1 Introducció

En aquest document s'explica l'estructura del codi font de l'aplicació PINBAL i es detalla la implementació dels diferents components de l'aplicació.

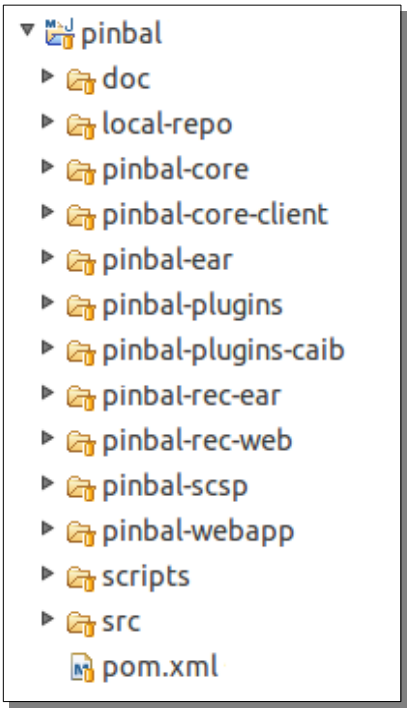
2 Codi font del projecte

El codi font del projecte es pot descarregar de la forja de la CAIB emprant un client de Subversion. L'adreça del repositori és la següent:

```
svn+ssh://scm.gforge.caib.es/svn/pinbal
```

Per a la compilació i desplegament de l'aplicació podeu consultar el manual de instal·lació.

Estructura



El codi font del projecte està dividit en els següents mòduls

- **pinbal-core**: Nucli de l'aplicació a on està implementada la major part de funcionalitat de l'aplicació. Aquesta funcionalitat s'exposa als clients mitjançant EJBs.
- **pinbal-core-client**: Interfícies per a accedir a la funcionalitat del CORE.
- **pinbal-scsp**: Capa de configuració i accés a la funcionalitat proporcionada per les llibreries SCSP.
- **pinbal-webapp**: Mòdul que proporciona una interfície web per a accedir a la funcionalitat de l'aplicació.
- **pinbal-ear**: Mòdul per a generar el fitxer de desplegament de l'aplicació.
- **pinbal-plugins**: Interfícies per als plugins.
- **pinbal-plugins-caib**: Implementació dels plugins per als sistemes d'informació disponibles a la CAIB.
- **pinbal-rec-web**: Mòdul per a generar el WAR del recobriment WS.
- **pinbal-rec-ear**: Mòdul per a generar el EAR del recobriment WS.

A més, a dins el projecte s'hi troben les següents carpetes:

- **doc**: Documentació de l'aplicació
- **local-repo**: Repositori local amb dependències pel Maven
- **scripts**: Scripts d'inicialització i actualització de la base de dades.
- **src**: Carpeta amb el codi font del mòdul principal del projecte.

L'aplicació està desenvolupada emprant tecnologia J2EE amb una arquitectura de 3 capes.

1. Interfície web: Implementada emprant el patró MVC. Les vistes es creen mitjançant pàgines JSP i les llibreries de tags JSTL.
2. Lògica de negoci: Implementada mitjançant EJBs.
3. Capa d'accés a dades: Independent de la base de dades emprant Hibernate com a ORM.

Interfície web

Per a implementar la interfície web s'ha utilitzat la llibreria spring-mvc. Al descriptor de l'aplicació (pinbal-webapp/src/main/webapp/WEB-INF/web.xml) es defineixen els dos arxius de Spring que configuren el context d'aplicació:

Per una banda està l'arxiu pinbal-webapp/src/main/webapp/WEB-INF/application-context-jboss.xml. Es defineix al web.xml de la següent forma:

```
	<listener>
		<listener-class>
			org.springframework.web.context.ContextLoaderListener
		</listener-class>
	</listener>
	<context-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>/WEB-INF/application-context-jboss.xml</param-value>
	</context-param>
```

En aquest arxiu s'hi defineix:

- Els fitxers de properties per al multi-idioma.
- Els beans per a accedir als EJBs del model.
- La importació dels beans que configuren i el control d'accés a l'aplicació mitjançant la llibreria spring-security.

D'una altra banda està l'arxiu de configuració de la llibreria spring-mvc pinbal-webapp/src/main/webapp/WEB-INF/pinbal-servlet.xml. Aquest arxiu es defineix al web.xml a partir del nom del DispatcherServlet de Spring:

```
	<servlet>
		<servlet-name>pinbal</servlet-name>
		<servlet-class>
			org.springframework.web.servlet.DispatcherServlet
		</servlet-class>
		<load-on-startup>1</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>pinbal</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>
```

En aquest arxiu s'hi defineix:

- El paquet a on anar a cercar els components de tipus Controller.
- El directori a on s'emmagatzemen els JSPs per a generar les vistes.
- Els interceptors per a les peticions als components de tipus Controller.

