are DB schemas, is coupled to the `hdbschema` markup.

`hdefrail` - defines the «rail»; the rail is the DB-schema coupled to the JNDI; it is so defined because I may need to access different DB schemas within the same JNDI. Example:
`hdefrail+0+hjndi+metalDS+hdbschema+copper`
`hdefrail+1+hjndi+woodDS+hdbschema+mahogany`

2.11 Markups of data validation (type v)

`vallabel` - message label, being defined

2.12 Markups of data validation (type f)

`formt`, `forma`, `formn`, `formc`, `formr` – the name of the control element, if the previous JSP (corresponds to the immediately preceding JSP markup) contains a form; using the word "form" we mean the presence of at least 2 input objects (i.e. graphic controls), which can be text-box (`formt`), text-area (`forma`), list-box (`formn`), check-box (`formc`), radio button (`formr`); the control name must begin with a lowercase letter, if it is uppercase it is automatically converted; `formn` is always linked to a DB table

`flabel` the label of a graphic control in a form

`fstart` initial checked option in a radio buttons group

`ftokens` the list of tokens to fill a dropdown graphic control, or the list of options in a radio button group

2.13 Markups of the Quartz group (type q)

If Quartz is present, the **`<nationalCode>/<myCompany>/<myApp>/task`** directory is automatically created. Only quartz 1 is managed.

Examples of using q-type markup:

```
quartz+1.8.6
qjobname+wood+qcron+ 0 0 *? * MON-SAT+hrail+0
```

or:

```
quartz+ 1.8.6
qjobname+wood+qsimple+4000,20000+hrail+0
```

`quartz` indicates the release;

`qjobname` indicates the job name as it is identified within Quartz;

`qsimple` and `qcron` are an alternative to the other, as foreseen by Quartz, and indicate the corresponding numerical formulas (consult the Quartz manual);

`hrail` indicates the number of rails; this is a progressive number, which refers to the definition of rail made with `hdefrail`.

For the beginners: in the automatically defined properties, the presence of
`<prop key = "org.quartz.scheduler.idleWaitTime">90000</prop>`
is not trivial, which aims to fine tune the timing of the burns.

2.14 Special markups

The markup **`file+fileName`** indicates that you want to create a file with fileName name; the content consists of the lines immediately following this markup, until a line starting with a valid markup is encountered; the file is initially recorded only in Java heap memory, and will be placed in a certain position as soon as another markup will recall it, establishing in which path it should be recorded. It will be called with the string indicated as fileName.
It is advisable to put these files at the end of the ele file, for readability reasons.

2.15 Combinations of markups

The only cases where the order of markups in the ELE file is essential are:

- `bdefmenu` must be followed by a set of di `blabel`: indeed a menu is composed of a set of clickable elements;
- `jsp` may be followed by a set of di `form*`: the markups `formt`, `formc`, `forma`, `formn`, `formr` declare that JSP is a form to fill out.

2.15.1 Markups being a definition, anyway locomotors

markup	action	called by	preceded by
bdefmenu	define a menu	bmenu	
blabel	define an item of a menu		bdefmenu
odefcss	define a Cascading Style Sheets	cssclass	
jsp	define a Java Server Page	goal	

3 The graphics

3.1 The CSS and the “o” markups

With the `odefcss` markup it is possible to define a CSS class that will be added in the specific **`src/main/webapp/resources/screwdriver.css`** file. The `defcss` markup is a locomotor, compulsorily followed by the following wagons:
`obackcolor` - background and border color
`ocolor` - text color
`omargin` - margin-top
`opadding` - padding-top, padding-bottom, padding-left and padding-right
`ofontsize` - font size of the text
`oalign` - the alignment of the text
`ooptbackcol1` e `ooptbackcol2` – the 2 colors of the main menu gradient (see the paragraph below “The menu”)
`ooptioncolor` – color of the menu “button” text
`ooptionhover` – color of the “button” when your mouse hovers it

A CSS class is called from a text present in a JSP with the `cssclass` markup.

4 Error handling

4.1 The Notice.java bean

This bean serves as a support for JSPs that communicate errors, warnings or simple messages.
The properties are 2, both of type String: screwFormattedText and screwFreeText. The first one is intended for the <c:out JSTL tag; the second one, typically inside a <p>, to be used as plain HTML text.

4.2 The JSP error.jsp

It is a simple JSP, with a title in the center: "Error"; and 2 references to the 2 properties of the **`Notice.java`** bean.

