

SCFramework

QUICK START

Documento per:	Uso Interno
Documento:	SCFQuickStart.doc
Data:	06 giu. 04
Autori:	Michele Barbagli

INDICE

SCFRAMEWORK	1
1. AMBITO	3
2. STRUMENTI.....	4
3. ANALISI DEI PACKAGE	7
Package it.sintraconsulting.common.taglib	7
UML Class Diagram.....	7
4. STRUTTURA DI UNA APPLICAZIONE	8
5. SVILUPPO DI UNA APPLICAZIONE	10
Step principali per lo sviluppo di una nuova applicazione	10
Sviluppo di una applicazione di gestione contatti (Rubrica)	12
APPENDICE A : DIAGRAMMA UML DELLA SCQUERYACTION (ELENCHI)25	

1. AMBITO

La complessità e difficoltà di gestione delle applicazioni Java web based, porta alla necessità di individuare una serie di strumenti che facilitino il compito dei programmatori; esistono numerosi tool nell' ambito della comunità open-source ma, in quasi tutti i casi, il costo cognitivo è particolarmente alto. In particolare, è molto complicato valutare la reale validità di tali strumenti ed integrarli insieme all' interno di una applicazione.

Si è resa pertanto necessaria il progetto di un framework di integrazione, che fornisca un pattern unico per la realizzazione di applicazioni java web-based ,che siano in qualche modo standardizzate. Prendendo spunto dai software in commercio, il framework ha come obiettivo la semplificazione di alcune delle operazioni fondamentali nella creazione di una applicazione:

- Gestione del login / logout
- Gestione dei Menu
- Gestione delle stringhe e delle etichette di testo in file di risorse esterni alla applicazione
- Creazione di elenchi paginati con funzioni di ricerca, ordinamento, filtro, cancellazione
- Creazione di form di inserimento / aggiornamento dati; per la gestione dei dati è stato integrato il Layer di persistenza Hibernate (www.hibernate.org)
- Gestione unificata degli errori

Il framework inoltre, contiene anche una serie di classi di utilità che permettono di eseguire procedure comuni:

- Convertire un file Excel in XML
- Convertire un ramo del file system, una JavaBean, una lista di JavaBean in XML
- Inviare una e-mail in formato txt o html
- Inviare un Fax attraverso web service di Interfax (www.interfax.net)
- Controllare la correttezza di una email (Interrogando il Server SMTP)
- Funzioni di manipolazione delle stringhe
- Inviare un SMS attraverso il servizio SMS-Web di Moby (www.mobyt.it)
- Interfacciamento con il Job scheduler Quartz (quartz.sourceforge.org)
- Interfacciamento con la SessionFactory di Hibernate (www.hibernate.org)

2. STRUMENTI

Gli strumenti utilizzati, sono stati selezionati tra le vari librerie disponibili gratuitamente sulla rete; è stato eseguito un lavoro di integrazione ed ottimizzazione delle librerie selezionate. Le librerie utilizzate, da intendersi anche come prerequisito per la piena comprensione del framework, sono:

- Struts1.1 (<http://jakarta.apache.org/struts>)
 - La libreria principale su cui si basa il framework
- Hibernate (<http://www.hibernate.org>)
 - Strumento di object to relational mapping; permette di mappare le tabelle del database su oggetti Java (Per esempio: tabella contatti ed oggetto Contatto con un rapporto 1-1 tra le proprietà dell'oggetto e le colonne della tabella)
- Dom4j (www.dom4j.org)
 - Libreria di manipolazione e parsing delle stringhe XML
- Log4j (<http://jakarta.apache.org/log4j>)
 - Libreria di gestione delle funzionalità di logging all'interno di una applicazione; permette la configurazione esterna del logging attraverso un file di risorse o XML
- JavaMail (<http://java.sun.com/javamail>)
 - Libreria Java per la gestione della Posta Elettronica (Invio, Ricezione, Allegati, etc.)
- XTags (<http://jakarta.apache.org/taglibs>)
 - Tag da inserire all'interno delle pagine JSP (Custom Tag) per la manipolazione die file e stringhe XML.

Gli strumenti SW utilizzati sono:

- PostgreSQL (www.postgresql.org) e driver jdbc (<http://jdbc.postgresql.org>)
- Tomcat (<http://jakarta.apache.org/tomcat>)

Partendo dal modello MVC (Model View Control), vediamo quali sono gli strumenti utilizzati e le integrazioni eseguite nei tre livelli MVC:

- Model:

