Sistema Integrado de Administración Tributaria
SIAT

Arquitectura SIAT
Anexo III – Ejemplo Paso a Paso

Noviembre 2007

Contenido

Características del documento

Versiones

Fecha	Version	Descripcion	Autor
08/11/07	1.1	Versión inicial	tecso:
26/12/07		Paginación sin conteo.	tecso:

Objetivos del documento

El objetivo del documento es describir, paso a paso, la construcción de un caso de uso de SIAT con la arquitectura SIAT.

Introducción

Previa lectura del presente, se recomienda leer la documentación de la arquitectura “demoda”.

En adelante explicaremos como codificar un caso de uso para el proyecto SIAT.

Partimos de la documentación del caso de uso y las tablas correspondientes al mismo creadas en la base de datos.

Como ejemplo utilizaremos el caso de uso "Mantenedor de Atributo".

Iface

Comenzaremos construyendo las clases correspondientes al iface y los métodos de la interfaz de servicio determinado que objetos reciben y los que devuelven.

Tendremos una Interfaz IService por cada submódulo, el o los Value Object, Adapter y SearchPage correspondientes organizados de la siguiente manera:

```
/siat/iface/src/ar/gov/rosario/siat/<modulo>/iface/model/
                                                    /<Bean>VO.java
                                                    /<Bean>SearchPage.java
                                                    /<Bean>Adapter.java
/siat/iface/src/ar/gov/rosario/<modulo>/def/iface/service/
                                                     /I<Submodulo>Service
```

AtributoVO.java

Esta ubicado en la siguiente carpeta:

```
/siat/iface/src/ar/gov/rosario/siat/def/iface/model
```

Es el Value Object que le corresponde al Bean "Atributo" del buss.
Extiende de SiatBussImageModel.
Contiene las propiedades y métodos del Atributo disponibles en la vista.

```
public class AtributoVO extends SiatBussImageModel {
```

Llamamos NAME a una propiedad estática que contiene al nombre de la clave utilizada para cargar una instancia de AtributoVO en el userMap del UserSession.

Además el mismo nombre es utilizado para enviar al objeto por request conteniendo los datos para renderizar las páginas jsp.

```
public static final String NAME = "atributoVO";
```

Propiedades simples:

```
private String codAtributo;
private String desAtributo;
private String valorDefecto;
```

Las propiedades que son ensambles a otros VO generalmente instanciadas:

Nota: Tener sumo cuidado de no crear referencias cíclicas, en tal caso no instanciarlas.

```
	private TipObjImpAtrVO tipObjImpAtr = new TipObjImpAtrVO();
	private TipoAtributoVO tipoAtributo = new TipoAtributoVO();
	private DomAtrVO domAtr = new DomAtrVO();
```

Propiedades que son enumeraciones pertenecientes a Iface:
La propiedad esAtributoBusqueda es una instancia de la enumeración SiNo. Corresponde a la propiedad esAtributoBusqueda de la clase Atributo del buss de tipo Integer.

```
	private SiNo esAtributoBusqueda = SiNo.OpcionSelecionar;
	private SiNo admBusPorRan = SiNo.OpcionSelecionar;
	private SiNo admBusMulVal = SiNo.OpcionSelecionar;
```

Banderas utilizadas en los logicEqual de los jsp para gestionar la visualización y/o habilitación de acciones en el jsp.

```
	// Buss Flag
	private String	modifCodAtributoEnabled;
	private String	modifTipoAtributoEnabled;
	private String	modifDomAtrEnabled;
```

Constructor de AtributoVO, aquí se setean los valores de las propiedades utilizadas por seguridad para determinar el permiso de acceso a las acciones sobre el atributo.

```
	// Constructores
	public AtributoVO() {
		super();
		// Acciones y Metodos para seguridad
		super.ACCION_VER		= "/def/AdministrarAtributo";
		super.ACCION_MODIFICAR		= "/def/AdministrarAtributo";
		super.ACCION_ELIMINAR		= "/def/AdministrarAtributo";
	}
```

Getters y Setters de todas las propiedades:

```
	// Getters y Setters
	public SiNo getAdmBusMulVal() {
		return admBusMulVal;
	}
	public void setAdmBusMulVal(SiNo admBusMulVal) {
		this.admBusMulVal = admBusMulVal;
	}
```

Getters adecuados para la Vista:
Para el caso de fechas, horas y los tipos de datos numéricos Integer, Double, Float y Long agregaremos una propiedad con el mismo nombre pero finalizada en la cadena "View" y de tipo String.

```
	private Date	fechaDesde;
	private Date	horaDesde;
	private Long	numeroLong;

	private String	fechaDesdeView;
	private String	horaDesdeView;
	private String	numeroLongView;
```

Para las cuales se crean los getters, pero no los setters, ya que en los setters de la propiedades originales, se seteará también el valor formateado a la correspondiente propiedad "View".

```
	public Long getNumeroLong() {
		return numeroLong;
	}
```

```
	public void setNumeroLong(Long numeroLong) {
		this.numeroLong = numeroLong;
		this.numeroLongView = numeroLong.toString();
	}

	public String getNumeroLongView() {
		return numeroLongView;
	}
```

Para el caso de fecha y horas se definieron las siguientes máscaras como anotaciones java:

```
	DDsMMsYYYY_MASK.java = "dd/MM/yyyy"
	DD_MM_YYYY_MASK.java = "dd-MM-yyyy"
	DDMMYYYY_MASK.java   = "ddMMyyyy"
	YYYYMM_MASK.java     = "yyyyMM"
	DDsMMsYY_MASK.java   = "dd/MM/yy"
	DD_MM_YY_MASK.java   = "dd-MM-yy"
	DDMMYY_MASK.java     = "ddMMyy"
	YYYY_MM_DD_MASK.java = "yyyy-MM-dd"
	YYYY_MM_DD_HH_MM_SS_MASK.java = "yyyy-MM-dd HH:MM:SS"
	DDsMMsYYYY_HH_MM_MASK.java    = "dd/MM/yyyy HH:MM"
	HOUR_MINUTE_MASK.java          = "HH:mm"
	MINUTE_MASK.java               = "mm"
```

De las cuales se asume por omisión que DDSMMSYYYY_MASK es la mascara por defecto.

```
	public Date getFechaDesde() {
		return fechaDesde;
	}
	public void setFechaDesde(Date fechaDesde) {
		this.fechaDesde = fechaDesde;
		this.fechaDesdeView = DateUtil.formatDesde(fechaDesde, "dd/MM/yyyy");
	}

	public String getFechaDesdeView() {
		return fechaDesdeView;
	}
```

O si se desea utilizar una máscara al momento de validar el formato, se puede agregar en modo de annotation arriba del setter.

```
	public Date getHoraDesde() {
		return horaDesde;
	}
	@HOUR_MINUTE_MASK
	public void setHoraDesde(Date horaDesde) {
		this.horaDesde = horaDesde;
		this.horaDesdeView = DateUtil.formatDesde(this.horaDesde, "HH:mm");
	}

	public String getHoraDesdeView() {
		return horaDesdeView;
	}
```

Al momento de construir la/s jsp/s donde se utilice este VO, siempre usaremos las propiedades terminadas en “View”.

AtributoSearchPage.java

Esta ubicado en la misma carpeta:

```
/siat/iface/src/ar/gov/rosario/siat/def/iface/model
```

Es el model utilizado por la vista para resolver la búsqueda de Atributos, posee filtros de búsqueda y lista de resultado. Posee banderas para resolver la seguridad de acceso a las acciones y listas para llenar los combos necesarios.
Extiende de SiatPageModel.
El Código es similar al de AtributoVO.

```
	public class AtributoSearchPage extends SiatPageModel {

		private static final long serialVersionUID = 1L;

		public static final String NAME = "atributoSearchPageVO";

		private String codAtributo="";
		private String desAtributo="";
		private List<TipoAtributoVO> listTipoAtributo = new
ArrayList<TipoAtributoVO>();

		public AtributoSearchPage() {
		super();

		// Acciones y Metodos para seguridad
		super.ACCION_AGREGAR    = "/def/AdministrarAtributo";
		super.ACCION_MODIFICAR  = "/def/AdministrarAtributo";
		super.ACCION_ELIMINAR   = "/def/AdministrarAtributo";
		}
		// Getters y Setters
		{...}
	}
```

AtributoAdapter.java

Cambien está ubicado en:

```
/siat/iface/src/ar/gov/rosario/siat/def/iface/model
```

Es el model utilizado por la vista para crear, ver, modificar, eliminar y demás acciones sobre un Atributo, posee banderas para resolver la seguridad de acceso a las acciones.

```
	public class AtributoAdapter extends SiatAdapterModel{

		public static final String NAME = "atributoAdapterVO";

		private AtributoVO atributo = new AtributoVO();

		private List<TipoAtributoVO> listTipoAtributo = new
ArrayList<TipoAtributoVO>();
		private List<DomAtrVO>      listDomAtr = new ArrayList<DomAtrVO>();
		private List<SiNo>          listSiNo = new ArrayList<SiNo>();
			// Constructor
		public AtributoAdapter(){
		}
		//  Getters y Setters
		{...}
	}
```

IAtributoService.java

Se ubica en la siguiente carpeta:

```
/siat/iface/src/ar/gov/rosario/siat/def/iface/service/
```

Representa la interfaz de servicio para el submódulo “Atributo” del modulo “Definición” (def).

En general, para el caso de un Mantenedor, los métodos públicos de **IAtributoService** son:

```
	public AtributoAdapter getAtributoAdapterForView(UserContext userContext,
CommonKey commonKey)
                                                                    throws
	DemodaServiceException;
	public AtributoAdapter getAtributoAdapterForCreate(UserContext userContext)
                                                                    throws
DemodaServiceException;
	public AtributoAdapter getAtributoAdapterForUpdate(UserContext userContext,
CommonKey                                                           commonKey)
throws DemodaServiceException;
	public AtributoAdapter getAtributoAdapterParamTipoAtributo(UserContext
userContext,AtributoAdapter
	atributoAdapter) throws DemodaServiceException;
	public AtributoVO createAtributo(UserContext userContext, AtributoVO atributoVO )
                                                                    throws
DemodaServiceException;
	public AtributoVO updateAtributo(UserContext userContext, AtributoVO atributoVO )
                                                                    throws
DemodaServiceException;
	public AtributoVO deleteAtributo(UserContext userContext, AtributoVO atributoVO )
                                                                    throws
DemodaServiceException;
	public AtributoSearchPage getAtributoSearchPageInit(UserContext usercontext)
                                                                    throws
DemodaServiceException;
	public AtributoSearchPage getAtributoSearchPageResult(UserContext usercontext,
AtributoSearchPage                                                  atributoSearchPage)
throws DemodaServiceException;
	public AtributoSearchPage getAtributoSearchPageParamTipoAtributo(UserContext
userContext,                                        AtributoSearchPage atributoSearchPage)
throws DemodaServiceException;
```

Buss

Primero se realiza el Diseño de Asignación de responsabilidades dentro del buss.

Continuamos construyendo las clases de negocio y la implementación de los servicios correspondientes, organizados de la siguiente manera.

```
	/siat/buss/src/ar/gov/rosario/siat/<modulo>/buss/bean/
                                                                    /<Bean>.java
	/siat/buss/src/ar/gov/rosario/siat/<modulo>/buss/dao/
                                                                  /<Bean>DAO.java
                                                                  /<Modulo>DAOFactory.java
	/siat/buss/src/ar/gov/rosario/siat/<modulo>/buss/service/
                                                                    /<Bean>ServiceHbmImpl.java
```

Atributo.java

Para el caso de Atributo tenemos en la siguiente ubicación:

```
/siat/buss/src/ar/gov/rosario/siat/def/buss/bean/
```

Clase de Negocio que extiende de BaseBO (perteneciente a demoda) y que encapsula la responsabilidad correspondiente al Atributo de siat.

Una vez determinadas las propiedades de la clase Atributo, mediante anotaciones de Hibernate se realiza el mapeo a la tabla “def_atributo”:

```
	@Entity
```

```
@Table(name="def_atributo")
public class Atributo extends BaseBO {
```

Y sus propiedades con los correspondientes campos de la tabla:

```
@Column(name="codAtributo")
private String codAtributo;

@Column(name="desAtributo")
private String desAtributo;
```

Determinadas las colaboraciones con otras clases de negocio como TipoAtributo, DomAtr, también se mapean si es necesario:

```
@ManyToOne(optional=false)
@JoinColumn(name="idTipoAtributo")
private TipoAtributo tipoAtributo;
```

Luego se escriben los getters y settes de todas las propiedades.

