FAM 4.0 - Estructura del projecte i classes

Taula de continguts

1. Estructura del projecte

L'esquema general d'un projecte és:

```
fam_backoffice
      ?
      ??? src
      ?    ?
      ?    ??? cat.diba.jee.core
      ?    ?
      ?    ??? cat.diba.jee.fam
      ?    ?
      ?    ??? filters
      ?    ?     ?
      ?    ?     ??? local-filter.properties
      ?    ?     ?
      ?    ?     ??? dev-filter.properties
      ?    ?     ?
      ?    ?     ??? int-filter.properties
      ?    ?     ?
      ?    ?     ??? pre-filter.properties
      ?    ?     ?
      ?    ?     ??? pro-filter.properties
      ?    ?
      ?    ??? META-INF
      ?          ?
      ?          ??? persistence.xml
      ?
      ??? generated-sources
      ?
      ??? resources
      ?
```

```
??? resources-test
?
??? scaffolding
?
??? test
?
??? WebRoot
?
??? target
```

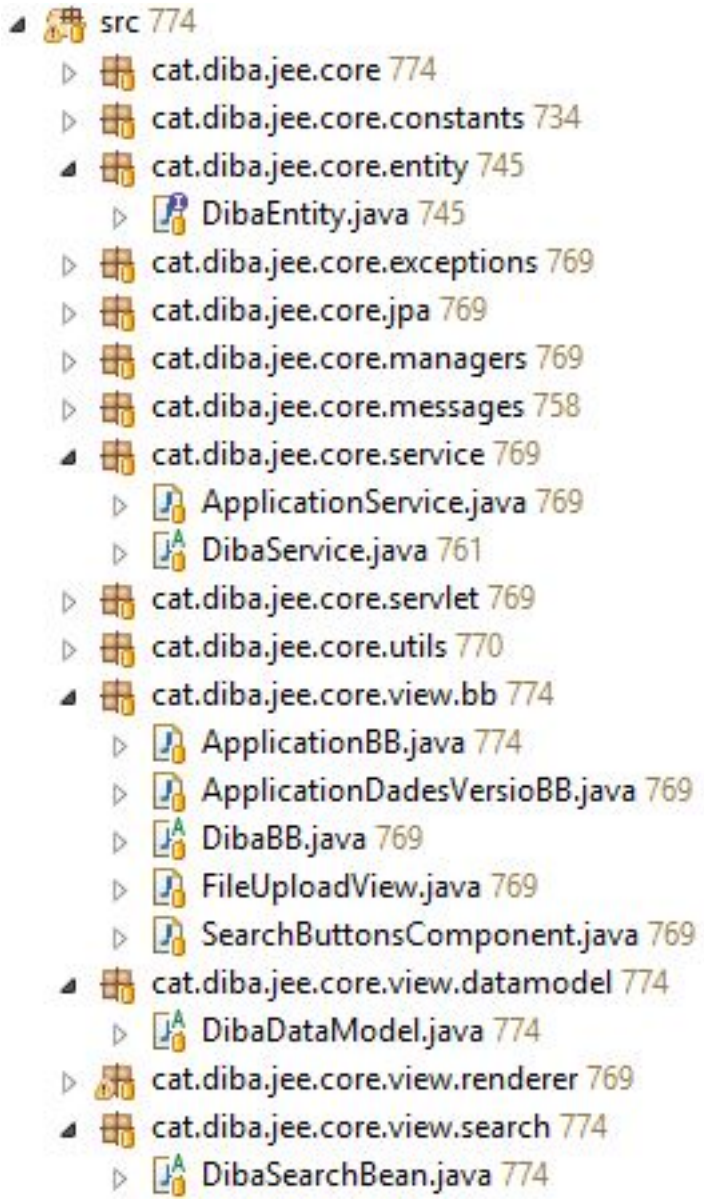
La següent taula descriu el significat dels directoris:

Directori	Descripció
src	Codi font de l'aplicació. En els següents apartats s'explica amb més detall. Directoris / paquets principals: • cat.diba.jee.core Classes base de l'aplicació, normalment no caldrà modificar-les • cat.diba.jee.<acrònim projecte> Classes concretes del projecte • filters Variables depenents de l'entorn que Maven fa servir per filtrar al generar el war • META-INF Descripció de la connexió amb el Data Source del servidor WebLogic 12c
generated-sources	Classes generades a partir de les entitats o models pel plugin de QueryDSL. En els següents apartats s'explica amb més detall
resources i resources-test	Recursos de l'aplicació pels entorns d'execució i de test. Exemple: log4j.properties
scaffolding	Plantilles per generar el codi mínim per les operacions típiques d'un CRUD (Create Read Update Delete). Veure l'article de scaffolding [1]
test	Codi font dels test
WebRoot	Contingut web del projecte: vistes, configuració, ... S'explica a l'apartat vistes
target	Directori on Maven genera el arxius temporals i el war