- Per la persistenza dei dati, è stato utilizzato il framework Hibernate (www.hibernate.org); per il salvataggio dei dati di un form su una tabella singola, sono state sviluppate una classe Form (SCObjectForm) ed una Action (SCObjectAction) che permettono di associare un oggetto persistente (un Java Bean che ha gli stessi campi della tabella associata) ad un oggetto form e di salvare / modificare / cancellare l'oggetto e quindi i record della tabella associata; la lettura e gli elenchi vengono gestiti tramite query sql dirette al database e la classe che esegue tali operazioni è la SCQueryAction;
- Control
 - Per la logica di controllo del flusso, è stato utilizzato Jakarta Struts, il noto framework di Jakarta e ora di Apache per il controllo del flusso di esecuzione di una applicazione web; <http://jakarta.apache.org/struts/> ; la versione utilizzata, è la 1.1 che permette anche di associare dei validatori ai form direttamente all'interno di files xml di configurazione;
 - E' stata implementata la SCQueryAction per la gestione degli eventi e la SCFilterForm (la form associata) per le ricerche ed i filtri.
 - Tutte le Action estendono SCAbstractAction e tutte le Form estendono SCActionForm .
- View
 - Struts Taglib (logic, html, bean)
 - sono state utilizzate le librerie di tag di struts per la gestione della visualizzazione dei messaggi di errore, per la visualizzazione delle stringhe in multilingua, per la generazione dei form
 - XTags: Per il parsing ed il rendering di stringhe ed elenchi XML
 - Sviluppo di custom tag: Sono stati sviluppati dei tag custom per integrarsi con le funzionalità sviluppate e per operazioni comuni come la formattazione di stringhe, il ridimensionamento di immagini, ma soprattutto per l' integrazione con i form (i tag devono leggere il valore dal form corrispondente che può essere SCFilterForm o SCObjectForm), la paginazione degli elenchi XML (SCTNavbarTag);
- Sono state sviluppate anche una serie di classi di Utilità (Package it.sintraconsulting.common.util) e di interfacciamento con Hibernate (Package it.sintraconsulting.common.db , classe SCHibernateUtil) per l'accesso alla connessione;

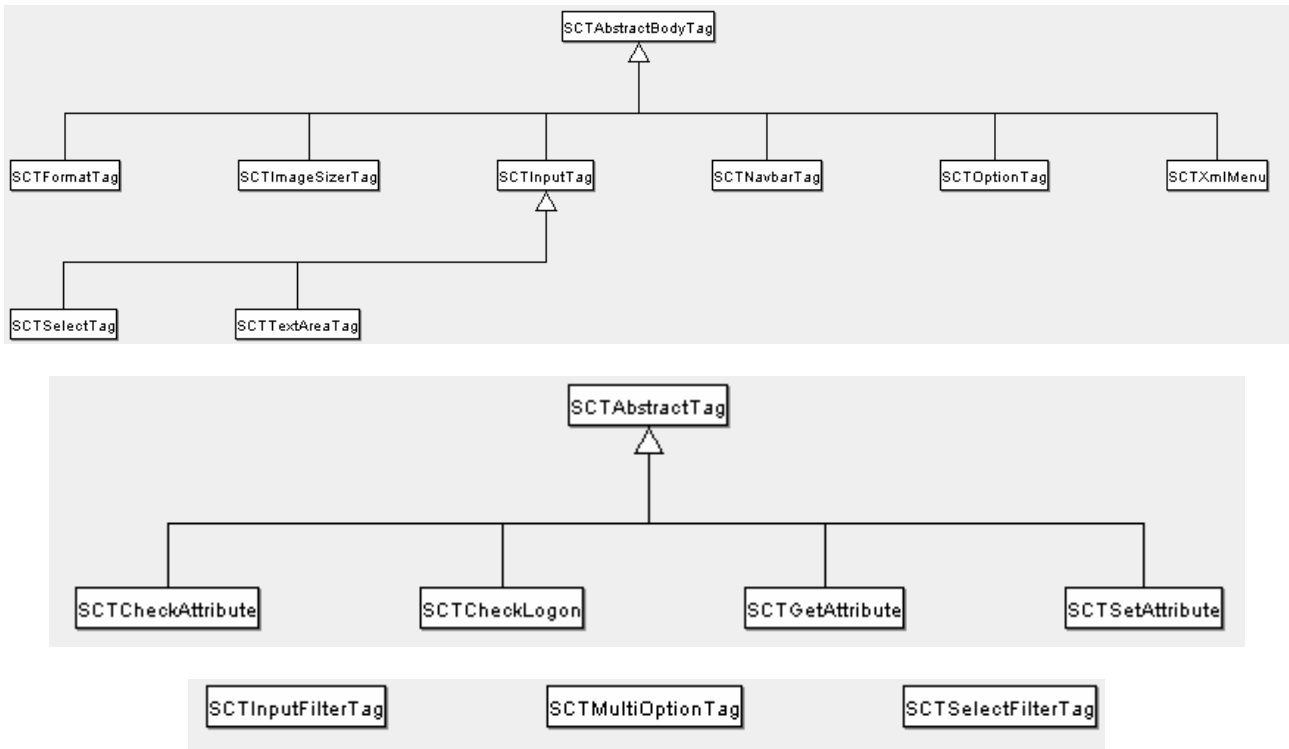
vengono utilizzate le connessioni di Hibernate per avere un unico punto di accesso al pool di connessioni attraverso tale framework;