5 The “silver” application

Silver is a sample web application.

First of all, we clarify that everything shown here is obtained without adding other java code to what is obtained from the ele filefirst of all, we clarify that everything shown here is obtained without adding other java code to what is obtained from the ELE file.

It starts with index.jsp.

indexjsp++bmenu+tray

(we find the sign + two times bedause indexjsp+ has non argument)

5.1 The menu

The presence of markup `indexjsp+` indicates that the application starts with index.jsp and its menu is "tray". Indeed, in class *DoorControllerBelong* we find:

```
@RequestMapping("/")
public String welcome() {
    return "home/index";
}
```

With the train:

indexjsp++bmenu+tray+ooptbackcol1+orange+ooptbackcol2+

brown+ooptioncolor+ black+ooptionhover+white

we indicate that index.jsp has a menu named "tray"; menu background is composed by 2 gradient colors: `ooptbackcol1` e `ooptbackcol2`; the label of each choice has the `ooptioncolor` color and, when the mouse hovers it, the `ooptionhover` color. Indeed, the orange, brown, black and white colors modify the bootstrap.css file v. 2.0.1 (http://www.apache.org/licenses/LICENSE-2.0.txt), with these results:


```
.navbar .brand {
    float: left;
    display: block;
    padding: 8px 20px 12px;
    margin-left: -100px;
    font-size: 24px;
    font-weight: 200;
    line-height: 1;
    color: black;
}
```

gradient of the menu background color:

```
.btn-navbar {
    display: none;
```

```
    float: right;
    padding: 7px 10px;
    margin-left: 5px;
    margin-right: 5px;
    background-color: #2c2c2c;
    background-image: -moz-linear-gradient(top, orange, brown);
    background-image: -ms-linear-gradient(top, orange, brown);
    background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#333333),
        to(#222222));
    background-image: -webkit-linear-gradient(top, orange, brown);
    background-image: -o-linear-gradient(top, orange, brown);
    background-image: linear-gradient(top, orange, brown);
    background-repeat: repeat-x;
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr
='#333333',endColorstr='#222222', GradientType=0);
    border-color: #222222 #222222 #000000;
    border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
    filter: progid:DXImageTransform.Microsoft.gradient(enabled= false);
    -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.1), 0 1px 0
        rgba(255, 255, 255, 0.075);
    -moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.1), 0 1px 0
        rgba(255, 255, 255, 0.075);
    box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.1), 0 1px 0
        rgba(255, 255, 255, 0.075);
}
```

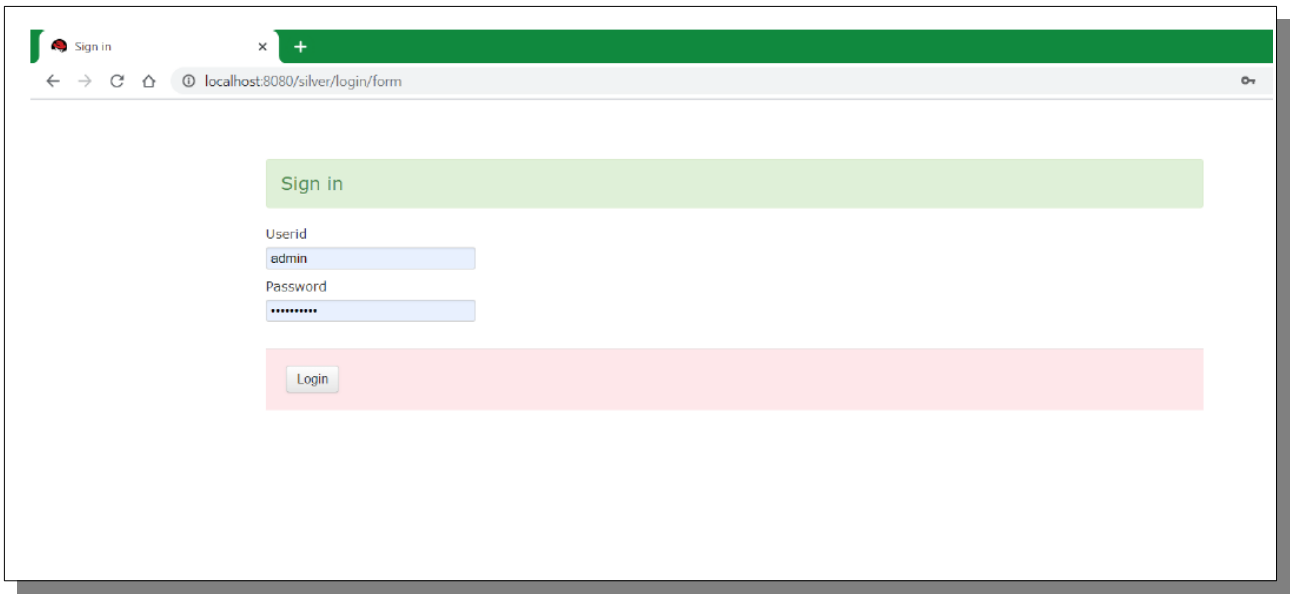
color of the other buttons:

```
.navbar .nav>li>* {
    float: none;
    padding: 13px 10px 11px;
    line-height: 19px;
    color: black;
    text-decoration: none;
    text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
}
```

