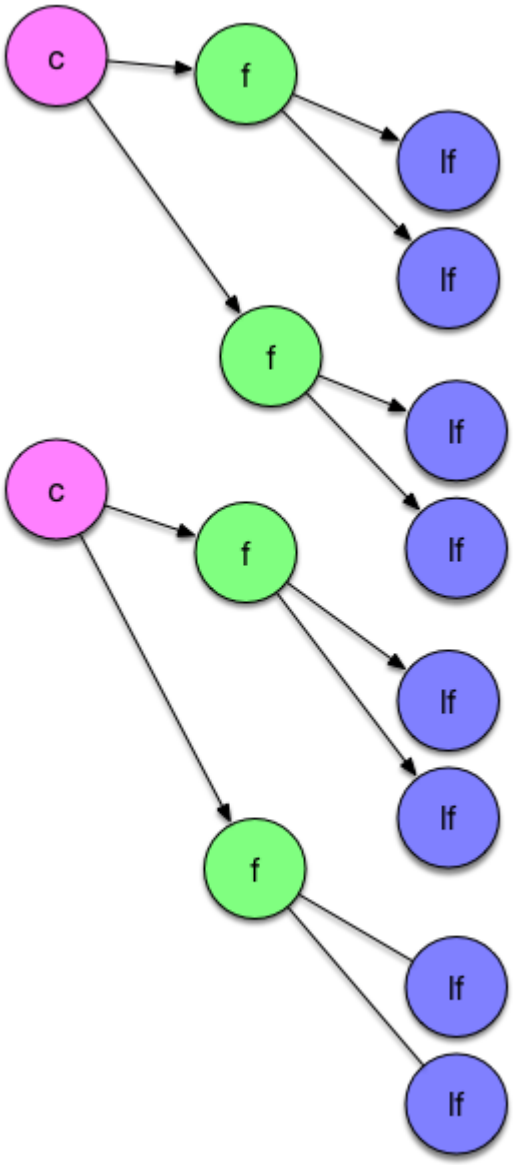
OpenSessionInView es uno de los AntiPatrones más clásicos de Java Enterprise y esta intimamente ligado a la capa de persistencia cuando utilizamos Hibernate o JPA. ¿Que problemática aborda este patrón y que problemática termina generando?. Vamos a explicarlo a detalle.

# OpenSessionInView como patrón

Cuando nosotros utilizamos un framework de persistencia como Hibernate o una especificación de persistencia como JPA . Usamos estas tecnologías para cargar un grafo de objetos en memoria. Por ejemplo Cliente , Factura y LineasFactura.

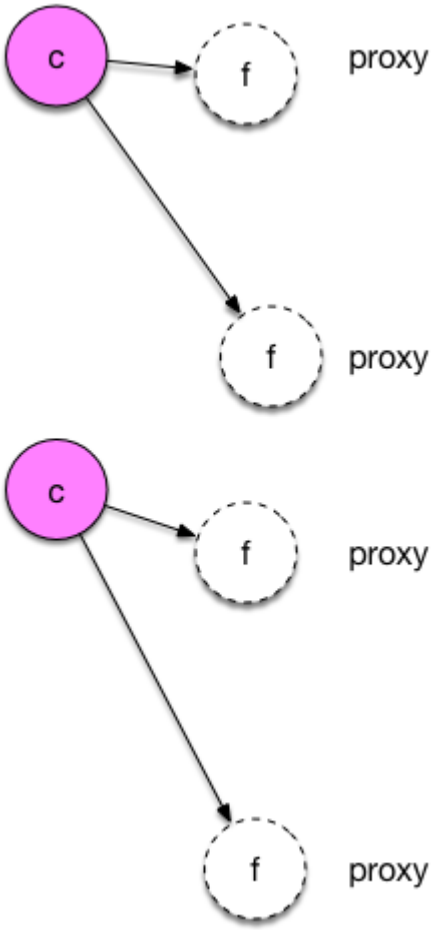



Sin embargo las cosas no siempre funcionan así. Ya que por ejemplo si hemos ejecutado una consulta del estilo.