Model

El model està format pels EJBs definits mitjançant l'arxiu de configuració application-context-jboss.xml. En aquest arxiu s'importen els beans que fan de proxy cap als EJBs mitjançant la següent línia:

```
<import resource="classpath:es/caib/pinbal/core/context/application-context-ejb-client.xml" />
```

Els EJBs estan definits al fitxer pinbal-core-client/src/main/resources/es/caib/pinbal/core/context/application-context-ejb-client.xml de la següent forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:jee="http://www.springframework.org/schema/jee"
               xsi:schemaLocation="
                               http://www.springframework.org/schema/beans
                               http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
                               http://www.springframework.org/schema/jee
                               http://www.springframework.org/schema/jee/spring-jee-3.0.xsd">
       <jee:local-slsb
                       id="entitatService"
                       jndi-name="pinbal/EntitatServiceBean/local"
                       business-interface="es.caib.pinbal.core.service.EntitatService"/>
       <jee:local-slsb
                       id="procedimentService"
                       jndi-name="pinbal/ProcedimentServiceBean/local"
                       business-interface="es.caib.pinbal.core.service.ProcedimentService"/>
       <jee:local-slsb
                       id="consultaService"
                       jndi-name="pinbal/ConsultaServiceBean/local"
                       business-interface="es.caib.pinbal.core.service.ConsultaService"/>
       <jee:local-slsb
                       id="serveiService"
                       jndi-name="pinbal/ServeiServiceBean/local"
                       business-interface="es.caib.pinbal.core.service.ServeiService"/>
       <jee:local-slsb
                       id="usuariService"
                       jndi-name="pinbal/UsuariServiceBean/local"
                       business-interface="es.caib.pinbal.core.service.UsuariService"/>
       <jee:local-slsb
                       id="propertyService"
                       jndi-name="pinbal/PropertyServiceBean/local"
                       business-interface="es.caib.pinbal.core.service.PropertyService"/>
       <jee:local-slsb
                       id="aclService"
                       jndi-name="pinbal/AclServiceBean/local"
                       business-interface="org.springframework.security.acls.model.AclService"/>
</beans>
```

La implementació detallada del model es descriu més endavant a la secció dedicada a la lògica de negoci.

Vistes

Els components de les vistes es generen mitjançant pàgines JSP. La situació d'aquestes pàgines JSP està configurada al fitxer pinbal-servlet.xml:

```
       <bean  id="defaultViewResolver"
              class="org.springframework.web.servlet.view.InternalResourceViewResolver">
              <property name="prefix" value="/WEB-INF/jsp/"/>
              <property name="suffix" value=".jsp"/>
       </bean>
```

Segons aquesta configuració els arxius JSP de les vistes estaran situats al directori WEB-INF/jsp de l'arxiu de desplegament, que correspon al directori pinbal-webapp/src/main/webapp/WEB-INF/jsp del projecte.

S'ha intentat reduir la inclusió de codi Java a dins les pàgines JSP al mínim indispensable. Per a

generar el codi HTML s'empren tags JSTL.

Controladors

Aquests components estan definits a dins la carpeta pinbal-webapp/src/main/java/es/caib/pinbal/webapp/controller i s'encarreguen de processar els accessos a les diferents URLs de l'aplicació.

Cada un des controladors s'encarrega de servir un determinat conjunt de URLs, segons la configuració feta amb les anotacions *@RequestMapping*.

Com a exemple el controlador hem agafat la classe pinbal-webapp/src/main/java/es/caib/pinbal/webapp/controller/IndexController.java:

```
package es.caib.pinbal.webapp.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import es.caib.pinbal.webapp.common.RolHelper;

