SISTEMAS DISTRIBUIDOS

PROGRAMA
DE INGENIERIA DE SISTEMAS

ING. DANIEL EDUARDO PAZ PERAFÁN

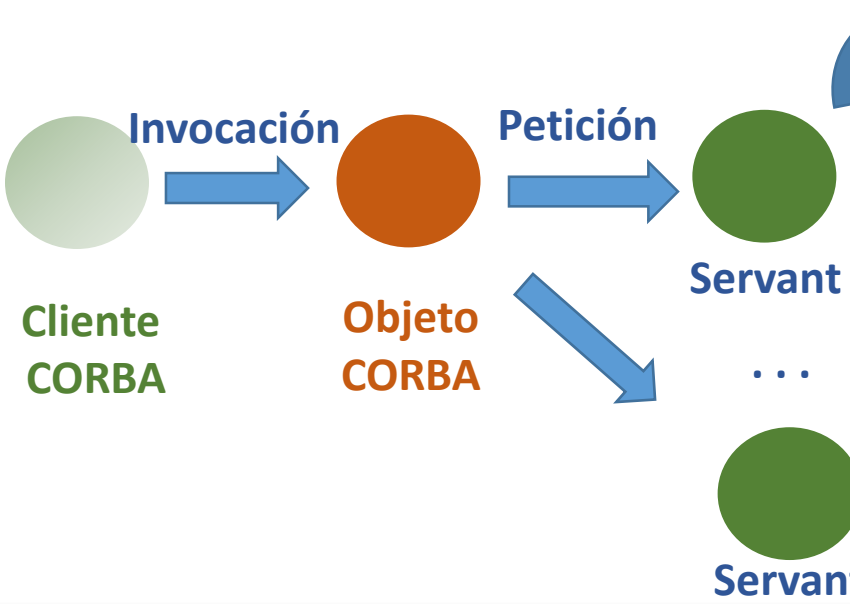
Portable object adapter

Que es un objeto CORBA

Un objeto CORBA es un objeto virtual es decir no tiene existencia por sí mismo.

Representa un servant

Programa que implementa un conjunto de aplicaciones que se pueden invocar

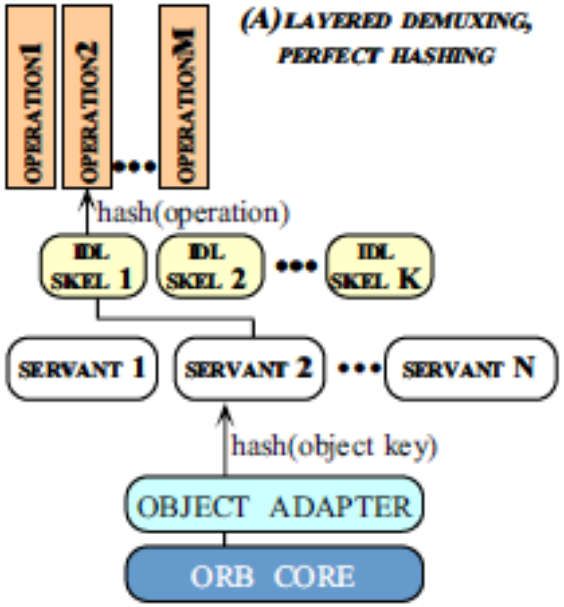


Un objeto corba puede vivir independientemente si un proceso servidor es restaurado