- Le classi di utilità svolgono una serie di operazioni comuni sulle stringhe (SCStringUtil.java) , sugli stream XML (SCXMLUtil), sui files (SCFileUtil), l'invio della posta (SCEmailUtil); nel package sono presenti inoltre alcune classi di che fungono da interfaccia verso Interfax (il servizio di invio dei Fax www.interfax.net , SCFaxUtil) e verso il servizio di invio / ricezione di SMS (SCSMSUtil verso www.mobyt.it); il servizio di invio SMS utilizza anche un file di configurazione memorizzato nel package "it.sintraconsulting.common.conf ";
- A tutti i livelli il logging è gestito attraverso la classe SCLog del package it.sintraconsulting.common.logging che fa uso della libreria log4j.

3. ANALISI DEI PACKAGE

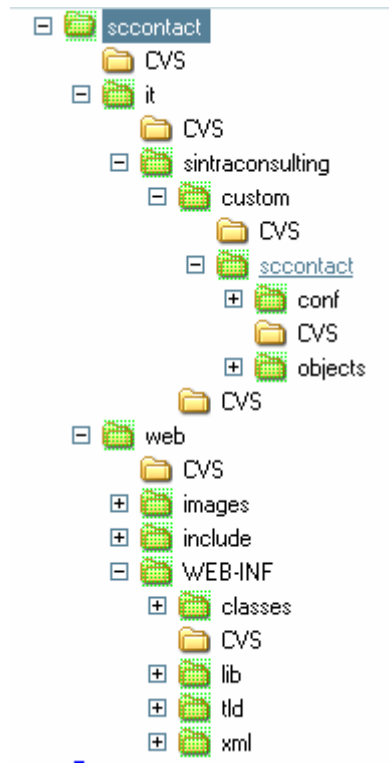
Package it.sintraconsulting.common.taglib

UML Class Diagram



4. STRUTTURA DI UNA APPLICAZIONE

Una applicazione “SCF” è strutturata come una applicazione web standard con la struttura delle cartelle definita nelle specifiche, più alcuni file aggiuntivi peculiari del framework. Di seguito, vediamo un esempio:



- **web:** contiene le pagine jsp, le immagini, i file html etc., eventualmente strutturati in cartelle e sottocartelle (nell'esempio in figura, tutti i file di front-end sono all'interno della cartella principale)
- **WEB-INF:** contiene i file *web.xml* e *struts-config.xml*.
- **WEB-INF/lib:** contiene le librerie java (jar file) necessari al corretto funzionamento della applicazione + 2 JAR file contenenti le classi comuni (it.sintraconsulting.common) del Framework il primo, e le classi Custom sviluppate su misura per la singola applicazione il secondo; questi file, andranno generati all'interno della WEB-INF/lib una volta sviluppata l'applicazione.
- **WEB-INF/classes:** contiene SCConfig.properties (il file di config. Della applicazione) , log4j.properties (il file di configurazione di log4j), hibernate.cfg.xml e Mapping.cfg.xml (i file di configurazione di Hibernate)
- **WEB-INF/xml :** contiene i file xml validator.xml e validator-rules.xml contenenti le configurazioni per il Validator framework di Struts.

- **WEB-INF/tld** : contiene il file TLD con le definizioni delle Tag Library di Struts e di quella sviluppata ad-hoc per il Framework.

5. SVILUPPO DI UNA APPLICAZIONE

Prendiamo come esempio lo sviluppo di una applicazione di gestione dei contatti aziendali (sccontact); seguendo i vari passi dello sviluppo, andremo ad approfondire le varie caratteristiche e funzionalità del framework.

Step principali per lo sviluppo di una nuova applicazione

Lo sviluppo di una applicazione tipo, dovrà seguire i seguenti passi:

1. Analisi applicativa e studio dei requisiti
2. Progettazione E-R e schema logico del database.
3. Realizzazione del database secondo il progetto E-R su PostgreSQL
4. Creazione del progetto su Netbeans
 - Per il progetto, verrà creata una nuova cartella sul cvs nel modulo “custom” ; il nome della cartella dovrà permettere di identificare il progetto (sccontact)
 - All’interno della cartella principale, ci saranno i sorgenti delle classi java e le risorse per il progetto, ed una cartella “web” contenente i file della applicazione web.
 - I sorgenti java, andranno creati nel package *it.sintraconsulting.custom.sccontact* ; all’interno, ci sarà “conf” per le risorse, “object” per gli oggetti POJO(Plain Old Java Objects) da mappare su Hibernate, “struts” con le eventuali estensioni delle classi SCQueryAction o SObjectAction per la gestione degli elenchi e delle anagrafiche; per applicazioni standard, non sarà necessario creare azioni Struts.
5. Copia dei file TLD nella cartella *WEB-INF/tld*; copia del file *web.xml* nella cartella WEB-INF; creazione del file *struts-config.xml*; copia dei file *validator-rules.xml* e creazione del file *validation.xml* nella cartella *WEB-INF/xml*; i file possono essere copiati da un’altra applicazione (fatelo con attenzione !).
6. Creazione dei file di configurazione del progetto nella cartella WEB-INF/classes:
 - *log4j.properties*: file di configurazione di Log4j (jogging)
 - *hibernate.cfg.xml*: file di configurazione di Hibernate (Connessione JDBC)
 - *Mapping.hbm.xml*: contiene la “mappatura” degli oggetti java sulle tabelle.
 - *SCConfig.properties*: file di configurazione della applicazione: vi viene definito il datasource da utilizzare per gli elenchi, il package contenente gli oggetti persistenti, una eventuale classe di inizializzazione (Implementa l’interfaccia *PostInIt*)