Y por convención siempre creamos los siguientes métodos:

El getById() para recuperar un objeto dado su Id.

```
public static Atributo getById(Long id) {
	return (Atributo) DefDAOFactory.getAtributoDAO().getById(id);
}
```

Y los tres métodos de validación para las acciones básicas create, update y delete.
Aquí se limpia la lista de errores.
Se ejecutan validaciones de datos requeridos y rangos.
Se ejecutan validaciones de negocio (FK, UK, Reglas de Negocio, etc.).
En caso de no cumplir las validaciones, se carga la lista de errores en el objeto y devuelve false.

```
public boolean validateCreate() {

	UniqueMap uniqueMap = new UniqueMap();

	//limpiamos la lista de errores
	clearError();

	// Validaciones de VO
	// validar que la descripcion no sea nula ni vacia
	if (StringUtil.isNullOrEmpty(getDesAtributo())) {
		addRecoverableError(DefError.ATRIBUTO_DESATRIBUTO_REQUIRED);
	}

	if (hasError()) {
		return false;
	}

	// Validaciones de reglas de negocio
	// Validar que codAtributo sea unico en la tabla
	uniqueMap.addString("codAtributo");
	if(!GenericDAO.checkIsUnique(this, uniqueMap)){
		addRecoverableError(DefError.ATRIBUTO_CODATRIBUTO_UNIQUE);
	}

	if (hasError()) {
		return false;
	}

	return true;
}
```

El código de validateUpdate() y validateDelete() es similar al de validateCreate():

```
public boolean validateUpdate()
public boolean validateDelete()
```

Y después se determinan los métodos de negocio que resuelven el caso de uso en cuestión de acuerdo al pattern GRASP, por ejemplo calcularMonto(), calcularFechaVencimiento(), devolverListaDominiosActivos(), etc.

Una vez construida la clase hay que agregarla a los dos archivos "hibernate.cfg" del buss y del view, ejemplo:

Para poder utilizarla en el test, en el archivo:

```
/siat/buss/src/siat.hibernate.cfg.xml
```

agregar la entrada:

```
<mapping class="ar.gov.rosario.siat.def.buss.bean.Atributo"/>
```

después del comentario:

```
<!-- List of annotated classes-->
```

Para la ejecución en Tomcat agregar la misma entrada en el archivo:

```
/siat/view/src/WEB-INF/src/siat.hibernate.cfg.xml
```

DefManager.java

Es un Singleton correspondiente al módulo definición, donde se delegan las responsabilidades que no pueden ser asumidas por ningún objeto de negocio, por ejemplo Create, Update y Delete de Atributos.

createAtributo()

Aquí se invoca la validación correspondiente.
Si es exitosa, invoca al método de AtributoDAO correspondiente.
Sino devuelve el Atributo con la lista de errores cargada.

```
public static Atributo createAtributo(Atributo atributo) throws Exception {
        // Validaciones de negocio
        if (!atributo.validateCreate()) {
                return atributo;
        }
        DefDAOFactory.getAtributoDAO().update(atributo);
        return atributo;
}
```

El código de updateAtributo() y deleteAtributo() es similar al de createAtributo():

```
public static Atributo updateAtributo(Atributo atributo) throws Exception{...}
public static Atributo deleteAtributo(Atributo atributo) throws Exception{...}
```

AtributoDAO.java

La sección DAO se encuentra en la carpeta:

```
/siat/buss/src/ar/gov/rosario/siat/def/buss/dao
```

Clase que extiende de GenericDAO.
Aquí se resuelve el armado de consultas HQL con la Base de Datos.

En el constructor le pasamos la clase del Bean correspondiente (Atributo) a la superclase GenericDAO.

```
	public AtributoDAO() {
		super(Atributo.class);
	}
```

El método findBySearchPage() resuelve la búsqueda, recibe un "atributoSearchPage" con los filtros seteados y retorna una lista de "Atributos".
Aquí podemos ver la utilización de *isNullOrEmpty() que evalua si es string recibido es null o vacio.*
*También escaparUpper(), la cual r*eemplaza los caracteres no validos para el informix y devuelve la cadena en mayusculas.

```
	public List<Atributo> findBySearchPage(AtributoSearchPage atributoSearchPage)
throws Exception {
		String funcName = DemodaUtil.currentMethodName();
		if (log.isDebugEnabled()) log.debug(funcName + ": enter");

		String queryString = "from Atributo t ";
		boolean flagAnd = false;

		if (log.isDebugEnabled()) {
			log.debug("log de filtros del AtributoSearchPage: " +
atributoSearchPage.infoString());
		}
		// Armamos filtros del HQL
		if (atributoSearchPage.getModoSeleccionar()) {
			queryString += flagAnd ? " and " : " where ";
			queryString += " t.estado = "+ Estado.ACTIVO.getId();
			flagAnd = true;
		}
		// filtro por codAtributo
		if (!StringUtil.isNullOrEmpty(atributoSearchPage.getCodAtributo())) {
		queryString += flagAnd ? " and " : " where ";
			queryString += " UPPER(TRIM(t.codAtributo)) like '%" +
			StringUtil.escaparUpper(atributoSearchPage.getCodAtributo()) + "%'";
			flagAnd = true;
		}
		// filtro por descAtributo
		if (!StringUtil.isNullOrEmpty(atributoSearchPage.getDesAtributo())) {
		queryString += flagAnd ? " and " : " where ";
			queryString += " UPPER(TRIM(t.desAtributo)) like '%" +
			StringUtil.escaparUpper(atributoSearchPage.getDesAtributo()) + "%'";
			flagAnd = true;
		}
		// Order By
		queryString += " order by t.codAtributo ";

		if (log.isDebugEnabled()) log.debug(funcName + ": Query: " + queryString);
```

El executeCountedSearch() se encarga de:
Obtener el conjunto de resultado.
Setear los valores necesarios para realizar la paginación en el searchPage recibido.

```
                List<Atributo> listAtributo = (ArrayList<Atributo>)
executeCountedSearch(queryString,
                                                atributoSearchPage);
                if (log.isDebugEnabled()) log.debug(funcName + ": exit");
                return listAtributo;
        }
```

DefDAOFactory.java

Esta clase es un singleton que devuelve una instancia de cada DAO del módulo Def.
Cada vez que se crea un nuevo DAO que corresponda al módulo Def, hay que agregarlo como propiedad:

```
        private AtributoDAO         atributoDAO;
```

Instanciarlo en el constructor:

```
        private DefDAOFactory() {
                super();
                this.atributoDAO            = new AtributoDAO();
```

Y agregar el método estático getter:

```
        public static AtributoDAO getAtributoDAO() {
                return INSTANCE.atributoDAO;
        }
```

AtributoServiceHbmImpl.java

Por último tenemos el service en :

```
/siat/buss/src/ar/gov/rosario/siat/def/buss/service
```

Implementación de la interfaz **IAtributoService** correspondiente.
Para el caso de un mantenedor, generalmente los métodos son:

- *getAtributoSearchPageInit()*
- *getAtributoSearchPageParam()*
- *getAtributoSearchPageResult()*
- *getAtributoAdapterForView()*
- *getAtributoAdapterForCreate()*
- *getAtributoAdapterForUpdate()*
- *getAtributoAdapterForDelete()*
- *getAtributoAdapterParamForUpdate() y/o getAtributoAdapterParamForCreate()*
- *createAtributo()*
- *updateAtributo()*
- *deleteAtributo()*

Ahora Describiremos cada uno en detalle:

getAtributoSearchPageInit()

Setea el userContext recibido en la propiedad userContext de tipo ThreadLocal de DemodaUtil, para tenerlo disponible posteriormente.
Se abre la session de Hibernate.
Se instancia un AtributoSearchPage
Inicializa los valores por defecto de la búsqueda de Atributo.
Carga las listas para los combos utilizados en la búsqueda (si es necesario), pasando cada lista de BO a listas de VO.

Cierra la session de Hibernate.
Devuelve el AtributoSearchPage
En caso de ocurrir una Exception, dispara una DemodaServiceException().

```
	public AtributoSearchPage getAtributoSearchPageInit(UserContext usercontext) throws

		DemodaServiceException {
		String funcName = DemodaUtil.currentMethodName();
		if (log.isDebugEnabled()) log.debug(funcName + ": enter");

		try {
			DemodaUtil.setCurrentUserContext(userContext);
			SiatHibernateUtil.currentSession();

			AtributoSearchPage atributoSearchPage = new AtributoSearchPage();

			atributoSearchPage.setListTipoAtributo( (ArrayList<TipoAtributoVO>)
				ListUtil.toVO(TipoAtributo.getList(),new TipoAtributoVO(-1,

	StringUtil.SELECT_OPCION_TODOS)));

			if (log.isDebugEnabled()) log.debug(funcName + ": exit");
			return atributoSearchPage;

		} catch (Exception e) {
			log.error("ServiceError en: ", e);
			throw new DemodaServiceException(e);
		} finally {
			SiatHibernateUtil.closeSession();
		}

	}
```

---

getAtributoSearchPageParam()

Generalmente se utiliza en una recarga de listas enlazadas.
Setea el userContext.
Abre la session de Hibernate.
Limpia la lista de errores.
Carga la/s lista/s enlazada/s dado el valor seleccionado, para esto invoca al método correspondiente del BO, y se pasa la lista de BO a VO de la siguiente manera:

```
	List<DomAtr> listDomAtr = DomAtr.findByTipoAtributoActivos(new
			TipoAtributo(atributoSearchPage.getTipoAtributo().getId()));


	atributoSearchPage.setListDomAtr((ArrayList<DomAtrVO>)ListUtil.toVO(listDomAtr,
				new DomAtrVO(-1,
StringUtil.SELECT_OPCION_TODOS)));
```

Luego se cierra la session de Hibernate.
Se devuelve el AtributoSearchPage
Y en caso de ocurrir una Exception, dispara una DemodaServiceException().

---

getAtributoSearchPageResult()

Se setea el userContext.
Abre la session de Hibernate.
Limpia la lista de errores.
Invoca el método findBySearchPage de AtributoDAO para obtener la lista de BO que son el resultado de la búsqueda.

```
	// Aqui obtiene lista de BOs
	List<Atributo> listAtributo =
DefDAOFactory.getAtributoDAO().findBySearchPage(atributoSearchPage);
```

Pasa la lista de BO a VO y setea la lista de resultados VO en la propiedad listResult del SearchPage.

```
//Aqui pasamos BO a VO
atributoSearchPage.setListResult(ListUtil.toVO(listAtributo));
```

Si es necesario se itera la lista de VOs para setear banderas.

```
for (AtributoVO atributoVO:(ArrayList<AtributoVO>)atributoSearchPage.getListResult())
{                       atributoVO.setEliminarBussEnabled(false);
                atributoVO.setModificarBussEnabled(false);
}
```

Si el método findBySearchPage realiza una búsqueda sin contar la cantidad de registros obtenidos (es decir, invoca al método `executeSearch de la clase GenericAbstractDAO de demoda`) y se desea paginar, recalculamos la máxima cantidad de registros del Page utilizando el siguiente método, pasando como parámetro la longuitud de la lista obtenida:

```
// recalculo de la max ctd de registros del page
atributoSearchPage.recalcularMaxRegistros(listAtributo.size());
```

Se cierra la session de Hibernate.
Devuelve el AtributoSearchPage
En caso de ocurrir una Exception, se dispara una DemodaServiceException().

---

getAtributoAdapterForView()

Setea el userContext recibido.
Abre la session de Hibernate.
Invoca el método getById() de Atributo para obtener el Objeto buscado dado su Id obtenido del atributoKey.

```
Atributo atributo = Atributo.getById(atributoKey.getId());
```

Instancia un AtributoAdapter.
Pasa el BO a VO en el nivel adecuado y lo setea en el Adapter.

```
atributoAdapter.setAtributo((AtributoVO) atributo.toVO(3));
```

Cierra la session de Hibernate.
Devuelve el AtributoAdapter
En caso de ocurrir una Exception, dispara una DemodaServiceException().

---

getAtributoAdapterForCreate()

Setea el userContext.
Abre la session de Hibernate.
Instancia un AtributoAdapter.
Si es necesario setea banderas de negocio en el AtributoVO del AtributoAdapter.
Si es necesario recupera, pasa a VO y setea listas en el AtributoAdapter para llenar combos.

```
// Seteo la lista de tipoAtributo
List<TipoAtributo> listTipoAtributo = TipoAtributo.getListActivos();

atributoAdapter.setListTipoAtributo((ArrayList<TipoAtributoVO>)ListUtil.toVO(listTipoAtributo,
                                        new TipoAtributoVO(-1,
StringUtil.SELECT_OPCION_SELECCIONAR)));
```

En caso de listas representadas por enumeraciones, se invoca el método getList() de la Enumeración y las setea en el AtributoAdapter.

```
atributoAdapter.setListSiNo(SiNo.getList(SiNo.OpcionSelecionar));
```

Cierra la session de Hibernate.
Devuelve el AtributoAdapter
En caso de ocurrir una Exception, dispara una DemodaServiceException().