color when the mouse hovers the buttons:

```
.navbar .brand:hover {
    text-decoration: none;
    color: white;
}
```

At the far right of the menu we find the Login element, which leads to a login screen. It was not generated by `blabel`, but by the presence of the `userpass` markup. Clicking this item opens a login screen.



5.2 The index JSP

With the train:

text+MY SILVER+cssclass+prince

I declare the text which has to appear in **index.jsp**, and with

defcss+prince+obackcolor+azure+ocolor+darkgreen+opadding+12+ofontsize+90+omargin+120+oalign+center

I establish the characteristics of this text (see paragraph «The CSS and the "o" markups»); with these results:

```
<html lang="en">
    <c:set var="pageTitle" value="index" scope="request" />
    <jsp:include page="../includes/header.jsp" />
    <jsp:include page="../includes/navigation_tray.jsp" />
    <!-- screwdriver_upper -->
    <!-- screwdriver_lower -->
<div class="prince">MY SILVER</div>
<jsp:include page="../includes/footer.jsp" />
</html>
```

Between `<!-- screwdriver_upper -->` and `<!-- screwdriver_lower -->` is possible insert any code.

5.2.1 A moment of discussion

"Not at all! Such a simple index.jsp is ridiculous! In reality, we have pages 1000 times more complex and full of javascript functions."

"Okay, it's very simple. But Screwdriver was not born to compose HTML code; instead Screwdriver puts the various frameworks together without the developer having to sweat to mix them together: do you know when you have to close a trunk of a car, full of luggage, you slam and slam, but it never closes? Screwdriver closes it. MVC promits us: when you sure that the application works, you can change the appearance of the JSPs without changing your business logic."

"Well, what if I have to put a new link on the home page?"

"Add a `burl+`, followed by a path, which corresponds to a `requmap+` that you put somewhere, and you're done. You will find the new link in the top menu, already operational; later your web designer will move it, make it become an image or something else, it doesn't matter."

5.3 Spring security

The ELE lines:

```
uv+3.1.4.RELEASE
userpass+trivial
xprovider+class
uspaformuser+ramecera,oxford
uspaformadmin+admin,cambridge
requmap+superior/singola+goal+master/singola.jsp+permit+admin
```

generates the file **`silver-security.xml`**:

```
    <s:http pattern="/resources/**" security="none"/>
    <s:http auto-config="true" use-expressions="true">
        <s:intercept-url pattern="/" access="permitAll"/>
        <s:intercept-url pattern="/login/*" access="permitAll"/>
        <s:intercept-url pattern="/logout" access="permitAll"/>
        <s:intercept-url pattern="/answers/**" access="permitAll"/>
        <s:intercept-url pattern="/superior/au_single" access="hasRole('ROLE_AD-
MIN')"/>
        <s:intercept-url pattern="/superior/screwau_single*" ac-
cess="hasRole('ROLE_ADMIN')"/>
        <s:intercept-url pattern="/superior/ac_single" access="hasRole('ROLE_AD-
MIN')"/>
        <s:intercept-url pattern="/superior/screwac_single*" ac-
cess="hasRole('ROLE_ADMIN')"/>
        <s:intercept-url pattern="/superior/no_single" access="hasRole('ROLE_AD-
MIN')"/>
        <s:intercept-url pattern="/superior/screwno_single*" ac-
cess="hasRole('ROLE_ADMIN')"/>
        <s:intercept-url pattern="/whole/list" access="hasRole('ROLE_ADMIN')"/>
        <s:intercept-url pattern="/**" access="permitAll"/>
        <s:access-denied-handler error-page="/answers/errorPage"/>
        <s:form-login login-page="/login/form"
            login-processing-url="/login"
            username-parameter="username"
```

```
            password-parameter="password"
            authentication-failure-url="/login/form?error"
            default-target-url="/default"
            always-use-default-target="false" />
        <s:logout logout-url="/logout"
            logout-success-url="/login/form?logout"/>
    </s:http>
    <s:authentication-manager>
        <s:authentication-provider>
            <s:user-service>
                <s:user name="ramecera" password="oxford" authorities="ROLE_USER" />
                <s:user name="admin" password="cambridge" authorities="ROLE_ADMIN" />
            </s:user-service>
        </s:authentication-provider>
    </s:authentication-manager>
```