7. Creazione del file contenente la definizione delle tabelle virtuali (sccontact.properties)

- Il file con la definizione delle tabelle virtuali (Query SQL per gli elenchi) andrà creato all'interno del package *it.sintraconsulting.custom.sccontact.conf* ; conterrà la definizione delle query strutturata come nell'esempio qui sotto riportato.

```
# Contatti
table.contatti.id=id
table.contatti.fields=*
table.contatti.from=contatti
table.contatti.where=
table.contatti.order=
```

Oltre a questo file di properties, il package conf conterrà anche un file di risorse (sempre un file di properties) contenente le stringhe (multilingua) per l'applicazione.

8. Creazione dei file JSP e delle eventuali classi, modifica dello struts-config.xml

- Creazione del Frameset ;
- Creazione del menu.xml e della pagina menu.jsp ;
- Creazione della pagina contatti.jsp (elenco e cancellazione) e della pagina setcontatti.jsp (Inserimento e Modifica) ; analogo per gruppi.jsp (elenco e cancellazione) e setgruppi.jsp (Inserimento e Modifica)
- Eventuale estensione delle classi SCQueryAction e SCObjectAction ;
- Creazione degli oggetti Java corrispondenti alle tabelle sul database
- Inserimento della "action" "contatti.do" e "setcontatti.do" all'interno del file struts-config.xml ;

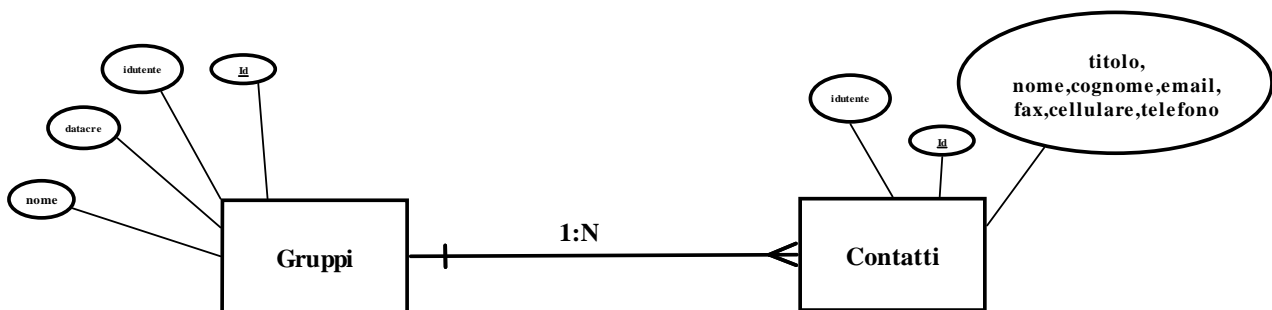
9. Creazione del Context all'interno di Tomcat

- Definizione di un nodo Context nel file server.xml di tomcat
- Definizione del datasource principale della applicazione (per la connessione al DB)

10. Avvio di Tomcat e test della applicazione

Sviluppo di una applicazione di gestione contatti (Rubrica)

- Analisi applicativa e studio dei requisiti (Punto 1)
 - L'applicazione dovrà permettere di gestire una anagrafica gruppi, una anagrafica contatti, e di raggruppare i contatti secondo i gruppi definiti; i contatti dovranno essere ricercabili per nome, cognome, gruppo, email; i gruppi dovranno essere ricercabili per nome; ogni contatto, potrà infine essere associato ad un solo gruppo.
- Progettazione E-R (Punto 2) e schema logico
 - Vediamo lo schema E-R ipotizzato per soddisfare le esigenze della applicazione di gestione contatti.