---

getAtributoAdapterParamForCreate()

En caso de ser necesario un param durante la creación del Atributo.
Similar al Param del SearchPage.

---

createAtributo()

Setea el userContext.
Abre la session de Hibernate.
Abre una “transacción” dentro de la session de Hibernate.

```
public AtributoVO createAtributo(UserContext userContext, AtributoVO atributoVO)
throws
                DemodaServiceException {
        String funcName = DemodaUtil.currentMethodName();
        Session session = null;
        Transaction tx = null;

        if (log.isDebugEnabled()) log.debug(funcName + ": enter");
        try {
                DemodaUtil.setCurrentUserContext(userContext);
                session = SiatHibernateUtil.currentSession();
                tx = session.beginTransaction();
```

Limpia la lista de errores y mensajes del AtributoVO recibido.

```
atributoVO.clearErrorMessages();
```

Instancia un Atributo (que es una clase BO) y le carga todos los valores del AtributoVO recibido.

```
Atributo atributo = new Atributo();
atributo.setCodAtributo(atributoVO.getCodAtributo());
atributo.setDesAtributo(atributoVO.getDesAtributo());
atributo.setValorDefecto(atributoVO.getValorDefecto());
atributo.setEsAtributoBusqueda(atributoVO.getEsAtributoBusqueda().getId());
atributo.setAdmBusPorRan(atributoVO.getAdmBusPorRan().getBussId());
atributo.setAdmBusMulVal(atributoVO.getAdmBusMulVal().getBussId());;
```

Si es necesario recupera y setea en el Atributo, objetos de negocio que lo componen.

```
TipoAtributo tipoAtributo = TipoAtributo.getById(atributoVO.getTipoAtributo().getId());
atributo.setTipoAtributo(tipoAtributo);

if (!ModelUtil.isNullOrEmpty(atributoVO.getDomAtr())){
        DomAtr domAtr = DomAtr.getById(atributoVO.getDomAtr().getId());
        atributo.setDomAtr(domAtr);
}

atributo.setEstado(atributoVO.getEstado().getId());
```

Invoca al método createAtributo del DefManager.

```
atributo = DefManager.createAtributo(atributo);
```

Si la lista de errores del atributo no esta vacía:
        Se realiza un rollback sobre la transacción abierta.
Si no:

Se realiza un commit sobre la transacción abierta y se pasa a VO el atributo.
Se pasa la lista de errores y mensajes del atributo al atributoVO.
Cierra la session de Hibernate.
Devuelve el AtributoVO.

```
		if (atributo.hasError()) {
			tx.rollback();
		if(log.isDebugEnabled()){log.debug(funcName + ": tx.rollback");}
		} else {
			tx.commit();
			if(log.isDebugEnabled()){log.debug(funcName + ": tx.commit");}
			atributoVO =  (AtributoVO) atributo.toVO(3);
		}

		atributo.passErrorMessages(atributoVO);

		log.debug(funcName + ": exit");
		return atributoVO;
```

En caso de ocurrir una Exception, se realiza un rollback sobre la transacción abierta y dispara una DemodaServiceException()

```
		} catch (Exception e) {
			log.error(funcName + ": Service Error: ",  e);
			if(tx != null) tx.rollback();
				throw new DemodaServiceException(e);
		} finally {
			SiatHibernateUtil.closeSession();
		}
	}
```

---

getAtributoAdapterForUpdate()

Setea el userContext.
Abre la session de Hibernate.
Invoca el método getById() de Atributo para obtener el Objeto buscado dado su Id.
Instancia un AtributoAdapter.
Si es necesario setea banderas de negocio en el AtributoVO del AtributoAdapter.
Si es necesario recupera, pasa a VO y setea listas en el AtributoAdapter para llenar combos.
En caso de listas representadas por enumeraciones, se invoca el método getList() de la Enumeración y se setea en el AtributoAdapter.
Cierra la session de Hibernate.
Devuelve el AtributoAdapter.
En caso de ocurrir una Exception, dispara una DemodaServiceException().

---

getAtributoAdapterParamForUpdate()

Se crea en caso de ser necesario un param durante la actualización del Atributo.
El código es similar al Param del SearchPage.

---

updateAtributo()

Actualiza los datos del Atributo. El código es similar al del createAtributo(), con la diferencia que se recupera el objeto de negocio a partir de su id, obtenido desde el VO.

---

getAtributoAdapterForDelete()

Se encarga de obtener el Atributo a eliminar, el código es similar al del getAtributoAdapterForView().

---

deleteAtributo()

Se encarga del borrado del Atributo. El código es similar al del updateAtributo().

Test

Prueba del Servicio AtributoServiceHbmImplTest ubicado en la siguiente carpeta:

```
siat/buss/test/ar/gov/rosario/siat/<modulo>/buss/service
```

Es necesario importar la clase TestCase del jar Junit.

```
import junit.framework.TestCase;
```

Extiende de TestCase.
Como estándar definimos que el nombre de la clase de prueba es igual al nombre de clase sobre la que se hace la prueba mas el sufijo “Test”.

Se realiza una prueba sobre cada uno de los métodos definidos en el servicio de Atributo.

```
public class AtributoServiceHbmImplTest extends TestCase {

	private static Logger log =
Logger.getLogger(AtributoServiceHbmImplTest.class);

	public void testMantenedorAtributo() {

	try {

		UserContext userContext = new UserContext();
		userContext.setUserName("test");

		// getAtributoAdapterForCreate
		AtributoAdapter atributoAdapter =
DefServiceLocator.getAtributoService().
	getAtributoAdapterForCreate(userContext);
		// carga de datos y ensambles necesarios
		AtributoVO atributoVO = atributoAdapter.getAtributo();
		atributoVO.setCodAtributo("0000-0001");
		atributoVO.setDesAtributo("desc prueba");
		TipoAtributoVO tipoAtributoVO = (TipoAtributoVO)
TipoAtributo.getById(
                                                                new
Long(1)).toVO(0);
		atributoVO.setTipoAtributo(tipoAtributoVO);

		// createAtributo
		atributoVO =
DefServiceLocator.getAtributoService().createAtributo(userContext,
                                        atributoAdapter.getAtributo());
		// getAtributoSearchPageInit
		AtributoSearchPage atributoSearchPage =
DefServiceLocator.getAtributoService().

	getAtributoSearchPageInit(userContext);
		// getAtributoSearchPageParamTipoAtributo
		atributoSearchPage = DefServiceLocator.getAtributoService().
			getAtributoSearchPageParamTipoAtributo(userContext,
atributoSearchPage);

		// getAtributoSearchPageResult
		atributoSearchPage = DefServiceLocator.getAtributoService().
```

```
                                                    getAtributoSearchPageResult(userContext,
atributoSearchPage);

                                // getAtributoAdapterForView
                                atributoAdapter = DefServiceLocator.getAtributoService().
                                        getAtributoAdapterForView(userContext, new
CommonKey(atributoVO.getId()));

                                // getAtributoAdapterForUpdate
                                atributoAdapter = DefServiceLocator.getAtributoService().
                                    getAtributoAdapterForUpdate(userContext, new
CommonKey(atributoVO.getId()));

                                // updateAtributo
                                atributoVO = DefServiceLocator.getAtributoService().
                                                                updateAtributo(userContext,
atributoAdapter.getAtributo());

                                // deleteAtributo
                                atributoVO = DefServiceLocator.getAtributoService().
                                                                deleteAtributo(userContext,
atributoAdapter.getAtributo());

                        } catch (Exception e) {
                                log.error(e.getMessage());
                                fail("Excepcion en testMantenedorAtributo");
                        }
                }
        }
```

View

En esta sección encontramos los archivos de configuración de struts y tiles, los jsp's, actions y properties organizados de la siguiente forma:

```
/siat/view/src/<modulo>/<submodulo>/
                                        /<bean>SearchPage.jsp
                                        /<bean>Adapter.jsp

/siat/view/src/WEB-INF/<modulo>/config/
                                          /tiles-defs-<modulo>.xml
                                          /struts-config-<modulo>.xml

/siat/view/src/WEB-INF/src/resources/
                                      /<modulo>.properties

/siat/view/src/WEB-INF/src/ar/gov/rosario/siat/<modulo>/view/struts/

/Administrar<Bean>DAction.java

/Buscar<Bean>DAction.java
/siat/view/src/WEB-INF/src/ar/gov/rosario/siat/<modulo>/view/util/

/<Modulo>Constants.java
```

Tiles

Existe un archivo de configuración de Tiles para cada módulo, en la correspondiente carpeta, donde se deberán ingresar las entradas a utilizar en el caso de uso que estamos construyendo.

Para el caso del “Mantenedor de Atributo” el archivo es el siguiente:

```
/siat/view/src/WEB-INF/def/config/tiles-defs-def.xml
```

Acá podemos ver como se le asigna un nombre lógico a una extensión de la plantilla ".siatStdLayout" en la cual se insertará en el atributo "body" el valor del jsp indicado.

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

 <!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration 1.1//EN"
                                                "http://jakarta.apache.org/struts/dtds/tiles-config_1_1.dtd">

<tiles-definitions>
        <!-- Atributo -->
        <definition name=".def.atributos.view.atributoSearchPage" extends=".siatStdLayout">
                <put name="body" value="/def/atributos/atributoSearchPage.jsp"/>
        </definition>

        <definition name=".def.atributos.view.atributoVEAdapter" extends=".siatStdLayout">
                <put name="body" value="/def/atributos/atributoVEAdapter.jsp"/>
        </definition>

        <definition name=".def.atributos.view.atributoAdapter" extends=".siatStdLayout">
                <put name="body" value="/def/atributos/atributoAdapter.jsp"/>
        </definition>
        <!-- Fin Atributo -->
</tiles-definitions>
```

Struts-config

Al igual que con los Tiles, están organizados en carpetas que corresponden a módulos, así como la última parte del nombre del archivo.
Entonces para el modulo definición (def) nos queda el archivo:

```
/siat/view/src/WEB-INF/def/config/struts-config-def.xml
```

En este archivo podemos ver dos secciones muy importantes, la declaración de los “form-beans” y la de los “action-mappings”.
En la primera se definen los DynaActionForms cuyo nombre coincide con los SearchPage y Adapter definidos en el IFace.
En la sección action-mappings tenemos:

path: Es el nombre lógico que se le da al action.
type: Es el nombre completo (sin .class ni .java) de la clase controladora, la cual extiende de BaseDispatchAction.
scope: Es el alcance de los objetos que manejamos y que para siat es "request".
validate ="false": Indica que no utilizamos las validaciones de struts.
name ="atributoSearchPage": Es el nombre del form que usa el action.
parameter ="method": Es el nombre del parámetro que utilizaremos para indicar que método ejecutar dentro del action.
input =".def.atributos.view.atributoSearchPage": Es el forward utilizado en caso de ocurrir errores recuperables.
attribute ="atributoSearchPage": ?????

Luego escribiremos un forward por cada destino utilizado dentro del action para decidir a donde navegar, Ejemplo:

```
<forward name="atributoSearchPage"
path=".def.atributos.view.atributoSearchPage" />
```

Para el caso del “Mantenedor de Atributo” el struts-config quedará de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN" "[WWW]
http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>
  <form-beans>
    <form-bean name="atributoSearchPage"
type="org.apache.struts.action.DynaActionForm"/>
    <form-bean name="atributoAdapter"
type="org.apache.struts.action.DynaActionForm"/>
  </form-beans>

  <action-mappings>
    <!-- Mantenedor Atributo -->
    <action path="/def/BuscarAtributo"
      type="ar.gov.rosario.siat.def.view.struts.BuscarAtributoDAction"
      scope="request"
      validate="false"
      name="atributoSearchPage"
      parameter="method"
      input=".def.atributos.view.atributoSearchPage"
      attribute="atributoSearchPage">

      <forward name="atributoSearchPage"
path=".def.atributos.view.atributoSearchPage" />
      <forward name="administrarAtributo"

path="/def/AdministrarAtributo.do?method=inicializar" />
    </action>
```

```
                <action path="/def/AdministrarAtributo"
                        type="ar.gov.rosario.siat.def.view.struts.AdministrarAtributoDAction"
                        scope="request"
                        validate="false"
                        name="atributoAdapter"
                        parameter="method"
                        input=".def.atributos.view.atributoAdapter"
                        attribute="atributoAdapter">

                            <forward name="atributoVEAdapter"


          path=".def.atributos.view.atributoVEAdapter" />
                            <forward name="atributoAdapter"
path=".def.atributos.view.atributoAdapter" />
                            <forward name="buscarAtributo"   path="/def/BuscarAtributo.do?
method=buscar" />                   </action>
                    <!-- Fin Mantenedor Atributo -->
              </action-mappings>
          </struts-config>
```

Action y JSP

Los Actions están organizados por módulos de misma manera que los archivos de configuración.
Por ejemplo para el “Mantenedor de Atributos”, nos queda de la siguiente manera en:

```
/siat/view/src/WEB-INF/src/ar/gov/rosario/siat/def/view/struts/
```

BuscarAtributoDAction.java

Contiene las acciones de búsqueda, inicializar búsqueda, limpiar, param, buscar y redireccionar al Administrar cuando se desea ver, agregar, modificar, activar, desactivar o cualquier otra acción sobre un elemento del resultado de la búsqueda.
Extiende de BaseDispatchAction

```
        public final class BuscarAtributoDAction extends BaseDispatchAction {
```

Inicializar()

Obtenemos el nombre del método actual.
Con este nombre obtenido, se lo pasamos a la función canAccess() para determinar el permiso de acceso a la acción.
Si no tiene permiso de acceso (userSession == null), redirige a la página de error correspondiente.

```
        public ActionForward inicializar(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

                String funcName = DemodaUtil.currentMethodName();
                if (log.isDebugEnabled()) log.debug(funcName + ": enter");

                UserSession userSession = canAccess(request, mapping, funcName);
                if (userSession == null) return forwardErrorSession(request);
```

Invocamos el Servicio de Atributo y obtenemos el SearchPage para realizar la búsqueda.