Portable object adapter

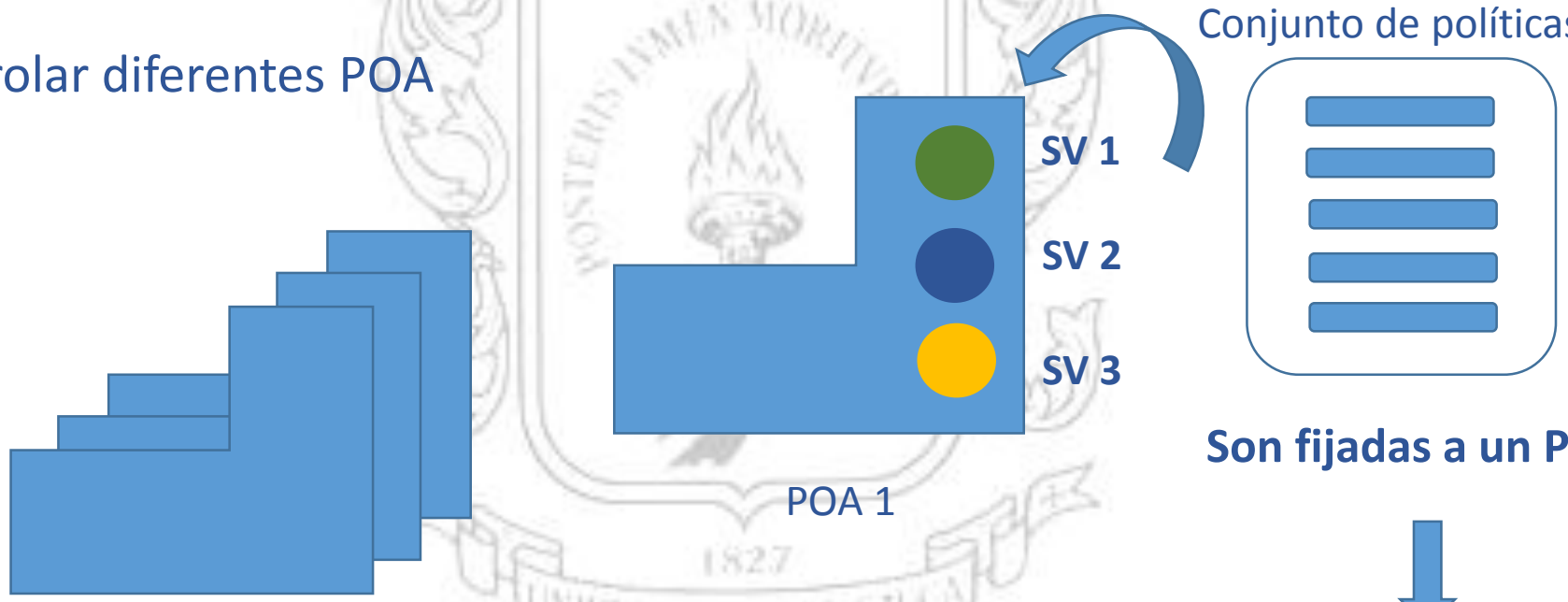
Funciones

- Creación de objetos CORBA y la generación de sus referencias.
- Activación y desactivación de servants.
- Demultiplexación de peticiones a servants
- Colaborar con los skeleton para invocar peticiones al servant apropiado.



Portable object adapter

Permite controlar diferentes POA



Son fijadas a un POA

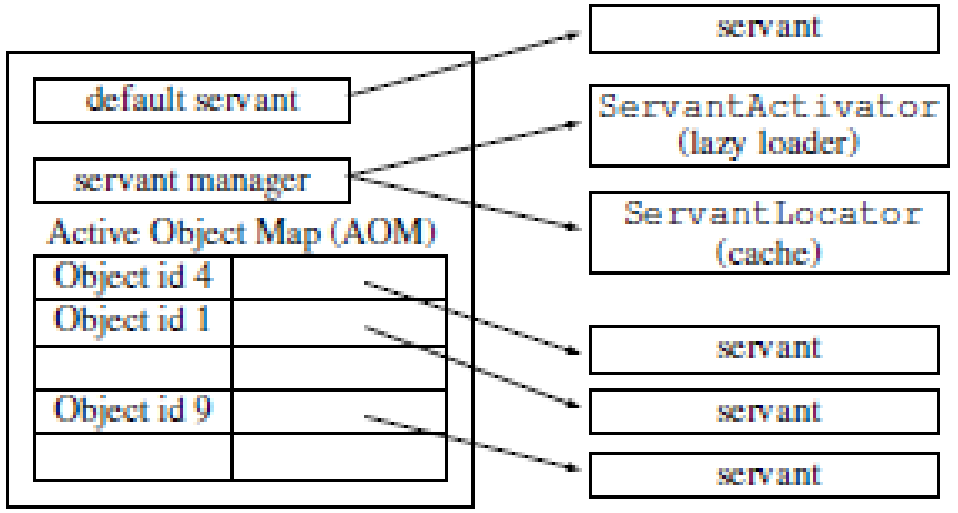
Ofrecen una calidad en el servicio

Aplicar políticas a diferentes POA, permite controlar la calidad de acceso a los servant