- Successivamente andiamo a definire lo schema logico del database, con il dettaglio di tutte le tabelle ed i campi. Andrà quindi creato un database sull'istanza PostgreSQL; le tabelle vengono create all'interno dello schema "dev" (per motivi legati non legati alla applicazione).
 - dev.gruppi(id, idutente, nome, datacre)
 - dev.contatti(id, titolo, nome, cognome, email, telefono, cellulare, fax,)
- Realizzazione del database su PostgreSQL (schema fisico)
 - Nella tabella di seguito, si riportano le definizioni SQL delle tabelle da creare sul DB; il prefisso "dev." Permette di creare le tabelle all'interno dello schema dev precedentemente creato.

```
CREATE TABLE dev.gruppi
(
  id serial NOT NULL,
  idutente int4,
  nome varchar(50),
  datacre timestamptz DEFAULT now(),
  CONSTRAINT gruppi_pkey PRIMARY KEY (id)
) WITH OIDS;
GRANT ALL ON TABLE dev.gruppi TO admin WITH GRANT OPTION;
GRANT ALL ON TABLE dev.gruppi TO sintra;
```

```

CREATE TABLE dev.contatti
(
  id serial NOT NULL,
  idgruppo int4,
  titolo varchar(10),
  nome varchar(40),
  cognome varchar(40),
  ragione varchar(50),
  indirizzo text,
  cap varchar(5),
  comune varchar(50),
  citta varchar(50),
  email text,
  email_ufficio text,
  telefono varchar(30),
  telefono_ufficio varchar(30),
  telefono_casa varchar(30),
  cellulare varchar(20),
  cellulare_ufficio varchar(20),
  fax varchar(30),
  fax_ufficio varchar(30),
  piva varchar(16),
  indirizzo_ufficio text,
  cap_ufficio varchar(5),
  citta_ufficio varchar(50),
  CONSTRAINT contatti_pkey PRIMARY KEY (id)
) WITH OIDS;
GRANT ALL ON TABLE dev.contatti TO admin WITH GRANT OPTION;
GRANT ALL ON TABLE dev.contatti TO sintra;

```

- Creazione del progetto su Netbeans
 - Andiamo a creare il progetto su Netbeans;
 - Montare la cartella common
 - Creare una apposita cartella custom (sccontact) per i sorgenti java; all'interno, creare una cartella web per contenere i file della applicazione web.
 - Copiare i file ".tld" in web/WEB-INF/tld
 - Copiare i file *validator-rules.xml* e *validator.xml* in web/WEB-INF/xml e modificare se necessario *validator.xml*
 - Creare la cartella web/WEB-INF/lib e copiare all'interno le librerie necessarie.
 - Copiare il file web/WEB-INF/web.xml ed apportare le opportune modifiche per adattarlo alla nuova applicazione (in generale è necessario modificare solo la parte relativa ad una eventuale autenticazione REALM)

- Copiare il file web/WEB-INF/struts-config.xml per poi apportare le opportune modifiche secondo le azioni previste; di seguito, il file struts-config.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
  <!-- ===== Form Beans Definitions -->
  <form-beans>
    <form-bean name="filterform" type="it.sintraconsulting.common.struts.SCFilterForm"/>
    <form-bean name="objectform" type="it.sintraconsulting.common.struts.SCObjectForm" />
    <form-bean name="uploadform" type="it.sintraconsulting.common.struts.SCUploadForm" />
  </form-beans>
  <!-- ===== Global Exception Definitions -->
  <global-exceptions>
  </global-exceptions>
  <!-- ===== Global Forward Definitions -->
  <global-forwards>
    <forward name="error" path="/error.jsp"/>
  </global-forwards>
  <!-- ===== Action Mapping Definitions -->
  <action-mappings>
    <action path="/contatti"
      scope="session"
      type="it.sintraconsulting.common.struts.SCQueryAction"
      name="filterform"
      validate="true">
      <forward name="success" path="/contatti.jsp"/>
    </action>

    <action path="/setcontatti"
      scope="request"
      type="it.sintraconsulting.common.struts.SCObjectAction"
      name="objectform"
      input="/setcontatti.jsp"
      validate="true">
      <forward name="form" path="/setcontatti.jsp"/>
      <forwardname="success"
path="/contatti.do?tablename=table.contatti&action=filter"/>
    </action>

    <action path="/gruppi"
      scope="session"
      type="it.sintraconsulting.common.struts.SCQueryAction"
      name="filterform"
      validate="true">
      <forward name="success" path="/gruppi.jsp"/>
    </action>
```

```

        <action path="/setgruppi"
            scope="request"
            type="it.sintraconsulting.common.struts.SCObjectAction"
            name="objectform"
            input="/setgruppi.jsp"
            validate="true">
            <forward name="form" path="/setgruppi.jsp"/>
            <forward
                path="/gruppi.do?tablename=table.gruppi&action=filter" />
                name="success"
        </action>

        <!-- Configurazione della dimensione massima per i file in upload attraverso Struts -->
        <!-- -1 Sta per illimitata; se non diversamente specificato, il parametro vale -1 -->
        <controller maxFileSize="2M">
            <!-- The "input" parameter on "action" elements is the name of a
                local or global "forward" rather than a module-relative path -->
            <set-property property="inputForward" value="true"/>
        </controller>
    </action-mappings>