The choice to establish passwords by writing them in the XML file is silly (**userpass+trivial**), but here it is adopted because silver is only a simple sample application. Screwdriver is able to manage much more complex accounts.

5.4 The first markup: primary

The first line of file ELE is:

```
primary+JBoss EAP 7.1_uv_s4_h5_jpa2
```

It calls the file **JBoss EAP 7.1_uv_s4_h5_jpa2.eli**, containing:

```
appse+JBoss EAP 7.1
mavmo+4.0.0
mavgr+it.ramecera.ict
mavbu+
mavcp+2.3.2
mavwa+2.1.1
mavcl+2.3
mv+0.0.1-SNAPSHOT
depen+4
pompr+0
mavco+1.8
jeev+6.0
sv+4.3.12.RELEASE
log4j+2.5
hv+5.1.0.Final+hjpav+2
uv+3.1.4.RELEASE
sreso+
```

defaulan+en

charset+UTF-8

These markups establish the main parameters of the application; the name of **eli** file was freely chosen to summarize these parameters (no markup refers to JBoss, ok); in particular, the markups mavmo, mavgr, appli (in **silver.ele**), mv produce the first part of the **pom.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>it.ramecera.ict</groupId>
    <artifactId>silver</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>silver</name>
```

while the markups mavbu, mavcp, mavwa, mavcl produce:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.3.2</version>
    <configuration>
        <source>${pom.java.version}</source>
        <target>${pom.java.version}</target>
    </configuration>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-war-plugin</artifactId>
    <version>2.1.1</version>
    <configuration>
        <packagingExcludes>.svn/,**/.svn/</packagingExcludes>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </configuration>
</plugin>
<plugin>
    <artifactId>maven-clean-plugin</artifactId>
    <version>2.3</version>
    <configuration>
        <filesets>
            <fileset>
                <directory>${basedir}/src/main/webapp/WEB-INF/lib</directory>
                <includes>
                    <include>*.jar</include>
                </includes>
            </fileset>
            <fileset>
        <directory>${basedir}/src/main/webapp/WEB-INF/classes</directory>
                <includes>
                    <include>*.properties</include>
                    <include>**/*.class</include>
                    <include>*.xml</include>
```

```
                </includes>
            </fileset>
        </filesets>
    </configuration>
</plugin>
```

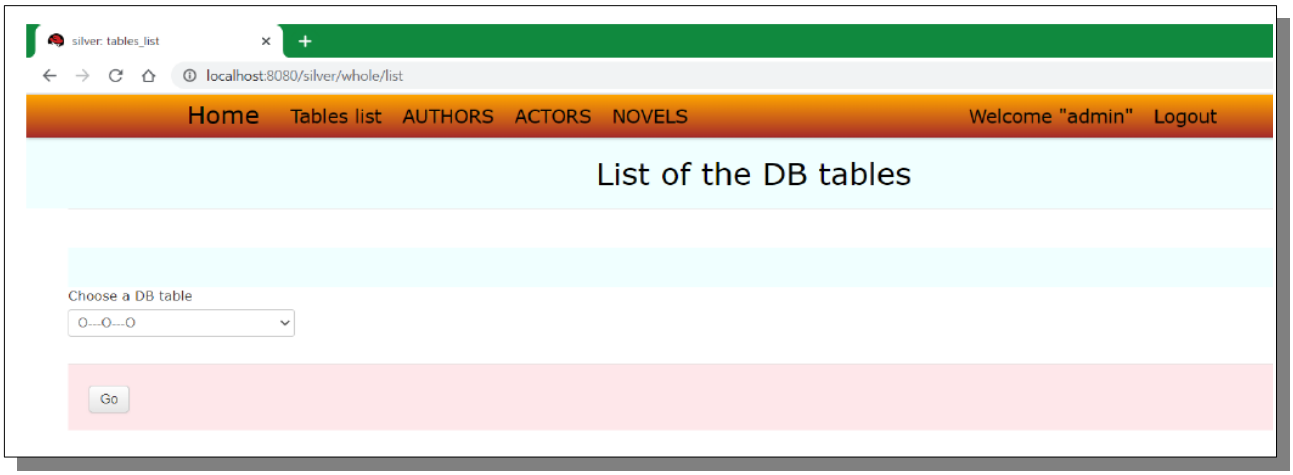
`mavbu` procuces tag <build> and all its content, `mavco` produces the file **`pom.properties`**.