Portable object adapter

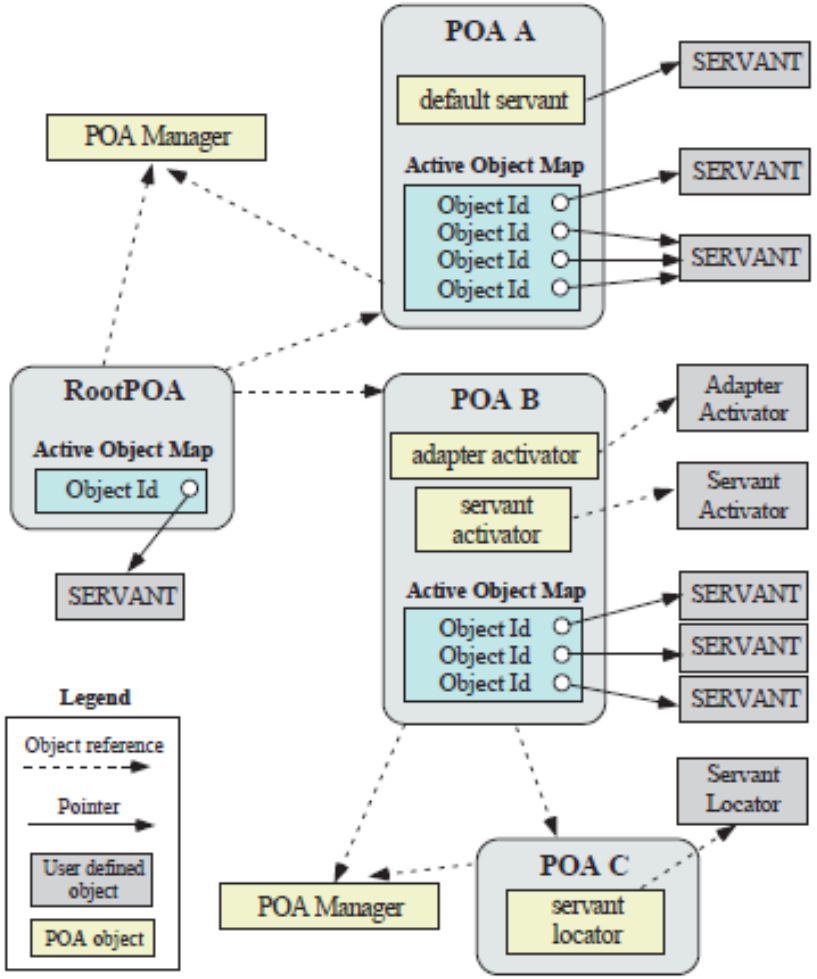
Un poa puede ser de varios tipos. Esto permite a los programadores hacer varias configuraciones respecto al ciclo de vida de los servidores y la relación entre los servidores y los objetos CORBA que representan.

- **POA "Simple"**
- **POA "Lazy Loader"**
- **POA "Cache"**
- **POA "servant por defecto"**



Componentes de un POA

Arquitectura Portable object adapter



rootPOA: POA raíz que permite crear POAs.

POA Manager: Encapsula el estado de procesamiento de uno o más POAs. Permite activar y desactivar POAs.

Servant Manager: Hay dos tipos, ServantActivator and ServantLocator. Permiten activar servants por demanda, gestionar la relación de un objeto con un servant y determinar si un object id existe.

Adapter Activator: Permite crear POAs.

POLÍTICAS PARA FIJAR EL TIPO DE POA

- Un POA se crea llamando a la operación create_POA (). Uno de los parámetros de esta operación es una secuencia de objetos de política.
- Permiten estableces un control sobre la identidad, estado, almacenamiento y ciclo de vida de un servant
- Las políticas fijadas a un POA determinan qué tipo de POA se crea.

```
enum ServantRetentionPolicyValue {RETAIN, NON_RETAIN}
enum IdUniquenessPolicyValue {UNIQUE_ID, MULTIPLE_ID}
enum RequestProcessingPolicyValue{USE_ACTIVE_OBJECT_MAP_ONLY,
USE_DEFAULT_SERVANT,
USE_ SERVANT_MANAGER};
```