/**
 * Controlador per a la pàgina inicial (index).
 */
@Controller
@RequestMapping("/index")
public class IndexController {
	@RequestMapping(method = RequestMethod.GET)
	public String get(HttpServletRequest request) {
		if (RolHelper.isRolActualDelegat(request)) {
			return "redirect:consulta";
		} else if (RolHelper.isRolActualRepresentant(request)) {
			return "redirect:representant/usuari";
		} else if (RolHelper.isRolActualAdministrador(request)) {
			return "redirect:entitat";
		} else if (RolHelper.isRolActualAuditor(request)) {
			return "redirect:auditor";
		} else if (RolHelper.isRolActualSuperauditor(request)) {
			return "redirect:superauditor";
		} else {
			return "delegatNoAutoritzat";
		}
	}
}
```

Les anotacions a nivell de classe *@Controller* i *@RequestMapping* indiquen que aquesta classe és un component de tipus controlador que s'encarregarà de processar les URLs relatives al context de l'aplicació amb el prefix "/index".

L'anotació *@RequestMapping* a nivell del mètode get indica que aquest mètode serà l'encarregat de processar les peticions de tipus GET per a l'URL "/index". Les altres peticions a una URL diferent o amb un mètode diferent de GET seran ignorades per aquest controlador.

El mètode get retorna un *string* indicant la vista encarregada de generar el codi HTML que es retornarà com a resposta. En aquest cas, re-dirigeix cap a diferents vistes en funció del rol de l'usuari actual.

Decoració de les vistes

Totes les pàgines HTML de l'aplicació tenen parts en comú com, per exemple, la capçalera i el peu. Per evitar tenir que repetir aquestes parts a cada una de les vistes empram un patró *decorator* mitjançant la llibreria Sitemesh (http://www.sitemesh.org).

Per a activar la funcionalitat d'aquesta llibreria es configura el següent filtre al descriptor de

l'aplicació web web.xml:

```
	<filter>
		<filter-name>sitemesh</filter-name>
		<filter-class>com.opensymphony.module.sitemesh.filter.PageFilter</filter-class>
	</filter>
	<filter-mapping>
		<filter-name>sitemesh</filter-name>
		<url-pattern>*</url-pattern>
	</filter-mapping>
```

El fitxer de configuració pinbal-webapp/src/main/webapp/WEB-INF/decorators.xml és el següent:

```
<decorators defaultdir="/WEB-INF/jsp/decorators">
	<excludes>
		<pattern>/js*</pattern>
		<pattern>/css*</pattern>
		<pattern>/img*</pattern>
		<pattern>/ico*</pattern>
	</excludes>
	<decorator name="default" page="default.jsp">
		<pattern>/*</pattern>
	</decorator>
</decorators>
```

Segons aquesta configuració, les URLs amb el prefixos js, css, img i ico no es decoraran. Les demés peticions es decoraran segons el contingut de la JSP pinbal-webapp/src/main/webapp/WEB-INF/jsp/decorators/default.jsp.

Seguretat i control d'accés

Per restringir l'accés a l'aplicació i per a controlar i gestionar els permisos que té un usuari per a accedir a algunes parts de l'aplicació empram la llibreria Spring Security (http://www.springsource.org/spring-security).

Aquesta llibreria es configura al descriptor de l'aplicació web web.xml:

```
	<filter>
		<filter-name>springSecurityFilterChain</filter-name>
		<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
	</filter>
	<filter-mapping>
		<filter-name>springSecurityFilterChain</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>
```

La configuració per String Security està inclosa a dins l'arxiu application-context-jboss.xml:

```
<import resource="classpath:es/caib/pinbal/webapp/context/application-context-security.xml" />
```

Model i lògica de negoci

El context de Spring del model de l'aplicació es defineix al fitxer pinbal-core/src/main/resources/es/caib/pinbal/core/context/application-context-model.xml. A continuació es detallen les seves parts principals:

- Definició de components (services i repositories):

```
	<context:annotation-config />
	<context:component-scan base-package="es.caib.pinbal.core.service" />
	<jpa:repositories base-package="es.caib.pinbal.core.repository" />
```

- Definició del datasource i del EntityManager de JPA:

```
	<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
		<property name="resourceRef" value="true" />
		<property name="jndiName" value="${config:es.caib.pinbal.datasource.jndi}" />
	</bean>
```

```
	<bean id="entityManagerFactory"
		class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
		<property name="dataSource" ref="dataSource" />
		<property name="packagesToScan">
			<list>
				<value>es.caib.pinbal.core.model</value>
			</list>
		</property>
		<property name="jpaDialect">
			<bean class="org.springframework.orm.jpa.vendor.HibernateJpaDialect" />
		</property>
		<property name="jpaVendorAdapter">
			<bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter" />
		</property>
		<property name="jpaProperties">
			<props>
				<prop key="hibernate.dialect">$
{config:es.caib.pinbal.hibernate.dialect}</prop>
				<prop key="hibernate.show_sql">$
{config:es.caib.pinbal.hibernate.show_sql}</prop>
				<prop key="hibernate.hbm2ddl.auto">$
{config:es.caib.pinbal.hibernate.hbm2ddl.auto}</prop>
			</props>
		</property>
	</bean>
```

- Definició del gestor de transaccions:

```
	<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
		<property name="entityManagerFactory" ref="entityManagerFactory" />
	</bean>
	<tx:annotation-driven transaction-manager="transactionManager" />
```

La lògica de negoci de l'aplicació està definida mitjançant interfícies. Aquestes interfícies es poden consultar a dins la carpeta pinbal-core-client/src/main/java/es/caib/pinbal/core/service.

Les classes Java que implementen aquestes interfícies es poden trobar a la carpeta pinbal-core/src/main/java/es/caib/pinbal/core/service. En aquestes classes es troben anotacions que son interpretades pel framework Spring per oferir diversa funcionalitat:

- @Service: Aquesta anotació defineix a la classe com un component de la lògica de negoci.
- @Resource: Els atributs de la classe amb aquesta anotació s'inicialitzaran automàticament fent una cerca a dins el context d'aplicació de Spring d'una classe amb el mateix tipus.
- @Transactional: Per indicar que les cridades mètodes amb aquesta anotació s'hauran d'executar a dins una transacció. Per a més detalls sobre els paràmetres es pot consultar la documentació d'Spring.

També s'ha desenvolupat una capa EJB cap a les implementacions de les interfícies de lògica de negoci. Els EJBs s'han implementat seguint l'estàndard EJB3 mitjançant anotacions i es poden consultar a la carpeta pinbal-core/src/main/java/es/caib/pinbal/core/ejb. Les classes dels EJBs no implementen cap tipus de lògica de negoci i deleguen tota la seva funcionalitat cap a les implementacions de la carpeta pinbal-core/src/main/java/es/caib/pinbal/core/service.

Accés a dades

Per a l'accés a les taules de base de dades s'empra l'implementació de l'estàndard JPA de Hibernate. Les classes que representen les taules de base de dades es poden trobar a la carpeta pinbal-core/src/main/java/es/caib/pinbal/core/model.

L'accés a dades es du a terme mitjançant la llibreria Spring Data JPA. Aquesta llibreria simplifica enormement les operacions bàsiques sobre les taules de base de dades mitjançant la classe JpaRepository. Per a mostrar la potència d'aquesta llibreria podem emprar el següent exemple:

```
package es.caib.pinbal.core.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import es.caib.pinbal.core.model.ServeiConfig;

public interface ServeiConfigRepository extends JpaRepository<ServeiConfig, Long> {
	ServeiConfig findByServei(String servei);
}
```

Una vegada definida aquesta classe, la podem emprar des d'un objecte de lògica de negoci amb el següent bocí de codi:

```
	@Resource
	private ServeiConfigRepository serveiConfigRepository;
```

I ja tenim disponibles les operacions bàsiques d'alta, modificació i baixa:

- serveiConfigRepository.save(serveiConfig);
- serveiConfigRepository.delete(id);

I les següents operacions de consulta:

- serveiConfigRepository.findOne(id);
- serveiConfigRepository.findAll();
- serveiConfigRepository.findByServei(servei);

Per a més informació podeu consultar la documentació de la llibreria Spring Data JPA.

3 Multi-idioma

Tots els missatges de l'aplicació es troben codificats en fitxers de properties per a facilitar la traducció de l'aplicació a altres idiomes. Els fitxers són els següents:

- Missatges de les vistes i controladors:
  pinbal-webapp/src/main/resources/es/caib/pinbal/webapp/i18n/messages.properties.
- Missatges del core:
  pinbal-core/src/main/resources/es/caib/pinbal/core/i18n/messages.properties.
- Missatges dels justificants:
  pinbal-scsp/src/main/resources/es/caib/pinbal/scsp/i18n/messages.properties.

Per defecte l'idioma emprat a l'aplicació és el Català. Si es volgués traduir, per exemple, al castellà s'haurien de crear els fitxers messages_es.properties a dins els mateixos directoris amb els missatges traduïts al castellà.

Multi-idioma dels justificants

El multi-idioma dels justificants que es generen automàticament a partir de les respostes a les peticions SCSP està definit al fitxer
pinbal-scsp/src/main/resources/es/caib/pinbal/scsp/i18n/messages.properties.

Donat que el format de les dades específiques difereix per a cada servei SCSP i que, en un futur, es poden anar afegint nous serveis a l'aplicació no ha estat possible definir tots els possibles missatges per a mostrar els justificants totalment traduïts.

Per a afegir un nou missatge d'un servei que encara no estigui traduït s'ha de modificar el fitxer messages.properties i afegir una línia amb el següent format:

dades.especifiques.<CODI_SERVEI>.<PATH_CAMP>=<MISSATGE>

Per exemple, si volem afegir un missatge per al camp de dades específiques /Domicilio/ProvinciaRespuesta del servei amb codi VDRSFWS02 amb el missatge “Província”, la línia quedaria així:

dades.especifiques.VDRSFWS02.Domicilio.ProvinciaRespuesta=Província