Si el SearchPage tiene errores recuperables:
Se logean los errores.
Se Obtienen los ActionMessages y se cargan en el request para que los errores puedan ser mostrados en la jsp.

Redirige al InputForward que visualiza los errores recuperables encontrados.

Si el SearchPage tiene errores NO recuperables:
Se logean los errores.
Y se redirige a una ventana de Confirmación para visualizar el trace del error.

```
		try {
			AtributoSearchPage atributoSearchPageVO =
DefServiceLocator.getAtributoService()
                                                                .getAtributoSearc
hPageInit(userSession);
			// Tiene errores recuperables
			if (atributoSearchPageVO.hasErrorRecoverable()) {
				log.error("recoverable error en: "funcName + ": " +
                                                atributoSearchPageVO.infoString());
				saveDemodaErrors(request, atributoSearchPageVO);
				return forwardErrorRecoverable(mapping, request, userSession,
                                                        AtributoSearchPage.NAME,
atributoSearchPageVO);
			}
			// Tiene errores no recuperables
			if (atributoSearchPageVO.hasErrorNonRecoverable()) {
				log.error("error en: "  + funcName + ": " +
atributoSearchPageVO.errorString());
				return forwardErrorNonRecoverable(mapping, request, funcName,
                                                        AtributoSearchPage.NAME,
atributoSearchPageVO);
			}
```

Si el SearchPage no tiene errores:
Se Invoca al baseInicializarSearchPage para:
Setear los valores de navegación en el PageModel.
Habilitar o deshabilitar las acciones de ABM o Selección.
Enviar el PageModel al request y subirlo al userMap del userSession con el nombre correspondiente.
Y redirigir al forward declarado en el struts-config-def.xml, usando la variable estática declarada en DefConstants.FWD_ATRIBUTO_SEARCHPAGE.

```
		// Si no tiene error
		baseInicializarSearchPage(mapping, request, userSession , AtributoSearchPage.NAME,

                                atributoSearchPageVO);
		return mapping.findForward(DefConstants.FWD_ATRIBUTO_SEARCHPAGE);
```

En caso de producirse una excepción, se ejecuta el baseException para visualizar el stacktrace de la excepcion en una ventana de confirmación.

```
		} catch (Exception exception) {
			return baseException(mapping, request, funcName, exception,
AtributoSearchPage.NAME);
		}
	}
```

---

limpiar()

Se invoca al método baseRefill para:
Setear los valores iniciales en el navModel y llamar de nuevo al método inicializar para que redibuje la jsp.

```
		public ActionForward limpiar(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

			String funcName = DemodaUtil.currentMethodName();
			return this.baseRefill(mapping, form, request, response, funcName,
```

```
        AtributoSearchPage.NAME);
        }
```

---

paramTipoAtributo()

Este método es utilizado para recargar listas, en este caso cuando se cambia el valor de tipoAtributo.

```
        public ActionForward paramTipoAtributo (ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response) throws Exception
{

                String funcName = DemodaUtil.currentMethodName();
                if (log.isDebugEnabled()) log.debug("entrando en " + funcName);
                UserSession userSession = canAccess(request, mapping, funcName);
                if (userSession==null) return forwardErrorSession(request);

                try {
```

Se obtiene el SearchPage del userSession:
Si es nulo:
Logear el error y
Redirige a una Ventana de Error.

```
        // Bajo el searchPage del userSession
        AtributoSearchPage atributoSearchPageVO = (AtributoSearchPage)


        userSession.get(AtributoSearchPage.NAME);
        // Si es nulo no se puede continuar
        if (atributoSearchPageVO == null) {
                log.error("error en: "  + funcName + ": " + AtributoASearchPage.NAME + " IS
NULL.                                                                   No se pudo
obtener de la sesion");
                return forwardErrorSessionNullObject(mapping, request, funcName,


        AtributoSearchPage.NAME);
        }
```

Cargamos el atributoSearchPageVO con los parámetros del request.(Si encuentra un error, carga la lista de errores del atributoSearchPageVO).
Manejo de errores recuperables.

```
        // Recuperamos datos del form en el vo
        DemodaUtil.populateVO(atributoSearchPageVO, request);

        // Tiene errores recuperables
        if (atributoSearchPageVO.hasErrorRecoverable()) {
                log.error("recoverable error en: "  + funcName + ": " +

        atributoSearchPageVO.infoString());
                saveDemodaErrors(request, atributoSearchPageVO);
                return forwardErrorRecoverable(mapping, request, userSession,
                                                        AtributoSearchPage.NAME,
atributoSearchPageVO);
        }
```

Llamada al método getAtributoSearchPageParamTipoAtributo() del AtributoService.
Manejo de errores recuperables.
Manejo de errores No recuperables.

```
        // llamada al servicio
        atributoSearchPageVO = DefServiceLocator.getAtributoService().
                        getAtributoSearchPageParamTipoAtributo(userSession,
atributoSearchPageVO);
        // Tiene errores recuperables
```

```
		if (atributoSearchPageVO.hasErrorRecoverable()) {
			log.error("recoverable error en: "  + funcName + ": " +
atributoSearchPageVO.infoString());
			saveDemodaErrors(request, atributoSearchPageVO);
			return forwardErrorRecoverable(mapping, request, userSession,
									AtributoSearchPage.NAME,
atributoSearchPageVO);
		}
		// Tiene errores no recuperables
		if (atributoSearchPageVO.hasErrorNonRecoverable()) {
			log.error("error en: "  + funcName + ": " +
atributoSearchPageVO.errorString());
			return forwardErrorNonRecoverable(mapping, request, funcName,
AtributoSearchPage.NAME,
			atributoSearchPageVO);
		}
```

Carga del VO al request, utilizando AtributoSearchPage.NAME como clave.
Subimos el apdater al userSession
Redirige al forward declarado en el struts-config-def.xml, usando la variable estática declarada en DefConstants.FWD_ATRIBUTO_SEARCHPAGE.

```
		// Envio el VO al request
		request.setAttribute(AtributoSearchPage.NAME, atributoSearchPageVO);
		// Subo el apdater al userSession
		userSession.put(AtributoSearchPage.NAME, atributoSearchPageVO);

		return mapping.findForward(DefConstants.FWD_ATRIBUTO_SEARCHPAGE);

		} catch (Exception exception) {
			return baseException(mapping, request, funcName, exception,
AtributoSearchPage.NAME);
		}
 }
```

---

buscar()

Realiza la búsqueda de Atributos.

Determinamos el permiso de acceso.
Bajamos el SearchPage del userSession.
El método buscar puede ser disparado desde la misma página de búsqueda o de cualquier otra página, de acuerdo al valor del “reqAttIsSubmittedForm” (que es un elemento del userMap del userSession).
Si el buscar fue disparado desde la página de búsqueda dado por reqAttIsSubmittedForm tiene valor “true”:
Se Carga el atributoSearchPageVO con los parámetros del request.
Se setea el PageNumber del PageModel con el reqAttPageNumber del userSession.
El PageNumber indica el nro de página de los resultados de la búsqueda.
Se realiza el manejo de errores recuperables porque la ejecución del populateVO realiza validaciones de formatos de tipos de datos que pueden dar lugar a carga de errores.

```
		public ActionForward buscar(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{

			String funcName = DemodaUtil.currentMethodName();
			if (log.isDebugEnabled()) log.debug(funcName + ": enter");
			UserSession userSession = canAccess(request, mapping, funcName);
			if (userSession == null) return forwardErrorSession(request);

			try {
				// Bajo el searchPage del userSession
				AtributoSearchPage atributoSearchPageVO = (AtributoSearchPage)

		userSession.get(AtributoSearchPage.NAME);
```

```
                        // Si es nulo no se puede continuar
                        if (atributoSearchPageVO == null) {
                                log.error("error en: "  + funcName + ": " +
AtributoSearchPage.NAME +
                                                " IS NULL. No se pudo obtener de
la sesion");
                                return forwardErrorSessionNullObject(mapping, request,
funcName,
        AtributoSearchPage.NAME);
                        }
                        // si el buscar diparado desde la pagina de busqueda
                        if (((String)userSession.get("reqAttIsSubmittedForm")).equals("true"))
{
                                // Recuperamos datos del form en el vo
                                DemodaUtil.populateVO(atributoSearchPageVO, request);
                                // Setea el PageNumber del PageModel
                                atributoSearchPageVO.setPageNumber(new

        Long((String)userSession.get("reqAttPageNumber")));
                        }
                        // Tiene errores recuperables
                        if (atributoSearchPageVO.hasErrorRecoverable()) {
                                log.error("recoverable error en: "  + funcName + ": " +

        atributoSearchPageVO.infoString());
                                saveDemodaErrors(request, atributoSearchPageVO);
                                return forwardErrorRecoverable(mapping, request,
userSession,
        AtributoSearchPage.NAME, atributoSearchPageVO);
                        }
```

Se llama al método getAtributoSearchPageResult del servicio AtributoService para obtener la lista de resultados.
Se realiza el manejo de errores recuperables porque la ejecución del servicio valida datos requeridos y reglas de negocio.
Se realiza manejo de errores No recuperables cargados en la ejecución del servicio.

Enviamos el atributoSearchPageVO al request, para renderizar sus datos en la jsp.
Vaciamos la propiedad listResult del atributoSearchPageVO para reducir el tamaño de este objeto en sesión.
Subimos el SearchPage al userSession para posteriormente tenerlo disponible.
Redirigir al forward declarado en el struts-config-def.xml, usando la variable estática declarada en DefConstants.FWD_ATRIBUTO_SEARCHPAGE.

```
        // Llamada al servicio
        atributoSearchPageVO = DefServiceLocator.getAtributoService().
                                        getAtributoSearchPageResult(userSession,
atributoSearchPageVO);
        // Tiene errores recuperables
        if (atributoSearchPageVO.hasErrorRecoverable()) { log.error("recoverable error en: "  +
funcName + ": "                                                                 +
atributoSearchPageVO.infoString());
                saveDemodaErrors(request, atributoSearchPageVO);
                return forwardErrorRecoverable(mapping, request, userSession,
                                                        AtributoSearchPage.NAME,
atributoSearchPageVO);
                        }
        // Tiene errores no recuperables
        if (atributoSearchPageVO.hasErrorNonRecoverable()) {
                log.error("error en: "  + funcName + ": " +
atributoSearchPageVO.errorString());
                return forwardErrorNonRecoverable(mapping, request, funcName,
                                                        AtributoSearchPage.NAME,
atributoSearchPageVO);
        }
        // Envio el VO al request
        request.setAttribute(AtributoSearchPage.NAME, atributoSearchPageVO);
```

```
		// Vacia el list result
		atributoSearchPageVO.setListResult(new ArrayList());
		// Subo en el el searchPage al userSession
		userSession.put(AtributoSearchPage.NAME, atributoSearchPageVO);

		return mapping.findForward(DefConstants.FWD_ATRIBUTO_SEARCHPAGE);

		} catch (Exception exception) {
			return baseException(mapping, request, funcName, exception,
AtributoSearchPage.NAME);
		}
	}
```

---

ver()