5.5 How does "silver" work?

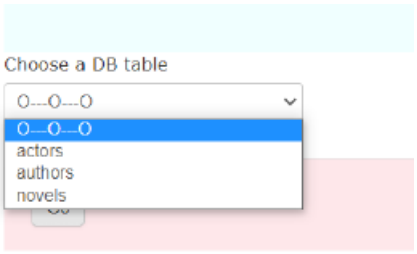
In the home page, click "Contribute":

`blabel+Contribute+burl+/whole/list`

therefore you are led to superior/list, i.e. to master/tables_list.jsp;



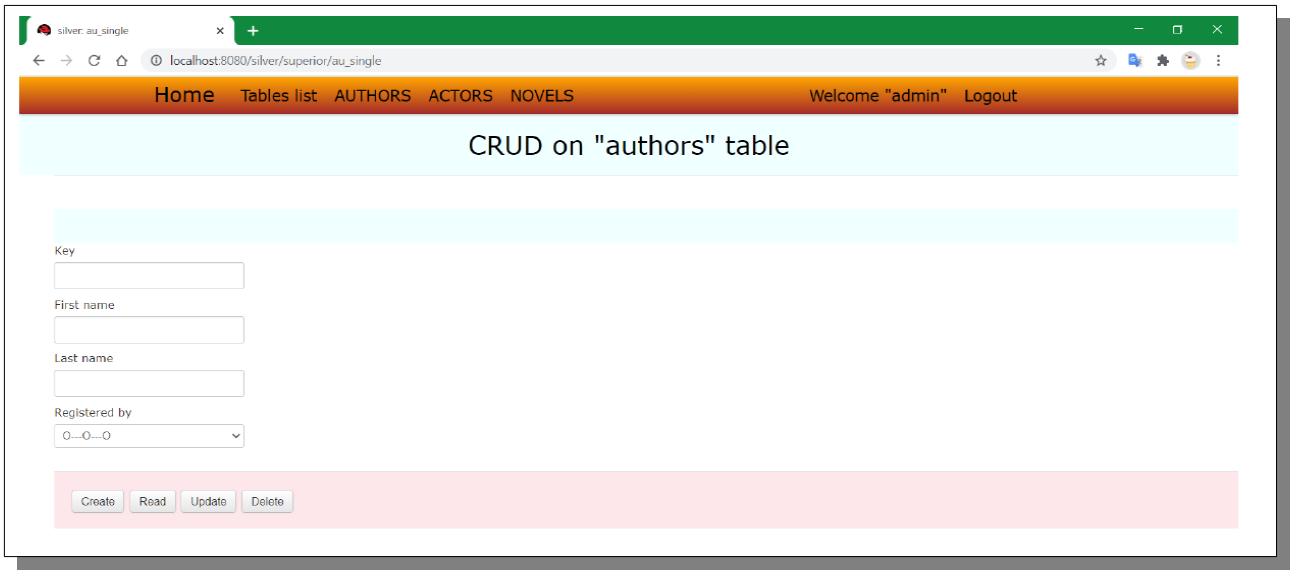
Please note the lisrìt box is already full, as directed by markup `ftokens`.



click a table, for example AUTHORS;
therefore you are led to superior/au_single, i.e. to master/au_single.jsp

requmap+superior/au_single+goal+master/au_single.jsp+permit+admin

permit+admin means you have to provide the adminstration password: really do it. we are led to this page:



Silver application works really if I link to it a DB. The DBMS used for my tests is postgres:

hdefrail+0+hjndi+waxds+hdbschema+copper+hdbms+postgres

The DDL to create the 3 tables are in eclipse project *waxds-copper*, the file is **dropCreateAll.sql**.

if I have not yet installed Postgres, it is sufficient to have the JPA *waxds-copper* project to start the Screwdriver application and create *silver*. The JPA project *waxds-copper* is installed simultaneously to Screwdriver.