2. Organització de les classes

2.1. Paquets del nucli o core

La següent imatge mostra els diferents paquets del core:



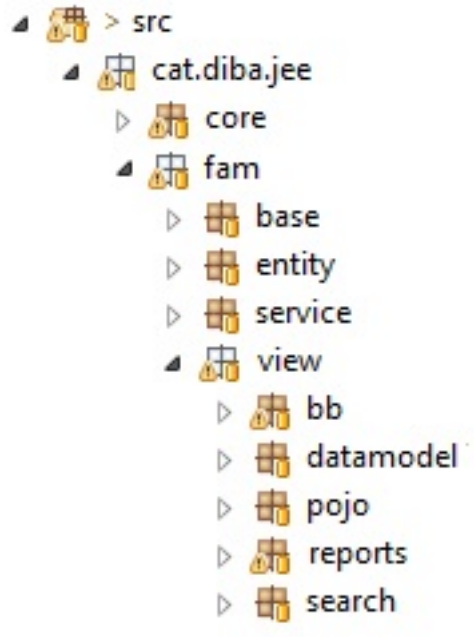
Les classes principals són:

- cat.diba.core.entity.DibaEntity: Classe arrel de totes les entitats del projecte
- cat.diba.core.service.DibaService Serveis generals implementats per gestionar un CRUD
- cat.diba.core.view.bb.DibaBB Backing Bean o Managed Bean base per gestionar les vistes JSF
- cat.diba.core.view.datamodel.DibaDataModel Model de dades per gestionar les taules amb paginació lazy
- cat.diba.core.view.search.DibaSearchBean Bean per gestionar les cerques que filtren els resultats de les taules

No existeix el paquet DAO perquè aquesta funció la realitza, en la majoria dels casos, JPA de forma que és el propi servei el que accedeix a la base de dades mitjançant l'Application Managed Persistence Context.

2.2. Paquets del projecte

La següent imatge mostra els paquets específics d'un projecte, en aquest cas del projecte FAM:



A continuació la descripció dels paquets i un exemple dels noms de classes amb l'entitat FamExpedient:

Paquet	Descripció
cat.diba.jee.fam.base	Per defecte dos classes: els missatges de l'aplicació i les constants.
cat.diba.jee.fam.entity	Entitats JPA Exemple: FamExpedient. **Al generar les entitats des de les taules (veure** Generació model de dades JPA **[2]) totes porten el prefix Fam, però alhora d'anomenar a la resta de classes s'elimina aquest prefix**
cat.diba.jee.fam.service	Serveis de les entitats. Són les classes amb el negoci de l'aplicació. Normalment trobarem un servei per entitat Exemple: ExpedientService
cat.diba.jee.fam.view.bb	Backed /Managed Bean que gestionen les pàgines JSF. Normalment trobarem un BB per entitat Exemple: ExpedientBB
cat.diba.jee.fam.view.datamodel	Models de dades de les entitats que permeten connectar-se a les taules de la vista amb paginació lazy i cerca. Normalment trobarem un DataModel per entitat Exemple: ExpedientDataModel
cat.diba.jee.fam.view.pojo	Paquet opcional. Trobarem aquelles classes que no tenen una relació directe amb la base de dades (no són entitats), però són el resultat de consultes de la base de dades o de càlculs finals
cat.diba.jee.fam.view.reports	Paquet opcional. Classes relacionades amb la generació d'informes: PDF (ireport), Word (Aspose), Excel (poi), ... Veure els exemples del FAM
cat.diba.jee.fam.view.search	Classes relacionades amb la cerca de les vistes índex. Normalment trobarem un SearchBean per entitat Exemple: ExpedientSearchBean

3. Entitats

Les entitats hauran de seguir el següent exemple (no es mostra el mapeig JPA de la taula):

```
@Entity
```

```
public class FamExpedient implements Serializable, DibaEntity {
  @Override
  public Long getId() {
    return camp amb l'identificador de la fila;
  }
}
```

3.1. Què és QueryDSL?

QueryDSL [3] é un framework DSL (domain-specific language - llenguatge específic del domini) que permet escriure consultes type-safe molt semblant a SQL i per múltiples formes d'accedir a les dades: JPA, JDO, SQL, MongoDB, ...