Políticas se crean utilizando enumeraciones

Referencia de : http://docs.oracle.com/javase/1.5.0/docs/guide/idl/POA.html

POLÍTICAS PARA FIJAR EL TIPO DE POA

Ejemplo para fijación de políticas

```
System.out.println("1. Crea e inicia el orb");
ORB orb = ORB.init(args, null);
System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");
org.omg.CORBA.Object objPOA = null;
objPOA=orb.resolve_initial_references("RootPOA");
POA rootPOA = POAHelper.narrow (objPOA);
```

```
Policy[] tpolicy = new Policy[3];

tpolicy[0] =rootPOA.create_lifespan_policy (LifespanPolicyValue.TRANSIENT );
tpolicy[1] = rootPOA.create_request_processing_policy ( RequestProcessingPolicyValue.USE_ACTIVE_OBJECT_MAP_ONLY );
tpolicy[2] = rootPOA.create_servant_retention_policy (ServantRetentionPolicyValue.RETAIN);
// Crea un POA pasando una política POA
transientPOA = rootPOA.create_POA("childPOA",null, tpolicy );
```

```
create_POA(String adapter_name, POAManager a_POAManager, Policy[] policies)                POA
create_id_assignment_policy(IdAssignmentPolicyValue value)                  IdAssignmentPolicy
create_id_uniqueness_policy(IdUniquenessPolicyValue value)                  IdUniquenessPolicy
create_implicit_activation_policy(ImplicitActivationPolicyValue value) ImplicitActivationPolicy
create_lifespan_policy(LifespanPolicyValue value)                              LifespanPolicy
create_reference(String intf)                                                          Object
create_reference_with_id(byte[] oid, String intf)                                      Object
create_request_processing_policy(RequestProcessingPolicyValue value)    RequestProcessingPolicy
create_servant_retention_policy(ServantRetentionPolicyValue value)      ServantRetentionPolicy
create_thread_policy(ThreadPolicyValue value)                                  ThreadPolicy
```

POLÍTICAS PARA FIJAR EL TIPO DE POA

Servant Retention Policy

Determina si los identificadores de objeto se conservan o no en el POA por un mapa de objetos activo (AOM).

enum ServantRetentionPolicyValue { **RETAIN,** **NON_RETAIN** }

Indica que el POA tiene un AOM

Indica que el POA NO tiene un AOM

POLÍTICAS PARA FIJAR EL TIPO DE POA

Id Uniqueness Policy

Determina si hay una asignación uno a uno o una asignación de varios a uno entre identificadores de objetos y servants.

enum IdUniquenessPolicyValue { **UNIQUE_ID**, **MULTIPLE_ID** }

Permite a un servant representar muchos objetos corba

Permite una asignación uno a uno entre servants y un objetos corba

Debes utilizar la política MULTIPLE_ID para construir el POA “por defecto”

POLÍTICAS PARA FIJAR EL TIPO DE POA

Request Processing Policy

Determina si el POA tiene una AOM, un servant por defecto y / o un servant manager:

enum RequestProcessingPolicyValue{

USE_ACTIVE_OBJECT_MAP_ONLY, → Indica que el POA tiene una AOM

USE_DEFAULT_SERVANT, → Indica que el POA tiene un servant por defecto

USE_ SERVANT_MANAGER → Indica que el POA tiene un servant manager

};

POLÍTICAS UTILIZADAS PARA CONSTRUIR POA

POLÍTICAS UTILIZADAS		POA OBTENIDO
USE_ACTIVE_OBJECT_MAP_ONLY RETAIN. UNIQUE_ID	No es necesario	POA "simple"
USE_DEFAULT_SERVANT MULTIPLE_ID NON_RETAIN	No es necesario	POA "servant por defecto"

Un servant por defecto debe ser fijado en el POA, usando el método set_servant(). La petición es enviada al servant por defecto.

```
enum ServantRetentionPolicyValue {RETAIN, NON_RETAIN}
enum IdUniquenessPolicyValue {UNIQUE_ID, MULTIPLE_ID}
enum RequestProcessingPolicyValue{USE_ACTIVE_OBJECT_MAP_ONLY,
USE_DEFAULT_SERVANT,
USE_ SERVANT_MANAGER};
```

POLÍTICAS UTILIZADAS PARA CONSTRUIR POA

POLÍTICAS UTILIZADAS		POA OBTENIDO
USE_SERVANT_MANAGER RETAIN. UNIQUE_ID	→ No es necesario	POA "Lazy loader"
USE_SERVANT_MANAGER NON_RETAIN.		POA "cache"