    <!-- ===== Message Resources Comuni -->
    <message-resources parameter="it.sintraconsulting.common.conf.SCFMessages" null="false" />
    <!-- ===== Message Resources Custom per questa applicazione -->
    <message-resources
        parameter="it.sintraconsulting.custom.scontact.conf.SCCMessages" null="false" />
        key="scontact"

    <!-- ===== Plugins -->
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property
            property="pathnames"
            value="/WEB-INF/xml/validator-rules.xml,/WEB-INF/xml/validation.xml" />
    </plug-in>
</struts-config>

```

- Creazione dei file di configurazione del progetto nella cartella WEB-INF/classes:
 - *log4j.properties*: file di configurazione di Log4j (logging)
 - *hibernate.cfg.xml*: file di configurazione di Hibernate (Definizione della Connessione JDBC per hibernate)

```

<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
    PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="connection.datasource">java:comp/env/jdbc/scontact</property>
        <property name="show_sql">true</property>
        <property name="dialect">net.sf.hibernate.dialect.PostgreSQLDialect</property>
        <!-- Mapping files -->
        <mapping resource="Mapping.hbm.xml"/>
    </session-factory>
</hibernate-configuration>

```

```
    </session-factory>
</hibernate-configuration>
```

Mapping.hbm.xml: contiene la "mappatura" degli oggetti java sulle tabelle attraverso Hibernate.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
    PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>
<!-- Gruppi -->
    <class name="it.sintraconsulting.custom.scontact.objects.Gruppo" table="dev.gruppi">
        <id name="id" type="long" column="id">
            <generator class="sequence">
                <param name="sequence">dev.gruppi_id_seq</param>
            </generator>
        </id>
        <property name="idutente"/>
        <property name="nome"/>
        <property name="datacre"/>
        <set name="contatti" inverse="true" lazy="true">
            <key column="idgruppo"/>
            <one-to-many class="it.sintraconsulting.custom.scontact.objects.Contatto"/>
        </set>
    </class>
<!-- Contatti -->
    <class name="it.sintraconsulting.custom.scontact.objects.Contatto"
        table="dev.contatti">
        <id name="id" type="long" column="id">
            <generator class="sequence">
                <param name="sequence">dev.contatti_id_seq</param>
            </generator>
        </id>
        <property name="idutente"/>
        <property name="idgruppo"/>
        <property name="titolo"/>
        <property name="nome"/>
        <property name="cognome"/>
        <property name="ragione"/>
        <property name="indirizzo"/>
        <property name="cap"/>
        <property name="comune"/>
        <property name="citta"/>
        <property name="email"/>
        <property name="email_ufficio"/>
        <property name="telefono"/>
        <property name="telefono_ufficio"/>
        <property name="cellulare"/>
        <property name="cellulare_ufficio"/>
        <property name="fax"/>
        <property name="fax_ufficio"/>
        <property name="piva"/>
    </class>
</hibernate-mapping>
```

```

        <property name="indirizzo_ufficio"/>
        <property name="cap_ufficio"/>
        <property name="citta_ufficio"/>
        <many-to-one          insert="false"          update="false"          name="gruppo"
class="it.sintraconsulting.custom.scontact.objects.Gruppo" column="idgruppo" />
    </class>
</hibernate-mapping>

```

- o **SCConfig.properties**: file di configurazione della applicazione: vi viene definito il datasource da utilizzare per gli elenchi, il package contenente gli oggetti persistenti, una eventuale classe di inizializzazione (Implementa l'interfaccia *PostInit*)

```

#Il Datasource della Applicazione
scf.config.datasource=java:comp/env/jdbc/scontact

#Il package contenente gli oggetti OR Mapped
scf.config.objects.package=it.sintraconsulting.custom.scontact.objects

#Il file di risorse della applicazione contenente la definizione delle tabelle virtuali
scf.custom.config=it.sintraconsulting.custom.scontact.conf.scontact

```

- Creazione del file contenente la definizione delle tabelle virtuali (scontact.properties) e di un file di risorse per le stringhe multilingua custom di questa applicazione (vedi la riga sottoriportata sullo struts-config.xml)

```

<!-- ===== Message Resources Custom per questa applicazione -->
<!--Questo File di Risorse, viene caricato dalla Struts Action Servlet e può essere recuperato
utilizzando il tab <bean> (Es: <bean:message bundle="scontact" key="....." />) dove bundle
corrisponde al parametro key di cui sotto -->
<message-resources key="scontact" parameter="it.sintraconsulting.custom.scontact.conf.SCCMessages"
null="false" />

```

e poi di seguito il file scontact.properties.

```

# Contatti
table.contatti.id=id
table.contatti.fields=*
table.contatti.from=dev.contatti
table.contatti.where=
table.contatti.order=

# Gruppi
table.gruppi.id=id
table.gruppi.fields=*
table.gruppi.from=dev.gruppi
table.gruppi.where=
table.gruppi.order=

# Select Gruppi
# Tabella virtuale utilizzata per la visualizzazione dell'elenco dei gruppi
# all'interno delle selectbox
tableselect.gruppi.id=id
tableselect.gruppi.fields=id,nome as value