Per defecte, l'entorn de desenvolupament està configurat per generar una classe que embolcalla l'entitat i permet aprofitar les característiques de QueryDSL. La nova classe es crea al directori generated-sources i el mateix paquet, el nom es genera segons el patró NomEntitat -> QNomEntitat.

3.2. Per què QueryDSL?

Veiem un exemple de cerca d'expedient amb la versió del framework FAM 2.0 que es pot trobar al DAO:

```
@Override
public Query creaQuery(ExpedientSearchFilter filtre, boolean nomesCount, int initrow,
 int rows) {
  LOG.debug("creaQuery(filtre, nomesCount, initrow, rows) - Inici");
  String whereClause = " where EXP_ESTAT_ID = ESTAT_ID"
                     + " and EXP_TIPUS_ID = TIPUS_ID"
                     + " and (EXP_BAIXA_DATA > SYSDATE OR EXP_BAIXA_DATA IS NULL)";

  if (FormatUtils.checkIdValid(filtre.getFiltreIdEstat())) {
    whereClause += " and EXP_ESTAT_ID = " + filtre.getFiltreIdEstat();
  }

  if (FormatUtils.checkIdValid(filtre.getFiltreIdTipus())) {
    whereClause += " and EXP_TIPUS_ID = " + filtre.getFiltreIdTipus();
  }

  if (FormatUtils.checkStringValid(filtre.getFiltreTitol())) {
    whereClause += " and fam_translate(EXP_TITOL) like '%' || fam_translate('"
                + filtre.getFiltreTitol() + "') || '%'";
  }

  if (FormatUtils.checkStringValid(filtre.getFiltreGestorCodiSap())) {
    whereClause += " and fam_translate(EXP_GESTOR_CODI_SAP) like '%' || fam_translate
('"
                + filtre.getFiltreGestorCodiSap() + "') || '%'";
  }

  if (FormatUtils.checkStringValid(filtre.getFiltreNumExpedientSap())) {
    whereClause += " and fam_translate(EXP_NUM_EXPEDIENT_SAP) like '%' || fam_transla
te('"
                + filtre.getFiltreNumExpedientSap() + "') || '%'";
  }

  if (FormatUtils.checkStringValid(filtre.getFiltreInfoXBMQ())) {
    whereClause += " and fam_translate(EXP_INFO_XBMQ) like '%' || fam_translate('"
```

```
                + filtre.getFiltreInfoXBMQ() + "') || '%'";
  }

  QueryDTO queryDTO = new QueryDTO();
  queryDTO.setFrom(" from FAM_EXPEDIENT, FAM_EXPEDIENT_TIPUS, FAM_EXPEDIENT_ESTAT ");
  queryDTO.setWhere(whereClause);
  queryDTO.setOrder(filtre.getSqlOrdre());

  Query query = createQuery(nomesCount, queryDTO, initrow, rows);

  LOG.debug("creaQuery(filtre, nomesCount, initrow, rows) - Fi");
  return query;
}
```

I ara la mateixa cerca amb QueryDSL que es pot trobar al SearchBean:

```
public Predicate where() {
  QFamExpedient expedient = QFamExpedient.famExpedient;
  // No carregar mai els eliminat lògics
  BooleanExpression onlyNotDeleted = expedient.expBaixaData.isNull()
          .or(expedient.expBaixaData.gt(Expressions.currentDate()));
  BooleanBuilder where = new BooleanBuilder(onlyNotDeleted);

  if (null != titol) {
    where = where.and(expedient.expTitol.containsIgnoreCase(titol));
  }
  if (null != expedientSap) {
    where = where.and(expedient.expNumExpedientSap.containsIgnoreCase(expedientSap));
  }
  if (null != codiSap) {
    where = where.and(expedient.expGestorCodiSap.containsIgnoreCase(codiSap));
  }
  if (null != estatId) {
    where = where.and(expedient.famExpedientEstat.estatId.eq(estatId));
  }
  if (null != tipusId) {
    where = where.and(expedient.famExpedientTipus.tipusId.eq(tipusId));
  }
  if (null != codiXBMQ) {
    where = where.and(expedient.expInfoXbmq.like(codiXBMQ));
  }
  return where;
}
```

Principals avantatges:

- Les condicions de consulta amb QueryDSL no estàn al DAO i no estàn lligades a una implementació (SQL natiu o JPA), llavors millor independència del servei corresponent i reaprofitament dels objectes de cerca.
- Les condicions de consulta amb QueryDSL són més naturals i cercanes al llenguatge SQL

4. Servei

El servei de l'entitat estén el servei DibaService que ja implementa la major part dels mètodes d'un CRUD. Els dos punts a tenir en compte al definir el servei són:

- El tipus de transacció: @TransactionManagement(TransactionManagementType.BEAN)

- L'entitat QueryDSL: public EntityPath<DibaEntity> entityPath()

Amb la tècnica de scaffolding [1], aquests dos punts ja es generen automàticament.

```
@TransactionManagement(TransactionManagementType.BEAN)
public class ExpedientService extends DibaService {
  /**
   * Constructor per defecte
   *
   */
  public ExpedientService() {
    super(FamExpedient.class);
  }

  @Override
  public EntityPath<FamExpedient> entityPath() {
    return QFamExpedient.famExpedient;
  }
}
```

5. Cerca

La cerca de l'entitat es basa en la classe DibaSearchBean. Per simplicar el codi d'exemple, en aquesta ocació fem servir l'entitat Zona i un camp de cerca pel nom. Llavors la classe ZonaSearchBean quedaría així:

```
@ManagedBean
@SessionScoped
public class ZonaSearchBean extends DibaSearchBean {

  private static final long serialVersionUID = 1L;
  private String name;

  public String getName() {
    return name;
  }

  @Override
  public OrderSpecifier<?> orderBy() {
    return QFamZona.famZona.famZonaNom.asc();
  }

  @Override
  public OrderSpecifier<?> orderBy(String sortField, SortOrder sortOrder) {
    OrderSpecifier<?> orderSpecifier = QFamZona.famZona.famZonaNom.asc();
    return orderSpecifier;
  }

  @Override
  public void reset() {
    this.name = null;
  }

  public void setName(String name) {
    this.name = name;
  }
```

```
  public Predicate where() {
    QFamZona famZona = QFamZona.famZona;
    BooleanBuilder where = new BooleanBuilder();

    if (null != name) {
      where = where.and(famZona.famZonaNom.containsIgnoreCase(name));
    }
    return where;
  }
}
```

És important remarcar l'abast del bean: @SessionScoped. Això permet mantenir les condicions de cerca al tornar a la pàgina índex: valors dels camps, resultat, pàgina concreta de la taula

Amb la tècnica de scaffolding [1] ja es genera automàticament l'anotació.

6. Model de dades

El model de dades s'utilitza a les taules i ja incorpora la paginació lazy i la cerca.

```
public class ExpedientDataModel extends DibaDataModel<FamExpedient> {

  @Inject
  private ExpedientService expedientService;

  @Override
  public DibaService<FamExpedient> service() {
    return expedientService;
  }
}
```

7. Vista

Per defecte, el directori views tindrà una carpeta per entitat on es podrà trobar les diferents vistes. Normalment seràn dos i una opcional, segons les necessitats de l'aplicació:

- Llista: és una vista on a la part superior apareixen els diferents camps de cerca i a la part inferior una taula amb informació de les entitats
- Detall: és una vista on apareix tota l'informació de l'entitat
- Diàleg modal: és una vista per mostrar la llista dins d'una finestra modal i poder seleccionar un element

Un exemple dels noms de les vistes, continuant amb l'entitat FamExpedient, és la següent:

Sense diàleg

```
WebRoot  ?  ??? views        ?        ???
expedient              ?              ???
expedientDetall.xhtml             ?
        ??? expedientLlista.xhtml
```

Amb diàleg

```
WebRoot  ?  ??? views        ?        ???
expedient              ?              ???
expedientDetall.xhtml             ?
        ??? expedientDialog.xhtml
     ?              ??? expedientLlista.xh
tml             ?             ??? expedi
entLlistaForm.xhtml
```

Amb la tècnica de scaffolding [1] es genera automàticament l'estructura sense diàleg

URL d'origen: https://comunitatdstsc.diba.cat/wiki/fam-30-estructura-del-projecte-classes

Enllaços:
[1] https://comunitatdstsc.diba.cat/wiki/FAM%203.0%20-%20Scaffolding
[2] https://comunitatdstsc.diba.cat/wiki/fam-generacio-del-model-de-dades
[3] http://www.querydsl.com/