Un servant manager debe ser registrado con el POA usando método set_servant_manager(). El servant manager permite localizar o activar un servant o lanzar una excepción.

```
enum ServantRetentionPolicyValue {RETAIN, NON_RETAIN}
enum IdUniquenessPolicyValue {UNIQUE_ID, MULTIPLE_ID}
enum RequestProcessingPolicyValue{USE_ACTIVE_OBJECT_MAP_ONLY,
USE_DEFAULT_SERVANT,
USE_ SERVANT_MANAGER};
```

POLÍTICAS PARA FIJAR EL TIPO DE POA

Políticas para un POA "simple"

```
System.out.println("1. Crea e inicia el orb");
ORB orb = ORB.init(args, null);
System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");
org.omg.CORBA.Object objPOA = null;
objPOA=orb.resolve_initial_references("RootPOA");
POA rootPOA = POAHelper.narrow(objPOA);
```

```
Policy[] tpolicy = new Policy[3];

tpolicy[0] =rootPOA.create_lifespan_policy (LifespanPolicyValue.TRANSIENT );
tpolicy[1] = rootPOA.create_request_processing_policy ( RequestProcessingPolicyValue.USE_ACTIVE_OBJECT_MAP_ONLY );
tpolicy[2] = rootPOA.create_servant_retention_policy (ServantRetentionPolicyValue.RETAIN);
// Crea un POA pasando una política POA
transientPOA = rootPOA.create_POA("POASimple",null, tpolicy );
```

```
create_POA(String adapter_name, POAManager a_POAManager, Policy[] policies)                POA
create_id_assignment_policy(IdAssignmentPolicyValue value)                  IdAssignmentPolicy
create_id_uniqueness_policy(IdUniquenessPolicyValue value)                  IdUniquenessPolicy
create_implicit_activation_policy(ImplicitActivationPolicyValue value) ImplicitActivationPolicy
create_lifespan_policy(LifespanPolicyValue value)                              LifespanPolicy
create_reference(String intf)                                                          Object
create_reference_with_id(byte[] oid, String intf)                                      Object
create_request_processing_policy(RequestProcessingPolicyValue value)    RequestProcessingPolicy
create_servant_retention_policy(ServantRetentionPolicyValue value)      ServantRetentionPolicy
create_thread_policy(ThreadPolicyValue value)                                  ThreadPolicy
```

Políticas establecer el acceso, ciclo de vida y estado de los servant en un POA

Thread Policy

Se utiliza para especificar cómo una POA utiliza subprocesos para enviar solicitudes entrantes:

Política	Objetivo
enum ThreadPolicyValue{	
ORB_CTRL_MODEL,	Especifica que el ORB tiene control sobre cómo se envían las solicitudes entrantes.
SINGLE_THREAD_MODEL,	Especifica la existencia de concurrencia entre múltiples POA
MAIN_THREAD_MODEL	Significa que todas las solicitudes entrantes se envían a través del hilo principal de la aplicación
}	

Políticas establecer el acceso, ciclo de vida y estado de los servant en un POA

Lifespan Policy

Determina el tiempo de vida de las referencias a los objetos.

```
enum LifespanPolicyValue
{
TRANSIENT,
PERSISTENT
};
```

TRANSIENT → Especifica que las referencias a OC son válidas sólo durante la duración del proceso del servidor.

PERSISTENT → Especifica que las referencias a OC son válidas incluso si el proceso servidor se destruye y reinicia.

Políticas establecer el acceso, ciclo de vida y estado de los servant en un POA

Id Assignment Policy

Especifica si los Object Ids en el POA son generados por el programador o por el ORB.

enum IdAssignmentPolicyValue {

USER_ID, ➡ Indica que el programador especifica el identificador de objeto al activar (insertar) un servant en un POA ➡ PERSISTENT

```
poa.activate_object_with_id(obj_id, sv);
```

SYSTEM_ID ➡ Indica que el sistema CORBA selecciona un identificador de objeto único cuando un servant se activa en un POA. ➡ TRANSIENT

```
poa.activate object(sv);
```

};

Políticas establecer el acceso, ciclo de vida y estado de los servant en un POA

Implicit and Explicit Activation Policy

Especifica si la activación implícita de los servants es realizada por el POA.

```
enum ImplicitActivationPolicyValue
{
```

IMPLICIT ACTIVATION, ➡ Indica una activación implícita de los servants. Esta política requiere las políticas SYSTEM_ID y RETAIN