```

```

tableselect.gruppi.from=dev.gruppi
tableselect.gruppi.where=
tableselect.gruppi.order=

```

- Creazione dei file JSP e delle eventuali classi, modifica dello struts-config.xml

- Creazione del Frameset;

```

<html>
<head>
<title>Gestione Contatti</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
  <frameset cols="20%,*" cols="*" framespacing="0" frameborder="NO" border="0">
    <frame src="menu.jsp" name="menu" scrolling="NO" noresize>
    <frame src="contatti.do?tablename=table.contatti" name="main">
  </frameset>
<noframes><body>
</body></noframes>
</html>

```

- Creazione del menu.xml e della pagina menu.jsp ;

menu.xml

```

<menu>
  <contatti>
    Gestione Contatti
    <cont href="gruppi.do?tablename=table.gruppi">
      Gruppi
    </cont>
    <cont href="contatti.do?tablename=table.contatti">
      Contatti
    </cont>
  </contatti>
</menu>

```

menu.jsp

```

<%@ taglib uri="struts/html" prefix="html" %>
<%@ taglib uri="scf/common" prefix="scf" %>
<html>
<head>
<link href="<html:rewrite page="/include/scf.css" />" rel="stylesheet" type="text/css">
</head>
<body>
  <%/ *
  XMLMenu
  name = Il nome simbolico del menu
  target = Il target Frame di destinazione (default)
  labelstyle = lo stile SCC per le label
  oicon,icon = le icone di ramo aperto/chiuso
  */%>

```

```

<scf:xmlmenu labelstyle="scf_navbar" name="testmenu" target="parent.main"
oicon="images/fldo.gif" icon="images/fldc.gif">
<%@include file="include/menu.xml"%>
</scf:xmlmenu>
</body>
</html>

```

- o Creazione della pagina contatti.jsp (elenco e cancellazione) e della pagina setcontatti.jsp (Inserimento e Modifica) ; analogo per gruppi.jsp (elenco e cancellazione) e setgruppi.jsp (Inserimento e Modifica).

Gruppi.jsp

```

<%@ taglib uri="struts/bean" prefix="bean" %>
<%@ taglib uri="struts/html" prefix="html" %>
<%@ taglib uri="struts/logic" prefix="logic" %>
<%@ taglib uri="scf/common" prefix="scf" %>
<%@ taglib uri="scf/xtags" prefix="xt" %>
<html>
<head>
<link rel="stylesheet" href="./include/scf.css" type="text/css">
</head>
<body>
<!--
Visualizzazione Messaggi e/o errori.
Message = True sta per i messaggi; l'altro blocco logic:logic:messagesPresent è per gli errori.
I messaggi e gli errori, sono aggiunti alla request all'interno delle azioni struts come oggetti
ActionError o ActionMessage.
Vedere la documentazione di Struts per ulteriori informazioni
-->
<logic:messagesPresent message="true">
<html:messages id="message" message="true"><div class="scf_message"><bean:write name="message"/>
</div></html:messages>
<hr/>
</logic:messagesPresent>
<logic:messagesPresent >
<html:messages id="message" ><div class="scf_error"><bean:write name="message"/>
</div></html:messages>
<hr/>
</logic:messagesPresent>
<!--
Form per le ricerche
I parametri tablename (Il nome della tabella virtuale) ed action (L'azione) sono obbligatori.
Possono essere aggiunti + campi di ricerca che vengono tutti messi in and nella Query risultante.
-->
<html:form action="/gruppi" name="filterform" type="it.sintraconsulting.common.struts.SCFilterForm">
<scf:input name="tablename" type="hidden" value="table.gruppi"/>
<scf:input name="action" type="hidden" value="filter"/>
<div class='scf_label'>
<bean:message key="scf.common.fields.display.name"/>&nbsp;<scf:inputfilter size='10'
styleclass='scf_input' name="nome" type="text" datatype="txt" comparator="ilk" />
<html:submit styleClass='scf_button' >Cerca</html:submit>

```



```

        </tr>
    </xt:forEach>
</form>
</table>
<table width="100%" height="21" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td width="10" background="./images/bas_sx_b.gif">&nbsp;</td>
        <td align="center" class="scf_th_foot">
            <!-- Pulsanti di eliminazione ed inserimento di un nuovo oggetto -->
                <html:button styleClass='scf_button' value="Nuovo" property="nuovo"
onclick="location.href='setgruppi.do?action=insert&object=Gruppo';"/>
                <html:button styleClass='scf_button' value="Elimina Selezionati" property="elimina"
onclick="dodeleteform.submit()"/>
            </td>
        <td width="9" background="./images/bas_dx_b.gif">&nbsp;</td>
    </tr>
</table>
</td>
</tr>
</table>
</body>
</html>

