# MongoDb

## Librairies

- gson
- Mongo-java-driver

## (dé)Sérialisation JSON

Gson va permettre de sérialiser et désérialiser pour passer d'objet java à JSON et inversement :

```
public class DBOAdapter {
    private static Gson gson;

    private static Gson getGson() {
        if (gson == null) {
            GsonBuilder gsonBuilder = new GsonBuilder();
            gsonBuilder.registerTypeAdapter(ObjectId.class, new ObjectIdAdapter());
            gsonBuilder.excludeFieldsWithoutExposeAnnotation().setDateFormat("MMMM
dd, yyyy HH:mm:ss");
            gson = gsonBuilder.create();
        }
        return gson;
    }

    public static <T extends Model> T dboToModel(DBObject dbObject, Class<T> clazz)
{
        gson = getGson();
        String json = gson.toJson(dbObject);
        return gson.fromJson(json, clazz);
    }

    public static BasicDBObject objectToDBObject(Object object) {
        BasicDBObject obj = (BasicDBObject) JSON.parse(getGson().toJson(object));
        return obj;
    }

    public static DBObject[] objectToDBObjectArray(Object object) {
        BasicDBObject[] objects = new BasicDBObject[] { objectToDBObject(object) };
        return objects;
    }
}
```

### Gestion de l'objectId mongoDB

```
public class ObjectIdAdapter
        implements JsonSerializer<ObjectId>, JsonDeserializer<ObjectId> {
```

```
    @Override
    public JsonElement serialize(ObjectId id, Type typeOfT,
JsonSerializationContext context) {
        JsonObject jo = new JsonObject();
        jo.addProperty("$oid", id.toHexString());
        return jo;
    }

    @Override
    public ObjectId deserialize(JsonElement json, Type typeOfT,
JsonDeserializationContext context) throws JsonParseException {
        try {
            return new ObjectId(json.getAsJsonObject().get("$oid").getAsString());
        } catch (Exception e) {
            return null;
        }
    }

}
```