Invocamos el método forwardVerSearchPage para forwardear al método inicializar del AdministrarAtributoDAction y redirigirnos a la pantalla de visualización del Atributo seleccionado.

```
		public ActionForward ver(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{

			String funcName = DemodaUtil.currentMethodName();
			return forwardVerSearchPage(mapping, request,

		funcName,DefConstants.ACTION_ADMINISTRAR_ATRIBUTO);
		}
```

---

agregar()

Invocamos el método forwardAgregarSearchPage para forwardear al método inicializar del AdministrarAtributoDAction y redirigirnos a la pantalla para agregar un nuevo Atributo.

```
		public ActionForward agregar(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{

			String funcName = DemodaUtil.currentMethodName();
			return forwardAgregarSearchPage(mapping, request, funcName,

		DefConstants.ACTION_ADMINISTRAR_ATRIBUTO);
		}
```

---

modificar()

Invocamos el método forwardModificarSearchPage para forwardear al método inicializar del AdministrarAtributoDAction y redirigirnos a la pantalla de modificación del Atributo seleccionado.

```
		public ActionForward modificar(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{

			String funcName = DemodaUtil.currentMethodName();
			return forwardModificarSearchPage(mapping, request, funcName,

		DefConstants.ACTION_ADMINISTRAR_ATRIBUTO);
		}
```

---

eliminar()

Invocamos el método forwardEliminarSearchPage para forwardear al método inicializar del AdministrarAtributoDAction y redirigirnos a la pantalla de eliminación del Atributo seleccionado.

```
	public ActionForward eliminar(ActionMapping mapping, ActionForm form,
		HttpServletRequest request, HttpServletResponse response) throws Exception
{

		String funcName = DemodaUtil.currentMethodName();
		return forwardEliminarSearchPage(mapping, request, funcName,

	DefConstants.ACTION_ADMINISTRAR_ATRIBUTO);
	}
```

---

seleccionar()

Forwardea a la acción y método definidos para seleccionar.

```
	public ActionForward seleccionar(ActionMapping mapping, ActionForm form,
		HttpServletRequest request, HttpServletResponse response)
		throws Exception {

		String funcName = DemodaUtil.currentMethodName();
		return baseSeleccionar(mapping, request, response, funcName,
AtributoSearchPage.NAME);
	}
```

---

activar() y desactivar()

Se Invoca al método forwardActivarSearchPage / forwardDesactivarSearchPage para forwardear al método inicializar del AdministrarAtributoDAction y redirigirnos a la pantalla de activación /desactivación del Atributo seleccionado.

```
	public ActionForward activar(ActionMapping mapping, ActionForm form,
		HttpServletRequest request, HttpServletResponse response) throws Exception
{

		String funcName = DemodaUtil.currentMethodName();
		return forwardActivarSearchPage(mapping, request, funcName,

	DefConstants.ACTION_ADMINISTRAR_ATRIBUTO);
	}
```

---

volver()

Invocamos al método baseVolver para:
regresar a donde indique el método getAccionVolver() del SearchPage ,
y remover el model del userSession.

```
	public ActionForward volver(ActionMapping mapping, ActionForm form,
		HttpServletRequest request, HttpServletResponse response)
		throws Exception {

		return baseVolver(mapping, form, request, response,
AtributoSearchPage.NAME);
	}
```

---

duplicar()

Es un ejemplo de otra Acción sobre un elemento de la grilla de resultado: Invocamos al método baseForwardSearchPage para redirigir al forward definido en DefConstants.ACT_DUPLICAR.

```
	public ActionForward duplicar(ActionMapping mapping, ActionForm form,
		HttpServletRequest request, HttpServletResponse response) throws Exception
{

		String funcName = DemodaUtil.currentMethodName();
		return baseForwardSearchPage(mapping, request, funcName,
```

```
                                DefConstants.ACTION_ADMINISTRAR_ATRIBUTO,
DefConstants.ACT_DUPLICAR );
        }
```

atributoSearchPage.jsp

De la misma manera que los actions, que están organizados por módulos, las jsp's se encuentran organizadas además por submódulos. Para el caso de Atributo, que está dentro del submódulo “atributos”, las jsp's se ubican en la siguiente carpeta:

```
/siat/view/src/def/atributos/
```

Es la página de búsqueda de Atributo.

Declaración de Tags utilizados en el jsp.
Inclusión del archivo/s javascript utilizados en la página.

```
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>

<script type="text/javascript" language="javascript">
   <%@include file="/base/submitForm.js"%>
</script>
```

Formulario de la página con la propiedad styleId="filter" y action= path declarado en el struts-config y de acuerdo al url-pattern *.do declarado en el web.xml.

```
<html:form styleId="filter" action="/def/BuscarAtributo.do">
```

Mensajes y/o Advertencias: visualiza la lista de errores y mensajes en el formato declarado en la hoja de estilo.

```
<!-- Mensajes y/o Advertencias -->
<%@ include file="/base/warning.jsp" %>
<!-- Errors  -->
<html:errors bundle="base"/>
```

Sección de Títulos y Filtros de la página, que incluyen al botón Limpiar y Buscar:

```
<h1><bean:message bundle="def" key="def.atributoSearchPage.title"/></h1>
<p><bean:message bundle="def" key="def.atributoSearchPage.legend"/></p>
<!-- Filtro -->
<fieldset>
  <legend><bean:message bundle="base"
key="base.parametrosBusqueda"/></legend>
    <table class="tabladatos">
          <tr>
            <td><label><bean:message bundle="def"
key="def.atributo.codAtributo.label"/>: </label>
            </td>
            <td class="normal"><html:text name="atributoSearchPageVO"
property="codAtributo"                  size="15" maxlength="20" styleClass="datos"
onchange="limpiaResultadoFiltro();"/>
            </td>
            <td><label><bean:message bundle="def"
key="def.atributo.desAtributo.label"/>: </label>
            </td>
            <td class="normal"><html:text name="atributoSearchPageVO"
property="desAtributo"                                                  size="20"
maxlength="100" styleClass="datos"

        onchange="limpiaResultadoFiltro();"/>
            </td>
```

```
                    </tr>
                </table>
                <p align="center">
                    <html:button property="btnLimpiar"  styleClass="boton"
onclick="submitForm('limpiar', '');">
                        <bean:message bundle="base" key="abm.button.limpiar"/>
                    </html:button>
                     
                    <html:button property="btnBuscar"  styleClass="boton"
onclick="submitForm('buscar', '');">
                        <bean:message bundle="base" key="abm.button.buscar"/>
                    </html:button>
                </p>
            </fieldset>
            <!-- Fin Filtro -->
```

Sección del Resultado de la búsqueda.
Si existe resultado de búsqueda Iteramos sobre el mismo.

```
            <!-- Resultado Filtro -->
            <div id="resultadoFiltro">
            <logic:notEmpty  name="atributoSearchPageVO" property="listResult">
                    <table class="tramonline" border="0" cellpadding="0" cellspacing="1"
width="100%">
                    <caption><bean:message bundle="base" key="base.resultadoBusqueda"/>
                    </caption>
                    <tbody>
                    <tr>
                            <th width="1"> </th> <!-- Ver/Seleccionar -->
                            <th width="1"> </th> <!-- Modificar -->
                            <th width="1"> </th> <!-- Eliminar -->
                            <th width="1"> </th> <!-- Otra Accion -->
                            <th align="left"><bean:message bundle="def"

            key="def.atributo.codAtributo.label"/>
                            </th>
                            <th align="left"><bean:message bundle="def"

            key="def.atributo.desAtributo.label"/>
                            </th>
                            <th align="left"><bean:message bundle="def"

            key="def.tipoAtributo.desTipoAtributo.ref"/>
                            </th>
                    </tr>
                    <logic:iterate id="AtributoVO" name="atributoSearchPageVO"
property="listResult">
                    <tr>
```

Existe una bandera modoSeleccionar del atributoSearchPageVO que indica si estamos en modo de Selección o en modo de Búsqueda con todas las acciones habilitadas.
De acuerdo a este modo se muestra la acción "Ver" o la acción "Seleccionar".

```
            <!-- Ver/Seleccionar -->
            <td>
            <logic:notEqual name="atributoSearchPageVO" property="modoSeleccionar"
value="true">
                    <a style="cursor: pointer; cursor: hand;" onclick="submitForm('ver',
'<bean:write                                    name="AtributoVO" property="id"
                                bundle="base" formatKey="general.format.id"/>');">
                            <img title="<bean:message bundle="base"
key="abm.button.ver"/>" border="0"                                      src="<
%=request.getContextPath()%>/images/iconos/selec0.gif"/>
                    </a>
                    </logic:notEqual>
                    <logic:equal name="atributoSearchPageVO" property="modoSeleccionar"
value="true">
                    <a style="cursor: pointer; cursor: hand;" onclick="submitForm('seleccionar',
'<bean:write                            name="AtributoVO" property="id"
```

```
                                bundle="base" formatKey="general.format.id"/>');">
                                <img title="<bean:message bundle="base"
key="abm.button.seleccionar"/>"                         border="0" src="<
%=request.getContextPath()%>/images/iconos/selec0.gif"/>
                        </a>
                        </logic:equal>
                </td>
```

Existe una bandera "ABMEnabled" del atributoSearchPageVO que indica si estamos en modo de Búsqueda con todas las acciones habilitadas.
Además Existe una bandera "modificarEnabled" del AtributoVO que indica si el usuario tiene permiso para modificar el elemento de la grilla.
En caso que tenga permiso: se muestra el botón habilitado, sino se muestra el botón deshabilitado.
Lo mismo ocurre con eliminar, activar, desactivar y otras acciones que se realizan sobre los elementos de la grilla.

```
                <!-- Modificar-->
                <td>
                <logic:equal name="atributoSearchPageVO" property="ABMEnabled"
value="enabled">
                        <logic:equal name="AtributoVO" property="modificarEnabled"
value="enabled">
                                <a style="cursor: pointer; cursor: hand;"
onclick="submitForm('modificar',
                '<bean:write name="AtributoVO" property="id"
                                                        bundle="base"
formatKey="general.format.id"/>');">
                                        <img title="<bean:message bundle="base"
key="abm.button.modificar"/>"                           border="0" src="<
%=request.getContextPath()%>/images/iconos/modif0.gif"/>
                                </a>
                                </logic:equal>
                                <logic:notEqual name="AtributoVO" property="modificarEnabled"
value="enabled">
                                        <img border="0"
                                                        src="<%=request.getContextPath()
%>/images/iconos/modif1.gif"/>
                                </logic:notEqual>
                        </logic:equal>
                        <logic:notEqual name="atributoSearchPageVO" property="ABMEnabled"
value="enabled">
                                 
                        </logic:notEqual>
                </td>
                <!-- Eliminar-->
                <td>
                <logic:equal name="atributoSearchPageVO" property="ABMEnabled"
value="enabled">
                        <logic:equal name="AtributoVO" property="eliminarEnabled"
value="enabled">
                                <a style="cursor: pointer; cursor: hand;"
onclick="submitForm('eliminar',
                '<bean:write name="AtributoVO" property="id"
                                                        bundle="base"
formatKey="general.format.id"/>');">
                                <img title="<bean:message bundle="base"
key="abm.button.eliminar"/>"                                    border="0" src="<
%=request.getContextPath()%>/images/iconos/eliminar0.gif"/>
                                </a>
                        </logic:equal>
                <logic:notEqual name="AtributoVO" property="eliminarEnabled" value="enabled">
                        <img border="0" src="<%=request.getContextPath()
%>/images/iconos/eliminar1.gif"/>
                </logic:notEqual>
                </logic:equal>
                <logic:notEqual name="atributoSearchPageVO" property="ABMEnabled"
value="enabled">
                         
                </logic:notEqual>
                </td>
                <!-- Duplicar-->
```

```
        <td>
                <logic:equal name="atributoSearchPageVO" property="ABMEnabled"
value="enabled">
                <logic:equal name="AtributoVO" property="modificarEnabled"
value="enabled">
                        <a style="cursor: pointer; cursor: hand;"
onclick="submitForm('duplicar',
        '<bean:write name="AtributoVO" property="id"
                                                bundle="base"
formatKey="general.format.id"/>');">
                        <img title="<bean:message bundle="base"
key="abm.button.duplicar"/>"                                    border="0"
                                src="<%=request.getContextPath()
%>/images/iconos/duplicar0.gif"/>
                        </a>
                </logic:equal>
                <logic:notEqual name="AtributoVO" property="modificarEnabled"
value="enabled">
                        <img border="0"
                                        src="<%=request.getContextPath()
%>/images/iconos/duplicar1.gif"/>
                </logic:notEqual>
                </logic:equal>
                <logic:notEqual name="atributoSearchPageVO" property="ABMEnabled"
value="enabled">
                         
                </logic:notEqual>
        </td>
```