```

Setgruppi.jsp

```

<%@ taglib uri="struts/bean" prefix="bean" %>
<%@ taglib uri="struts/html" prefix="html" %>
<%@ taglib uri="struts/logic" prefix="logic" %>
<%@ taglib uri="scf/common" prefix="scf" %>
<%@ taglib uri="scf/xtags" prefix="xt" %>
<html>
<head>
    <link rel="stylesheet" href="./include/scf.css" type="text/css">
</head>
<body>
<div class='scf_title'> <bean:message bundle="sccontact" key="sccontact.title.managegroups"/>
</div>
<!-- Visualizzazione di messaggi di errore e non -->
<logic:messagesPresent message="true">
    <html:messages id="message" message="true"><div class="scf_message"><bean:write name="message"/>
</div></html:messages>
    <hr/>
</logic:messagesPresent>
<logic:messagesPresent >
    <html:messages id="message" ><div class="scf_error"><bean:write name="message"/>
</div></html:messages>
    <hr/>
</logic:messagesPresent>
<!-- Form di inserimento di un gruppo -->
<center><table>
    <html:form action="/setgruppi" name='objectform'
type='it.sintraconsulting.common.struts.SCObjectForm'>
        <scf:input name="object" type="hidden"/>
        <scf:input name="action" type="hidden"/>
        <scf:input name="objectId" type="hidden"/>

```

```

        <tr>
<!-- Si noti il nome del campo di input che inizia con "db_"; questo indica al framework di
memorizzare il valore nell'oggetto java POJO associato ed indicato nel parametro "object" -->
        <td class="scf_label">Nome: </td><td ><scf:input styleclass='scf_input' name="db_nome"
type="text" />&nbsp;</td>
        </tr>
        <tr>
                <td ><input class='scf_button' onclick="history.go(-1)" type='reset'
value="<bean:message key="scf.common.fields.display.reset"/>" />&nbsp;</td>
                <td class="scf_label"><input class='scf_button' type='submit' value="<bean:message
key="scf.common.fields.display.submit"/>" /></td>
        </tr>
</html:form>
</table></center>
</body>
</html>

```

- Eventuale estensione delle classi SCQueryAction e SObjectAction ; in questo caso non necessitano estensioni delle pagine di gestione dei contatti e dei gruppi.

- Creazione degli oggetti Java corrispondenti alle tabelle sul database

All' interno del package it.sintraconsulting.custom.scontact.objects vanno creati gli oggetti Gruppo e Contatto; per motivi di brevità non vengono qui riportati.

- Inserimento della "action" "contatti.do" e "setcontatti.do" all'interno del file struts-config.xml ; lo stesso per "gruppi.do" e "setgruppi.do"

- Creazione del Context all'interno di Tomcat ed avvio dell'application Server.

```

<Context path="/scontact" docBase="C:\java\cvs\java\custom\scontact\web" debug="9">
  <Resource name="jdbc/scontact" auth="Container" type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/scontact">
    <parameter>
      <name>factory</name>
      <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>org.postgresql.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:postgresql://localhost/scsintra?charSet=utf-8</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>admin</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>admin</value>
    </parameter>
  </ResourceParams>

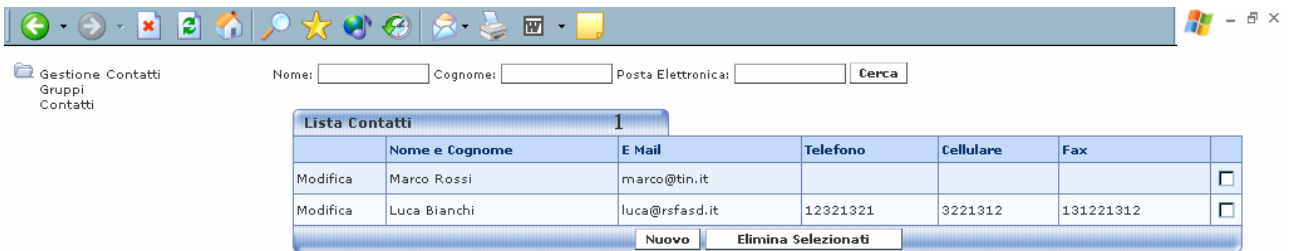
```

```

<parameter>
  <name>maxActive</name>
  <value>100</value>
</parameter>
<parameter>
  <name>maxIdle</name>
  <value>10</value>
</parameter>
<parameter>
  <name>maxWait</name>
  <value>-1</value>
</parameter>
<parameter>
  <name>removeAbandoned</name>
  <value>true</value>
</parameter>
<parameter>
  <name>removeAbandonedTimeout</name>
  <value>60</value>
</parameter>
<parameter>
  <name>logAbandoned</name>
  <value>true</value>
</parameter>
</ResourceParams>
</Context>

```

- Screenshot





APPENDICE A : DIAGRAMMA UML DELLA SCQUERYACTION (ELENCHI)

