Pasos para crear ASO

[Crear paquetes 1](#_Toc154421313)

[Configuración ASO 1](#_Toc154421314)

[Crear y desarrollar dto 1](#_Toc154421315)

[Crear y desarrollar interfaz en facade 2](#_Toc154421316)

[Crear y desarrollar implementación 2](#_Toc154421317)

[Crear y desarrollar mapper 2](#_Toc154421318)

[Crear y desarrollar dto en business 2](#_Toc154421319)

[Desarrollar mapper v2 2](#_Toc154421320)

[Desarrollar implementación v2 3](#_Toc154421321)

[Desarrollar implementación business v2 3](#_Toc154421322)

[Desarrollar DAO business v2 3](#_Toc154421323)

[Desarrollar implementación business v3 4](#_Toc154421324)

[Desarrollar implementación v3 4](#_Toc154421325)

# Crear paquetes

En business:

* dao
* impl
* dto
* impl

En facade:

* dto
* impl
* mapper
* Impl

# Configuración ASO

## Crear y desarrollar dto

En facade >> dto, crear las clases necesarias. En caso de utilizar POST, crear una para enviar y otra para recibir información.

En cada una definir los atributos que tendrá, indicar sus requisitos (@NotNull por ejemplo), y poner esto antes de la clase para que se instancien automáticamente los getter, setter y el constructor:

*@Getter  
@Setter  
@NoArgsConstructor*

## Crear y desarrollar interfaz en facade

Crear métodos:

* Para los POST:

ServiceResponseCreated createData(CreateProposalsIn createProposalsIn);

## Crear y desarrollar implementación

En facade >> impl, crear clase

Copiar antes de la clase:

*@Service  
@SN*(registryID = "SN000000008", logicalID = "helloWorld")  
*@VN*(vnn = "v0")  
*@Path*("/v0")

Implementar la interfaz a la clase e implementar los métodos.

Copiar antes del método:

*@Override  
@POST  
@Path*("/create")  
*@Produces*({MediaType.APPLICATION\_JSON})  
*@Consumes*({MediaType.APPLICATION\_JSON})  
*@SMC*(registryID = "SMC0000000008", logicalID = "createProposals", forcedCatalog = "gabiCatalog")

IMPORTANTE: Si no se envía información, como puede ser en los GET, solo ponemos @Produces

## Crear y desarrollar mapper

En facade >> mapper crear la interfaz y mapear los atributos de la entrada al dto.

*@Mapping*(target = "id", source = "createProposalsIn.id")  
*@Mapping*(target = "participantsNumber", source = "createProposalsIn.participantsNumber")  
*@Mapping*(target = "isVisible", source = "createProposalsIn.isVisible")

Antes de la interfaz, iniciar el Mapper.

*@Mapper*(componentModel = "spring")

## Crear y desarrollar dto en business

Copiamos y pegamos las clases del dto de facade al dto de business, poniendo “B” antes del nombre de la clase o interfaz

## Desarrollar mapper v2

En la interfaz, creamos el mapIn, para poder enviar los datos que hemos recibido en facade, a business:

BCreateProposalsIn mapIn(CreateProposalsIn createProposalsIn);

Mapeamos los datos de salida.

*@Mapping*(target = "id", source = "bCreateProposalsOut.id")  
*@Mapping*(target = "name", source = "bCreateProposalsOut.name")  
*@Mapping*(target = "amount", source = "bCreateProposalsOut.amount")

Creamos el mapOut, para poder recibir los datos procesados del business en el facade:

CreateProposalsOut mapOut(BCreateProposalsOut bCreateProposalsOut);

## Desarrollar implementación v2

En el método que se ha implementado, indicamos los parámetros de entrada que se recibirán.

*public* ServiceResponseCreated createData(CreateProposalsIn createProposalsIn) {

Llamamos al mapper

ISrvCreateProposalsMapper mapper = Mappers.getMapper(ISrvCreateProposalsMapper.*class*);

Copiamos el mapIn de la interfaz del mapper y cambiamos el parámetro de entrada, llamando solo al parámetro que se inicializó al implementar el método. Inicializamos el objeto business.

BCreateProposalsIn

create = mapper.mapIn(createProposalsIn);

Para mandar el objeto business a la implementación del business, antes de llamar a la implementación (antes del @Override), ponemos el nombre de la implementación del business (se indica en el @Component de la clase del impl del facade):

*@Resource*(name = "bSrvCreateProposals")

Llamamos a la interfaz del business justo debajo

IBSrvCreateProposals ibSrvCreateProposals;

De esta forma, traemos todas las dependencias y lo que necesitemos para usar la implementación del business.

Modificamos la interfaz del business, eliminamos el método que ya viene y creamos el business:

BCreateLoanOut business(BCreateLoanIn bCreateLoanIn);

## Desarrollar implementación business v2

Modificamos la clase de la implementación del business. Encima de la clase solo invocamos @Component.

Cambiamos la interfaz que implementa la clase por la del business, e implementamos el método de la interfaz.

*@Component  
public class* BSrvCreateProposals *implements* IBSrvCreateProposals

## Desarrollar DAO business v2

Antes de configurar el método invocado, creamos en el dao del business la interfaz y el paquete impl, donde crearemos la clase del DAO.

En la interfaz del DAO, creamos el objeto de salida del DAO, al que pasaremos por parámetro la clase de entrada:

BCreateProposalsOut createProposalsDAO(BCreateProposalsIn bCreateProposalsIn);

Creamos la clase del DAO en business >> dao >> impl. En esta indicamos antes de la clase el @Component, con el nombre de la clase como valor, e implementamos la interfaz.

*@Component* (value = "srvCreateProposalsDAO")  
*public class* SrvCreateProposalsDAO *implements* ISrvCreateProposalsDAO {

Implementamos el método de la interfaz, creamos el objeto que devolverá las comprobaciones que realicemos al facade, y en este aplicamos la lógica del DAO, como verificar que el campo isVisible sea falso.

*@Override  
public* BCreateProposalsOut createProposalsDAO(BCreateProposalsIn bCreateProposalsIn) {  
 BCreateProposalsOut bCreateProposalsOut = *new* BCreateProposalsOut();  
 *if*(bCreateProposalsIn.isVisible() == *false*){

## Desarrollar implementación business v3

Antes del método que hemos implementado en la clase de business >> impl, invocamos @Resource, e indicamos el valor del @Resource de la clase de la implementación del DAO, donde hemos desarrollado la lógica.

*@Resource*(name = "srvCreateProposalsDAO")  
ISrvCreateProposalsDAO iSrvCreateProposalsDAO;

Ya dentro del método que hemos implementado, creamos el objeto que devolverá las comprobaciones que hemos realizado en el dao, al cual le asignaremos:

BCreateProposalsOut bCreateProposalsOut = *new* BCreateProposalsOut();  
bCreateProposalsOut = iSrvCreateProposalsDAO.createProposalsDAO(bCreateProposalsIn);  
*return* bCreateProposalsOut;

## Desarrollar implementación v3

Iniciamos en la impl del facade el objeto que nos devolverá el business, asignándole el nombre de la interfaz.business(el nombre del business).

BCreateProposalsOut bCreateProposalsOut = ibSrvCreateProposals.business(create);

Iniciamos el objeto de salida del facade, asignándole el mapper.mapOut(objeto que nos devuelve el business).

CreateProposalsOut createProposalsOut = mapper.mapOut(bCreateProposalsOut);

Devolvemos el resultado en un objeto data y lo construimos con .build().

*return* ServiceResponseCreated.data(createProposalsOut).build();