Columnas de información de los resultados de la búsqueda:

```
        <td><bean:write name="AtributoVO" property="codAtributo"/></td>
        <td><bean:write name="AtributoVO" property="desAtributo" /></td>
        <td><bean:write name="AtributoVO" property="tipoAtributo.desTipoAtributo"
/></td>
        </tr>
      </logic:iterate>
```

Paginador del resultado de búsqueda:

```
        <tr>
                <td class="paginador" align="center" colspan="20">
                        <bean:define id="pager" name="atributoSearchPageVO"/>
                        <%@ include file="/base/pager.jsp" %>
                </td>
        </tr>
      </tbody>
    </table>
  </logic:notEmpty>
```

En caso que estemos paginando sin realizar el conteo del resultado de la búsqueda utilizamos el include pagerSinPU.jsp en vez del anterior:

```
        <tr>
                <td class="paginador" align="center" colspan="20">
                        <bean:define id="pager" name="personaSearchPageVO"/>
                        <%@ include file="/base/pagerSinPU.jsp" %>
                </td>
        </tr>
```

Si el resultado de búsqueda es vacío: se muestra el mensaje declarado en /siat/view/src/WEB-INF/classes/resources/base.properties , donde se encuentran todos los mensajes genéricos.

```
        <logic:empty name="atributoSearchPageVO" property="listResult">
        <logic:equal name="userSession" property="metodoActual" value="buscar">
                <table class="tramonline" border="0" cellpadding="0" cellspacing="1"
                        width="100%">
                  <caption>
                     <bean:message bundle="base" key="base.resultadoBusqueda"/>
                  </caption>
                  <tbody>
```

```
                        <tr>
                            <td align="center">
                                <bean:message bundle="base" key="base.resultadoVacio"/>
                            </td>
                        </tr>
                    </tbody>
                    </table>
            </logic:equal>
            </logic:empty>
            </div>
            <!-- Fin Resultado Filtro -->
```

Si el resultado de búsqueda es vacío y no estamos usando el conteo utilizamos los siguientes tags logics en lugar del anterior.
El pageNumber igual o mayor a uno significa que estamos en una página que no contiene resultados, por lo cual vamos a tener habilitado regresar a la página anterior.
El pageNumber menor que uno significa que no existen resultados de busqueda para la consulta realizada:

```
            <logic:greaterThan name="personaSearchPageVO" property="pageNumber"
                    value="1">
                    <tr>
                        <td align="center">
                            <bean:message bundle="base"
key="base.resultadoPaginaVacio"/>
                        </td>
                    </tr>
                    <tr>
                        <td class="paginador" align="center" colspan="20">
                            <bean:define id="pager" name="personaSearchPageVO"/>
                            <%@ include file="/base/pagerSinPU.jsp" %>
                        </td>
                    </tr>
            </logic:greaterThan>
            <logic:lessEqual name="personaSearchPageVO" property="pageNumber" value="1">
                    <tr>
                        <td align="center">
                            <bean:message bundle="base" key="base.resultadoVacio"/>
                        </td>
                    </tr>
            </logic:lessEqual>
```

Botones Volver y Agregar:
En caso de que el "metodoActual" es "buscar" y la propiedad "agregarEnabled" esté habilitada, se visualiza el botón Agregar.

```
            <table class="tablabotones">
                    <tr>
                            <td align="left">
                            <html:button property="btnVolver"  styleClass="boton"

            onclick="submitForm('volver', '');">
                                    <bean:message bundle="base" key="abm.button.volver"/>
                            </html:button>
                            </td>
                            <logic:equal name="userSession" property="metodoActual"
value="buscar">
                                    <td align="right">
                                            <bean:define id="agregarEnabled"
name="atributoSearchPageVO"
            property="agregarEnabled"/>
                                            <input type="button" <%=agregarEnabled%>
class="boton"
                                                    onClick="submitForm('agregar', '0');"
                                                    value="<bean:message bundle="base"

            key="abm.button.agregar"/>"
                                            />
                                    </td>
                            </logic:equal>
```

```
                </tr>
        </table>
```

Sección de propiedades hidden:
method: método que se ejecutó para visualizar la jsp.
selectedId: id del elemento seleccionado.
pageNumber: número de página del conjunto de resultados utilizado en la paginación.
isSubmittedForm: bandera que indica que la acción a disparar se llama desde un submit desde el browser y no desde otro action o desde otro página.
Recordar que en el método buscar del action BuscarAtributoDAction, se llama al populateVO condicionalmente según el valor tomado de esta bandera.

```
                <input type="hidden" name="method" value=""/>
                <input type="hidden" name="selectedId" value=""/>
                <input type="hidden" name="pageNumber" value="1" id="pageNumber">
                <input type="hidden" name="isSubmittedForm" value="true"/>

        </html:form>
        <!-- Fin Formulario -->
```

AdministrarAtributoDAction.java

Contiene las acciones para: inicializar abm, insertar, borrar, actualizar un Atributo, paramTipoAtributo y la acción volver.

```
        public final class AdministrarAtributoDAction extends BaseDispatchAction {

                private Log log = LogFactory.getLog(AdministrarAtributoDAction.class);
```

---

inicializar()

Inicializa las acciones que se realizan sobre el Atributo, dependiendo del valor de la propiedad “act” del navmodel.

Se obtiene el nombre del método actual.
Se realiza el Logeo.
Se determina el permiso de acceso.

```
        public ActionForward inicializar(ActionMapping mapping, ActionForm form,
                        HttpServletRequest request, HttpServletResponse response) throws
Exception {
                String funcName = DemodaUtil.currentMethodName();
                if (log.isDebugEnabled()) log.debug("entrando en " + funcName);

                UserSession userSession = canAccess(request, mapping, funcName);
                if (userSession == null) return forwardErrorSession(request);
```

Se obtiene del NavModel y se inicializan las variables.

```
        NavModel navModel = userSession.getNavModel();
        AtributoAdapter atributoAdapterVO = null;
        String stringServicio = "";
        ActionForward actionForward = null;
```

Construcción del CommonKey a partir del id del objeto seleccionado.

```
        try {
                CommonKey commonKey = new CommonKey(navModel.getSelectedId());
```

Dependiendo del Act del Nav Model, se define qué servicio ejecutar y a dónde redirigir la navegación.

```
		if (navModel.getAct().equals(BaseConstants.ACT_VER)) {
			stringServicio	= "getAtributoAdapterForView(userSession, commonKey)";
			atributoAdapterVO =
DefServiceLocator.getAtributoService().getAtributoAdapterForView(

userSession, commonKey);
			actionForward =
mapping.findForward(DefConstants.FWD_ATRIBUTO_VE_ADAPTER);
		}
		if (navModel.getAct().equals(BaseConstants.ACT_MODIFICAR)) {
			stringServicio	= "getAtributoAdapterForUpdate(userSession, commonKey)";
			atributoAdapterVO =
DefServiceLocator.getAtributoService().getAtributoAdapterForUpdate(

userSession, commonKey);
			actionForward =
mapping.findForward(DefConstants.FWD_ATRIBUTO_ADAPTER);
		}
		if (navModel.getAct().equals(BaseConstants.ACT_ELIMINAR)) {
			stringServicio	= "getAtributoAdapterForDelete(userSession, commonKey)";
			atributoAdapterVO =
DefServiceLocator.getAtributoService().getAtributoAdapterForView(

userSession, commonKey);
			actionForward =
mapping.findForward(DefConstants.FWD_ATRIBUTO_VE_ADAPTER);
}
		if (navModel.getAct().equals(BaseConstants.ACT_AGREGAR)) {
			stringServicio	= "getAtributoAdapterForCreate(userSession)";
			atributoAdapterVO =
DefServiceLocator.getAtributoService().getAtributoAdapterForCreate(

			  userSession);
			actionForward =
mapping.findForward(DefConstants.FWD_ATRIBUTO_ADAPTER);
		}
		if (navModel.getAct().equals(DefConstants.ACT_DUPLICAR)) {
			stringServicio	= "getAtributoAdapterForDuplicate(userSession,
commonKey)";
			atributoAdapterVO =
DefServiceLocator.getAtributoService().getAtributoAdapterForDuplicate(
                                                                        userSession,
commonKey);
			actionForward =
mapping.findForward(DefConstants.FWD_ATRIBUTO_ADAPTER);
		}
		if (log.isDebugEnabled()) log.debug(funcName + " salimos de servicio: " +
stringServicio );
		// Nunca Tiene errores recuperables
```

Manejo de errores No recuperables.

```
		// Tiene errores no recuperables
		if (atributoAdapterVO.hasErrorNonRecoverable()) {
			log.error("error en: "  + funcName + ": servicio: " + stringServicio + ": " +

atributoAdapterVO.errorString());
			return forwardErrorNonRecoverable(mapping, request, funcNAME,
AtributoAdapter.NAME,
                                                atributoAdapterVO);
		}
```

Se cargan en el Adapter los valores de navegación contenidos en el NavModel, para posteriormente usarlos por el action al ejecutar la acción Volver.

```
		// Seteo los valores de navegacion en el adapter
		atributoAdapterVO.setValuesFromNavModel(navModel);

		if (log.isDebugEnabled()) log.debug(
			funcName + ": " + AtributoAdapter.NAME + ": " +
atributoAdapterVO.infoString());
```

Carga del atributoAdapterVO en el request para que esté disponible cuando se renderiza la jsp.

```
		// Envio el VO al request
		request.setAttribute(AtributoAdapter.NAME, atributoAdapterVO);
```

Se coloca el atributoAdapterVO en el userMap del userSession, con clave definida por AtributoAdapter.NAME.

```
		// Subo el apdater al userSession
		userSession.put(AtributoAdapter.NAME, atributoAdapterVO);
```

Carga de mensajes a nivel de request para que estén disponibles cuando se renderiza la jsp.

```
		saveDemodaMessages(request, atributoAdapterVO);
```

Se redirige al forward determinado por “actionForward” que se obtuvo previamente a partir del act.

```
		return actionForward;
```

En caso de producirse una excepción, ejecutamos el baseException para visualizar el stacktrace de la excepción en una ventana de confirmación.

```
		 } catch (Exception exception) {
			return baseException(mapping, request, funcName, exception,
AtributoAdapter.NAME);
		}
	}
```

---

insertar()

Inserta un nuevo Atributo.

```
		public ActionForward insertar(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{
```

Obtención del nombre del método actual. Logeo.
Determinación del permiso de acceso.

```
		String funcName = DemodaUtil.currentMethodName();
		if (log.isDebugEnabled()) log.debug("entrando en " + funcName);

		UserSession userSession = canAccess(request, mapping, funcName);
		if (userSession == null) return forwardErrorSession(request);
```

Obtención del atributoAdapterVO desde el userSession, usando como clave el valor de AtributoAdapter.NAME.
Si no se puede obtener (atributoAdapterVO == null): Se logea el error y se redirige a una Ventana que muestra el Error.

```
		try {
			// Bajo el adapter del userSession
			AtributoAdapter atributoAdapterVO = (AtributoAdapter)

		userSession.get(AtributoAdapter.NAME);

			// Si es nulo no se puede continuar
			if (atributoAdapterVO == null) {
				log.error("error en: "  + funcName + ": " + AtributoAdapter.NAME +
									" IS NULL. No se pudo obtener de
la sesion");
				return forwardErrorSessionNullObject(mapping, request, funcName,
```

```
AtributoAdapter.NAME);
		}
```

Carga de datos submitidos en el request al atributoAdapterVO y validaciones de tipos, en la ejecución del método populateVO.
Manejo de errores recuperables, que el populateVO pudo haber cargado en el atributoAdapterVO.

```
	// Recuperamos datos del form en el vo
	DemodaUtil.populateVO(atributoAdapterVO, request);

	// Tiene errores recuperables
	if (atributoAdapterVO.hasErrorRecoverable()) {
		log.error("recoverable error en: "  + funcName + ": " +
atributoAdapterVO.infoString());
		saveDemodaErrors(request, atributoAdapterVO);
		return forwardErrorRecoverable(mapping, request, userSession,
AtributoAdapter.NAME,
		    atributoAdapterVO);
	}
```

Llamada al Servicio.
Manejo de errores recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.
Manejo de errores No recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.

```
	// llamada al servicio
	AtributoVO atributoVO =
DefServiceLocator.getAtributoService().createAtributo(userSession,
                                                       atributoAdapterVO.getAtributo());

	// Tiene errores recuperables
	if (atributoVO.hasErrorRecoverable()) {
		log.error("recoverable error en: "  + funcName + ": " +
atributoVO.infoString());
		saveDemodaErrors(request, atributoVO);
		return forwardErrorRecoverable(mapping, request, userSession,
AtributoAdapter.NAME,

	atributoAdapterVO);
	}

	// Tiene errores no recuperables
	if (atributoVO.hasErrorNonRecoverable()) {
		log.error("error en: "  + funcName + ": " + atributoVO.errorString());
		return forwardErrorNonRecoverable(mapping, request, funcName,
AtributoAdapter.NAME,
	atributoAdapterVO);
	}
```