NO IMPLICIT ACTIVATION ➡ No hay activación implícita de los servant.

```
};
```

Políticas establecer el acceso, ciclo de vida y estado de los servant en un POA

Políticas por defecto

- **USE_ACTIVE_OBJECT_MAP_ONLY.**
- **RETAIN.**
- **UNIQUE_ID.**
- **ORB_CTRL_MODEL.**
- **TRANSIENT.**
- **SYSTEM_ID.**
- **IMPLICIT_ACTIVATION.**

http://docs.oracle.com/javase/7/docs/technotes/guides/idl/POA.html#policy

Servant Managers

Son opcionales. Permite que el POA active servants cuando la petición sobre un objeto activo es recibida.

Si el servidor carga todos los objetos cuando se inicia, no necesita un Servant Manager.

- **ServantActivator**

Cuando el POA tiene la política RETAIN , utiliza Servant Managers que son ServantActivators.

- **ServantLocator**

Cuando la POA tiene la política NON_RETAIN, utiliza Servant Managers que son ServantLocator

Información para crear un POA

Se crea un POA como hijo de un POA existente mediante la operación créate_POA en el POA principal. Para crear un nuevo POA, es necesaria la siguiente información:

Nombre del POA: Nombre que debe ser único con respecto a todos los otros POA.

POA Manager: Especifica el POA Manager que se va a asociar con el nuevo POA. Si se pasa null para este parámetro, se creará un nuevo POA Manager.

Lista de políticas: Especifique la lista de políticas que se va a asociar con el POA para controlar su comportamiento.

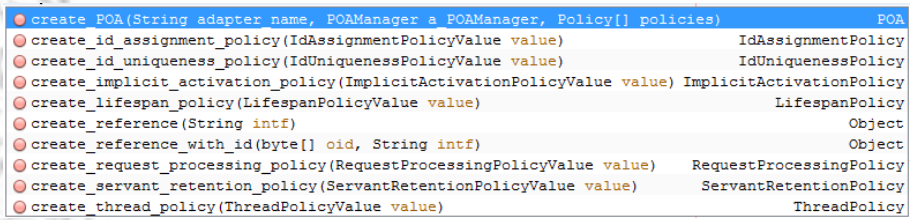
```
//Ejemplo creación de un POA
persistentPOA = rootPOA.create_POA("POASimple", rootPOA.the_POAManager(), tpolicy );
```

POA MANAGER

- Cada objeto POA tiene un objeto POA Manager asociado que controla el estado de procesamiento de los POA con los que está asociado, por ejemplo si las solicitudes a los POA están en cola o descartadas.
- Permite desactivar el POA.
- Puede estar asociado con uno o más objetos POA.
- El POAManager puede tener los siguientes estados:

Mantenimiento: En este estado, los POA asociados registraran en cola las solicitudes entrantes.
Activo: En este estado, los POA asociados comenzarán a procesar las solicitudes.
Descarte: en este estado, los POA asociados descartarán las solicitudes entrantes.
Inactivo: En este estado, los POA asociados rechazarán las solicitudes que no han comenzado a ejecutarse, así como las solicitudes nuevas.

POLÍTICAS PARA FIJAR EL TIPO DE POA



Ejemplo creación de POA

```
Policy[] tpolicy = new Policy[3];

tpolicy[0] =rootPOA.create_lifespan_policy (LifespanPolicyValue.TRANSIENT );
tpolicy[1] = rootPOA.create_request_processing_policy ( RequestProcessingPolicyValue.USE_ACTIVE_OBJECT_MAP_ONLY );
tpolicy[2] = rootPOA.create_servant_retention_policy (ServantRetentionPolicyValue.RETAIN);
// Crea un POA pasando una política POA
transientPOA = rootPOA.create_POA("POASimple", rootPOA.the_POAManager(), tpolicy );

//Crea el objeto servant
RegistroImpl ObjServant = new RegistroImpl();

byte[] Id = "idObjeto".toString().getBytes();

transientPOA.activate_object_with_id(Id, ObjServant );

CORBA.Object referencia = myPOA.servant_to_reference(ObjServant );
```

Ver ejemplo de la plataforma

Para crear un POA correspondiente a “lazy loader” y “cache” debe fijar un servant_manager
set_servant_manager(obj)

Para crear un POA por defecto debe fijar un servant por defecto.
set_servant(obj)