

PART 1- DAO Layer testing

■ War A.2_JDO-DAO_BooksCopies_Server This is an example for testing the insertion, selection, update and removal of a set of related objects. es.deusto.ingenieria.sd.datanucleus.jdo Book inherits from Product (Inheritance). A Book has many DAOTest_Copies.java Copies (1:N). A Copy is stored in Shelf. A Shelf can store many es.deusto.ingenieria.sd.datanucleus.jdo.data Copies (M:N). Book.java D Copy.java Creating copies means adding the copy to the shelf where it will ▶ I Product.java be stored. Shelf.java ▶ Æ es.deusto.ingenieria.sd.datanucleus.jdo.remote es.deusto.ingenieria.sd.org.datanucleus.jdo.dao ▶ IProductDAO.java ▶ I ProductDAO.java **DAO Pattern and Datanucleus** Edit datanucleus.properties There is a class ProductDAO.java (implementing IProductDAO) where all the coding related to Persistence is isolated. Notice the annotation in the domain objects @ PersistenceCapable(detachable = "true") Notice the code in DAOTest_Copies.java. No reference, at all, at any implementation details. Notice the annotation in the fields that need to be retrieved (because they are not fetched by default) @Persistent(defaultFetchGroup="true") In code, in the retrieval methods in the DAO layer = pm.getFetchPlan().setMaxFetchDepth(3); - Creating 2 maw products ...

Storing an object: Product: Sony Xperis [A standard mobile from Sony]
Storing an object: Product: Sony Xperis [A standard mobile from Sony]
Storing an object: Product: Sony Xperis [A standard iPad from Apple]
Storing an object: Book: J.R.R. Tolkiam - The Mobbit [Basic Edition]
Storing an object: Book: Bram Stoker - Draculs [Paperhack Edition]
Retrieving an Extent for Products.

* Mixture of Products and Books ...
Product: Sony Xperis [A standard mobile from Sony]
Product: Sony Xperis [A standard mobile from Sony]
Product: Sony Xperis — The Mobbit [Basic Edition]
Books Bram Stoker * Draculs [Paperhack Edition]

* Querying a Product: Sony Xperis

Otaching and ugdating a product: Product: Sony Xperis [A standard mobile from Sony]
- Attaching an Extent for Books.

Updating book copies (new copy in new shelf): Book: J.R.R. Tolkien - The Mobbit [Basic Sony New York of the Sony In New Sony In New Shelf added: Book: J.R.R. Tolkien - The Hobbit [Basic Edition]

Retrieving an Product: Oracula
Retrieving updated book: Book: J.R.R. Tolkien - The Hobbit [Basic Edition]
Humber of copies of this book: 2
Copy: 6 [Brasonable]in shelf: First Shelf
Copy: 6 [Brasonable]in shelf: First Shelf
Querying # Product: The Mobbit
Retrieving updated book: Book: J.R.R. Tolkien - The Mobbit [Basic Edition]
Querying # Shelf: First Shelf
Adding a new copy to an existing shelf: First Shelf
Adding a new copy to an existing shelf: First Shelf
Adding a new copy to an existing shelf: First Shelf
Adding a new copy to an existing shelf: First Shelf
Adding a new copy to an existing shelf: First Shelf
Adding a new copy to an existing shelf: First Shelf
Adding a new copy to an existing shelf: Book: J.R.R. Tolkien - The Mobbit [Basic Edition]
A product deleted from the DB.
A product deleted **Executing the example** Start MySQL (mysqld --console --explicit_defaults_for_timestamp) Start MySQL Workbench Create the database schema using the script in db folder Execute ant run Use ant deleteschema for reseting the db tables.



PART 2 - RMI + DAO Integration

<u> </u>	4
ProductFinderServer.java is a small RMI server that will use IProductDAO.	4.2_JDO-DAO_BooksCopies_Server
Edit the remote interface IProductFinder and notice that we are tranferring JDO domain objects, not special hand-made DTO. public interface IProductFinder extends Remote { public List <product> getProducts() throws RemoteException; }</product>	
	▶
Execute ant export in order to generate the library needed for the RMI client side of the application. This jar file must contain not only the remote interface, but also Product.class Sample Client source code: List <product> products = rsl.getService().getProducts();</product>	runServer: [iava] * Storing an object: Product: iPad [A standard iPad from Apple] [iava] * Storing an object: Book: Bram Stoker - Dracula [First Edition] [iava] Server *//127.8.8.1:1899/InventoryService* active and waiting
Launch the rmiregistry (util folder)	
Start the server using ant runserver	
Go to the Client Project	4.2_JDO-DAO_BookCopies_Client
Do not forget to copy the InventoryService.jar library	
Run ant compile	
Compile the client and ant run	What happens? It seems to be a very very simple omission in our code ??????
	Edit the code Product.java. Any idea? Do not forget to revise Copy and Shelf.
Client-side of the Application	Server-side of the application
<pre>client: [java] Client looking for service : //127.0.0.1:1099/InventoryService [java] Mixture of Products and Books [java] Product: iPad [A standard iPad from Apple] [java] Book: Bram Stoker - Dracula [First Edition] [java] Copy details: 30, Perfect, almost new [java] In Shelf: First Shelf [java] Copy details: 45, Scratches on front cover [java] In Shelf: First Shelf BUILD SUCCESSFUL</pre> Table 11 corends	Tunberver [lawa]
Total time: 11 seconds	