Redirección a la Página de Confirmación de la acción ejecutada.

```
	// Fue Exitoso
	return forwardConfirmarOk(mapping, request, funcName, AtributoAdapter.NAME);
```

En caso de producirse una excepción, ejecutamos el baseException para visualizar el stacktrace de la excepción en una ventana de confirmación.

```
	} catch (Exception exception) {
		return baseException(mapping, request, funcName, exception,
AtributoAdapter.NAME);
	}
    }
```

---

actualizar()

Actualiza el atributo.

```
	public ActionForward actualizar(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws
Exception {
```

Obtención del nombre del método actual. Logeo.
Determinación del permiso de acceso.

```
	String funcName = DemodaUtil.currentMethodName();
	if (log.isDebugEnabled()) log.debug("entrando en " + funcName);

	UserSession userSession = canAccess(request, mapping, funcName);
	if (userSession == null) return forwardErrorSession(request);
```

Obtención del atributoAdapterVO desde el userSession, usando como clave el AtributoAdapter.NAME.
Si no se puede obtener (atributoAdapterVO == null): Logea el error y redirige a una Ventana que muestra el Error.

```
	try {
		// Bajo el adapter del userSession
		AtributoAdapter atributoAdapterVO = (AtributoAdapter)

	userSession.get(AtributoAdapter.NAME);

		// Si es nulo no se puede continuar
		if (atributoAdapterVO == null) {
			log.error("error en: "  + funcName + ": " + AtributoAdapter.NAME +
				" IS NULL. No se pudo obtener de
la sesion");
			return forwardErrorSessionNullObject(mapping, request, funcName,

	AtributoAdapter.NAME);
		}
```

Carga de datos submitidos en el request al atributoAdapterVO y validaciones de tipos, en la ejecución del método populateVO.
Manejo de errores recuperables, que el populateVO pudo haber cargado en el atributoAdapterVO.

```
	// Recuperamos datos del form en el vo
	DemodaUtil.populateVO(atributoAdapterVO, request);

	// Tiene errores recuperables
	if (atributoAdapterVO.hasErrorRecoverable()) {
		log.error("recoverable error en: "  + funcName + ": " +
atributoAdapterVO.infoString());
		saveDemodaErrors(request, atributoAdapterVO);
		return forwardErrorRecoverable(mapping, request, userSession,
AtributoAdapter.NAME,
		   atributoAdapterVO);
	}
```

Llamada al Servicio.
Manejo de errores recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.
Manejo de errores No recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.

```
	// llamada al servicio
	AtributoVO atributoVO = DefServiceLocator.getAtributoService()
		.updateAtributo(userSession, atributoAdapterVO.getAtributo());

	// Tiene errores recuperables
	if (atributoVO.hasErrorRecoverable()) {
		log.error("recoverable error en: "  + funcName + ": " +
atributoAdapterVO.infoString());
		saveDemodaErrors(request, atributoVO);
```

```
                return forwardErrorRecoverable(mapping, request, userSession,
AtributoAdapter.NAME,
                        atributoAdapterVO);
        }

        // Tiene errores no recuperables
        if (atributoVO.hasErrorNonRecoverable()) {
                log.error("error en: "  + funcName + ": " + atributoAdapterVO.errorString());
                return forwardErrorNonRecoverable(mapping, request, funcName,
AtributoAdapter.NAME,

atributoAdapterVO);
        }
```

Redirección a Página de Confirmación de la acción ejecutada.

```
        // Fue Exitoso
        return forwardConfirmarOk(mapping, request, funcName, AtributoAdapter.NAME);
```

En caso de producirse una excepción, ejecutamos el baseException para visualizar el stacktrace de la excepción en una ventana de confirmación.

```
        } catch (Exception exception) {
                return baseException(mapping, request, funcName, exception,
AtributoAdapter.NAME);
        }
    }
```

---

borrar()

Borra el atributo seleccionado.

```
        public ActionForward borrar(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response) throws Exception
{
```

Obtención del nombre del método actual. Logeo.
Determinación del permiso de acceso.

```
        String funcName = DemodaUtil.currentMethodName();
        if (log.isDebugEnabled()) log.debug("entrando en " + funcName);
        UserSession userSession = canAccess(request, mapping, funcName);
        if (userSession == null) return forwardErrorSession(request);
```

Obtención del atributoAdapterVO desde el userSession, usando como clave el AtributoAdapter.NAME.
Si no se puede obtener (atributoAdapterVO == null): Logea el error y redirige a una Ventana que muestra el Error.

```
        try {
                // Bajo el adapter del userSession
                AtributoAdapter atributoAdapterVO = (AtributoAdapter)

        userSession.get(AtributoAdapter.NAME);

                // Si es nulo no se puede continuar
                if (atributoAdapterVO == null) {
                        log.error("error en: "  + funcName + ": " + AtributoAdapter.NAME +
                                                        " IS NULL. No se pudo obtener de
la sesion");
                        return forwardErrorSessionNullObject(mapping, request, funcName,

        AtributoAdapter.NAME);
                }
```

Llamada al Servicio.

Manejo de errores recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.
Manejo de errores No recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.

```
		// llamada al servicio
		AtributoVO atributoVO =
DefServiceLocator.getAtributoService().deleteAtributo(userSession,
                                                        atributoAdapterVO.getAtributo());

		// Tiene errores recuperables
		if (atributoVO.hasErrorRecoverable()) {
			log.error("recoverable error en: "  + funcName + ": " +
atributoAdapterVO.infoString());
			saveDemodaErrors(request, atributoVO);
			request.setAttribute(AtributoAdapter.NAME, atributoAdapterVO);
			return mapping.findForward(DefConstants.FWD_ATRIBUTO_VE_ADAPTER);
		}

		// Tiene errores no recuperables
		if (atributoVO.hasErrorNonRecoverable()) {
			log.error("error en: "  + funcName + ": " + atributoAdapterVO.errorString());
			return forwardErrorNonRecoverable(mapping, request, funcName,
AtributoAdapter.NAME,

		atributoAdapterVO);
		}
```

Redirección a la página de confirmación de la acción ejecutada.

```
		// Fue Exitoso
		return forwardConfirmarOk(mapping, request, funcName, AtributoAdapter.NAME);
```

En caso de producirse una excepción, ejecutamos el baseException para visualizar el stacktrace de la excepción en una ventana de confirmación.

```
		} catch (Exception exception) {
			return baseException(mapping, request, funcName, exception,
AtributoAdapter.NAME);
		}
	}
```

Invocación al baseVolver pasando el nombre del atributoAdapter
Vuelve a donde indique el atributoAdapter y remueve el model del userSession.

---

volver()

Regresa a la ejecución del action anterior, para lo cual invoca al método baseVolver.

```
		public ActionForward volver(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{

			return baseVolver(mapping, form, request, response, AtributoAdapter.NAME);
		}
```

---

paramTipoAtributo()

Realiza la recarga de datos disparada por el cambio en la selección del combo Tipo de Atributo, llamando al método getAtributoAdapterParamTipoAtributo del servicio AtributoService.

```
		public ActionForward paramTipoAtributo (ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response) throws Exception
{
```

Obtención del nombre del método actual. Logeo.
Determinación del permiso de acceso.
Si no tiene permiso, redirige a la pantalla de error.

```
		String funcName = DemodaUtil.currentMethodName();
		if (log.isDebugEnabled()) log.debug("entrando en " + funcName);
		UserSession userSession = canAccess(request, mapping, funcName);
		if (userSession==null) return forwardErrorSession(request);
```

Obtención del atributoAdapterVO desde el userSession, usando como clave el AtributoAdapter.NAME.
Si no se puede obtener (atributoAdapterVO == null): Logea el error y redirige a una Ventana que muestra el Error.

```
		try {
			// Bajo el adapter del userSession
			AtributoAdapter atributoAdapterVO = (AtributoAdapter)

		userSession.get(AtributoAdapter.NAME);		// Si es nulo no se puede continuar
			if (atributoAdapterVO == null) {
				log.error("error en: "  + funcName + ": " + AtributoAdapter.NAME +
									" IS NULL. No se pudo obtener de
la sesion");
				return forwardErrorSessionNullObject(mapping, request, funcName,

		AtributoAdapter.NAME);
			}
```

Carga de datos submitidos en el request al atributoAdapterVO y validaciones de tipos, en la ejecución del método populateVO.
Manejo de errores recuperables, que el populateVO pudo haber cargado en el atributoAdapterVO.

```
		// Recuperamos datos del form en el vo
		DemodaUtil.populateVO(atributoAdapterVO, request);

		// Tiene errores recuperables
		if (atributoAdapterVO.hasErrorRecoverable()) {
			log.error("recoverable error en: "  + funcName + ": " +
atributoAdapterVO.infoString());
			saveDemodaErrors(request, atributoAdapterVO);
			return forwardErrorRecoverable(mapping, request, userSession,
AtributoAdapter.NAME,
			    atributoAdapterVO);
		}
```

Llamada al servicio.
Manejo de errores recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.
Manejo de errores No recuperables, que el servicio pudo haber cargado en el atributoAdapterVO.

```
		// llamada al servicio
		atributoAdapterVO =
DefServiceLocator.getAtributoService().getAtributoAdapterParamTipoAtributo(
								userSession, atributoAdapterVO);

		// Tiene errores recuperables
		if (atributoAdapterVO.hasErrorRecoverable()) {
			log.error("recoverable error en: "  + funcName + ": " +
atributoAdapterVO.infoString());
			saveDemodaErrors(request, atributoAdapterVO);
			return forwardErrorRecoverable(mapping, request, userSession,
								AtributoAdapter.NAME,
atributoAdapterVO);
		}
```

```
		// Tiene errores no recuperables
		if (atributoAdapterVO.hasErrorNonRecoverable()) {
			log.error("error en: "  + funcName + ": " + atributoAdapterVO.errorString());
			return forwardErrorNonRecoverable(mapping, request, funcName,
                                                                AtributoAdapter.NAME,
atributoAdapterVO);
		}
```

Carga del atributoAdapterVO en el request para que esté disponibles cuando se renderiza la página jsp.

```
		// Envio el VO al request
		request.setAttribute(AtributoAdapter.NAME, atributoAdapterVO);
```

Colocamos el atributoAdapterVO en el userMap del userSession, con clave definida por AtributoAdapter.NAME.

```
		// Subo el apdater al userSession
		userSession.put(AtributoAdapter.NAME, atributoAdapterVO);
```

Redirigir al forward declarado en el struts-config-def.xml, usando la variable estática declarada en DefConstants.FWD_ATRIBUTO_ADAPTER.

```
		return mapping.findForward(DefConstants.FWD_ATRIBUTO_ADAPTER);
```

En caso de producirse una excepción, ejecutamos el método baseException para visualizar el stacktrace de la excepción en una pantalla de confirmación.

```
		} catch (Exception exception) {
			return baseException(mapping, request, funcName, exception,
AtributoAdapter.NAME);
		}
	}
  }
```

atributoAdapter.jsp

Utilizado en la creación y modificación del Atributo.

Declaración de Tags utilizado en la jsp.
Inclusión del archivo submitForm.js.
Formulario.
Inclusión del jsp que realiza el Manejo de Mensajes y Advertencias.
Inclusión del jsp que realiza el Manejo de Errores.
Título correspondiente a la Administración de Atributos.

```
		<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
		<%@ taglib uri="/tags/struts-html" prefix="html" %>
		<%@ taglib uri="/tags/struts-logic" prefix="logic" %>

		<script type="text/javascript" language="javascript">
			<%@include file="/base/submitForm.js"%>
		</script>

		<!-- Tabla que contiene todos los formularios -->
		<html:form styleId="filter" action="/def/AdministrarAtributo.do">

			<!-- Mensajes y/o Advertencias -->
			<%@ include file="/base/warning.jsp" %>
			<!-- Errors  -->
			<html:errors bundle="base"/>

			<h1><bean:message bundle="def" key="def.atributoAdapter.title"/></h1>
```

Datos del Atributo:
Contiene:

propiedades no editables.
propiedades editables.
combos con las listas de valores seleccionables.

Utiliza el tag de Struts <logic:equal> para determinar la edición o solo visualización de propiedades.

Si la propiedad se puede editar: utilizar el tag <htm:text>.
Si la propiedad no se puede editar utilizar <bean:write>.

```
<!-- Atributo -->
<fieldset>
        <legend><bean:message bundle="def" key="def.atributo.title"/></legend>

        <table class="tabladatos">
          <tr>
            <td>
                <label>
                  <bean:message bundle="def"
key="def.atributo.codAtributo.label"/> :
                </label>
            </td>
            <td class="normal">
                <logic:equal name="atributoAdapterVO"
                             property="atributo.modifCodAtributoEnabled"
                             value="disabled">
                  <bean:write name="atributoAdapterVO"
property="atributo.codAtributo"/>
                </logic:equal>
                <logic:notEqual name="atributoAdapterVO"
                                property="atributo.modifCodAtributoEnabled"
                                value="disabled">
                  <html:text name="atributoAdapterVO"
property="atributo.codAtributo" size="15"                                maxlength="20"
/>
                </logic:notEqual>
            </td>
            <td><label>
                  <bean:message bundle="def"
key="def.atributo.desAtributo.label"/>:                                </label>
            </td>
            <td class="normal">
                <html:text name="atributoAdapterVO"
property="atributo.desAtributo" size="20"
                 maxlength="100"/>
          </td>
          </tr>
          <tr>
            <td><label><bean:message bundle="def"
                key="def.tipoAtributo.label"/> : </label>
            </td>
            <td class="normal">
                <logic:equal name="atributoAdapterVO"
                             property="atributo.modifTipoAtributoEnabled"
                             value="disabled">
                  <bean:write name="atributoAdapterVO"
                              property="atributo.tipoAtributo.desTipoAtributo"/>
                </logic:equal>
                <logic:notEqual name="atributoAdapterVO"
                          property="atributo.modifTipoAtributoEnabled"
value="disabled">
                          <html:select name="atributoAdapterVO"
                                       property="atributo.tipoAtributo.id"
styleClass="select"
onchange="submitForm('paramTipoAtributo', '');">
                                <html:optionsCollection name="atributoAdapterVO"
                                        property="listTipoAtributo"
label="desTipoAtributo"
                                     value="id" />
                                </html:select>
                </logic:notEqual>
```

```
                    </td>
                    <td><label><bean:message bundle="def" key="def.domAtr.label"/>:
</label></td>
                    <td class="normal">
                        <logic:equal name="atributoAdapterVO"
property="atributo.modifDomAtrEnabled"
                                value="disabled">
                            <bean:write name="atributoAdapterVO"
                                property="atributo.domAtr.desDomAtr"/>
                        </logic:equal>
                        <logic:notEqual name="atributoAdapterVO"
                                property="atributo.modifDomAtrEnabled"
value="disabled">
                            <html:select name="atributoAdapterVO"
property="atributo.domAtr.id"
                                styleClass="select">
                              <html:optionsCollection name="atributoAdapterVO"
property="listDomAtr"
                                label="desDomAtr" value="id" />
                            </html:select>
                        </logic:notEqual>
                    </td>
                  </tr>
                  <tr>
                        <td>
                          <label>
                            <bean:message bundle="def"
key="def.atributo.valorDefecto.label"/>:
                          </label>
                        </td>
                        <td class="normal" colspan="3">
                          <html:text name="atributoAdapterVO"
property="atributo.valorDefecto"
                                size="20" maxlength="100"/>
                        </td>
                  </tr>
                  <tr>
                    <td>
                      <label>
                        <bean:message bundle="def"
key="def.atributo.esAtributoBusqueda.label"/>:                    </label>
                  </td>
                    <td class="normal" colspan="3">
                        <html:select name="atributoAdapterVO"
                            property="atributo.esAtributoBusqueda.id"
styleClass="select"
                            onchange="submitForm('paramEsAtributoBusqueda', '');">
                          <html:optionsCollection name="atributoAdapterVO"
property="listSiNo"
                                label="value" value="id" />
                          </html:select>
                    </td>
                  </tr>
                  <tr>
                    <td>
                        <label>
                          <bean:message bundle="def"
key="def.atributo.admBusPorRan.label"/>:
                        </label>
                    </td>
                    <td class="normal">
                        <logic:equal name="atributoAdapterVO"

property="atributo.modifAdmBusPorRanEnabled" value="disabled">
                          <bean:write name="atributoAdapterVO"


          property="atributo.admBusPorRan.value"/>
                        </logic:equal>
                        <logic:notEqual name="atributoAdapterVO"
                            property="atributo.modifAdmBusPorRanEnabled"
value="disabled">
                          <html:select name="atributoAdapterVO"
property="atributo.admBusPorRan.id"
                                styleClass="select">
```

```
                                <html:optionsCollection name="atributoAdapterVO"
property="listSiNo"
                                                                label="value" value="id" />
                                </html:select>
                                </logic:notEqual>
                        </td>
                        <td>
                                <label>
                                <bean:message bundle="def"
key="def.atributo.admBusMulVal.label"/>:
                                </label>
                        </td>
                        <td class="normal">
                                <logic:equal name="atributoAdapterVO"
                                                property="atributo.modifAdmBusMulValEnabled"
value="disabled">
                                <bean:write name="atributoAdapterVO"
property="atributo.admBusMulVal.value"/>
                                </logic:equal>
                                <logic:notEqual name="atributoAdapterVO"
                                        property="atributo.modifAdmBusMulValEnabled"
value="disabled">
                                        <html:select name="atributoAdapterVO"
                                                property="atributo.admBusMulVal.id"
styleClass="select">
                                        <html:optionsCollection name="atributoAdapterVO"
property="listSiNo"
                                                                label="value" value="id" />
                                        </html:select>
                                </logic:notEqual>
                        </td>
                        </tr>
                        <tr>
                        <td><label><bean:message bundle="base" key="base.estado.label"/>:
</label></td>
                        <td class="normal" colspan="3">
                                <html:select name="atributoAdapterVO"
property="atributo.estado.id"
                                        styleClass="select">
                                <html:optionsCollection name="atributoAdapterVO"
property="listEstado"
                                                                label="value" value="id" />
                                </html:select>
                        </td>
                        </tr>
                        </table>
```

Botones: Modificar, Agregar y Duplicar.
Son visualizados en la página, de acuerdo al valor de la propiedad “act” del atributoAdapterVO.

```
        <p>
                <logic:equal name="atributoAdapterVO" property="act" value="modificar">
                        <html:button property="btnAceptar"  styleClass="boton"

        onclick="submitForm('actualizar', '');">
                                <bean:message bundle="base"
key="abm.button.modificar"/>
                        </html:button>
                </logic:equal>
                 
                <logic:equal name="atributoAdapterVO" property="act" value="agregar">
                        <html:button property="btnAceptar"  styleClass="boton"

        onclick="submitForm('insertar', '');">
                                <bean:message bundle="base"
key="abm.button.agregar"/>
                        </html:button>
                </logic:equal>
                 
                <logic:equal name="atributoAdapterVO" property="act" value="duplicar">
                        <html:button property="btnAceptar"  styleClass="boton"
```

```
        onclick="submitForm('insertar', '');">
                                <bean:message bundle="base"
key="abm.button.agregar"/>
                            </html:button>
                    </logic:equal>
            </p>
    </fieldset>
    <!-- Atributo -->
```

Tabla de botones: contiene al botón Volver.

```
            <table class="tablabotones">
                    <tr>
                            <td align="left">
                              <html:button property="btnVolver" styleClass="boton"
                                                    onclick="submitForm('volver', '');">
                                        <bean:message bundle="base" key="abm.button.volver"/>
                                 </html:button>
                            </td>
                    </tr>
            </table>
```

Propiedades hidden utilizadas por el action para el control de navegación:
method: nombre del método a ejecutar.
selectedId: Id del elemento seleccionado.
isSubmittedForm: Indica si es submitido por una acción desde browser y no
desde otro action.

```
                    <input type="hidden" name="method" value=""/>
                    <input type="hidden" name="selectedId" value=""/>
                    <input type="hidden" name="isSubmittedForm" value="true"/>

            </html:form>
            <!-- Fin Tabla que contiene todos los formularios -->
```

atributoVEAdapter.jsp

Archivo JSP utilizado para Ver, Eliminar (Activar y Desactivar) un Atributo.
Muestra datos del Atributo y la acción previamente determinada.
La construcción es similar al atributoAdapter.jsp, pero los datos que contiene son de solo lectura.

Properties:

Son los archivos de recursos de struts, donde se ingresan todos los mensajes que serán mostrados en las diferentes pantallas, tales como títulos, leyendas, etiquetas, mensajes de advertencia o error, etc.

Se encuentran todos en la misma carpeta y existe una por cada módulo, quedando de la siguiente manera:

```
siat/view/src/WEB-INF/src/resources/<modulo>.properties
```

Cada archivo está divido en dos grandes secciones “Entidad” y “GUI”.
En la primera están las entradas correspondientes a cada entidad definida en el iface para el módulo en cuestión, osea que habrá un grupo de etiquetas para cada VO, y solo para estos y no para los Adapter ni SearchPage.

En las segunda sección “GUI”, estarán las etiquetas correspondientes a cada jsp y que no se pueda obtener de la primer sección.

Sección Entidad

Después del comentario que indica el comienzo de la entidad, y siguiendo la siguiente regla para la escritura de las claves:

```
[modulo].[bean].[propiedad].[modificador]
```

Tenemos:

```
# Atributo ----------------------------------------
def.atributo.label=Atributo
def.atributo.title=Datos del Atributo
def.atributo.unique=El Atributo ya Existe
def.atributo.codAtributo.label=C\u00F3digo
def.atributo.codAtributo.ref=C\u00F3digo Atributo
def.atributo.codAtributo.required=Debe ingresar un C\u00F3digo de Atributo
def.atributo.codAtributo.unique=El C\u00F3digo ya Existe
def.atributo.desAtributo.label=Descripci\u00F3n
def.atributo.desAtributo.ref=Desc. Atributo
def.atributo.valorDefecto.label=Valor por Defecto
def.atributo.esAtributoBusqueda.label=Es Atributo B\u00FAsqueda
def.atributo.admBusPorRan.label=Admite B\u00FAsqueda por Rango
def.atributo.admBusMulVal.label=Admite B\u00FAsqueda Multivalor
def.atributo.fechaDesde.label=Fecha Desde
def.atributo.fechaDesde.formatError=Formato invalido en Fecha Desde
def.atributo.fechaHasta.label=Fecha Hasta
def.atributo.fechaHasta.formatError=Formato invalido en Fecha Hasta
```

Donde:

Para representar el nombre descriptivo de la entidad usamos “label”.

```
def.atributo.label=Atributo
```

Para reutilizar cada vez que se necesite mostrar un cuadro con los datos se usa “title”.

```
def.atributo.title=Datos del Atributo
```

Para representar las propiedades del bean, en su propio mantenedor o en casos de uso que lo afectan directamente, usamos “label”.

```
def.atributo.codAtributo.label=C\u00F3digo
def.atributo.desAtributo.label=Descripci\u00F3n
def.atributo.valorDefecto.label=Valor por Defecto
```

Cuando necesitamos hacer referencia a una propiedad del bean desde otro mantenedor, caso de uso, desde columnas en el resultado de una búsqueda, etc., se utiliza “ref”.

```
def.atributo.codAtributo.ref=C\u00F3digo Atributo
def.atributo.desAtributo.ref=Desc. Atributo
```

Para cargar los mensajes relacionados a propiedades requeridas, utilizamos “required”.

```
def.atributo.codAtributo.required=Debe ingresar un C\u00F3digo de Atributo
```

Para la validación de formatos de fecha, horas y tipos numéricos, se utiliza “formatError”, esta regla no se puede variar, ya que la función populateVO() utilizada en el controlador para cargar los datos submitidos por una página al correspondiente model, generará clave de esta forma y las cargará a las lista de errores para ser mostradas.

```
def.atributo.fechaDesde.formatError=Formato invalido en Fecha Desde
def.atributo.fechaHasta.formatError=Formato invalido en Fecha Hasta
```

Cuando tengamos que realizar validaciones de unicidad, usaremos “unique”.

```
def.atributo.unique=El Atributo ya Existe
def.atributo.codAtributo.unique=El C\u00F3digo ya Existe
```

Sección GUI

Para esta sección la regla es:

```
[modulo].[jsp].[modificador]
```

Casos Search Page:

```
def.atributoSearchPage.title=Administraci\u00F3n de Atributos
def.atributoSearchPage.legend=Permite buscar, ver, modificar y agregar Atributos y sus Dominios
```

Caso Formularios de abms(Adapter):

```
def.atributoAdapter.title=Administraci\u00F3n de Atributos
```

Casos no contemplados

Las reglas de arriba representan la mayoría de los casos, así y quedan casos que no contemplan:

Para tales casos siempre se crean labels que quedan referidos a las páginas JSP en las cuales se da el caso extraordinario.

Las reglas dadas representan la mayoría de los casos. Para los casos no cubierto por las mismas, se crean labels referidas a las páginas JSP.