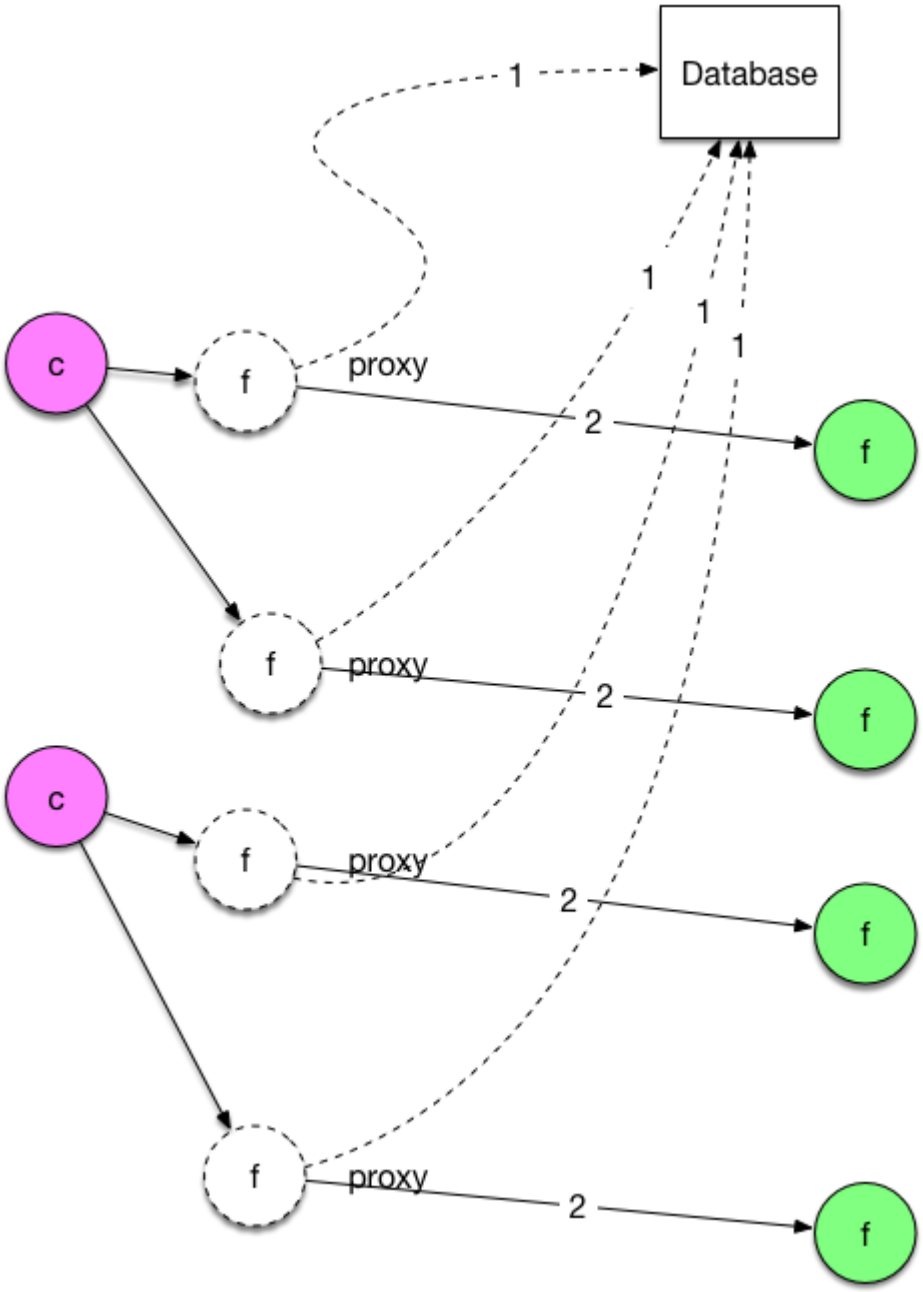
select c as Cliente c

Utilizando JPA Query Language o Hibernate Query Language el framework no cargará evidentemente el grafo completo sino que se encargará de cargar el primer nivel y generará

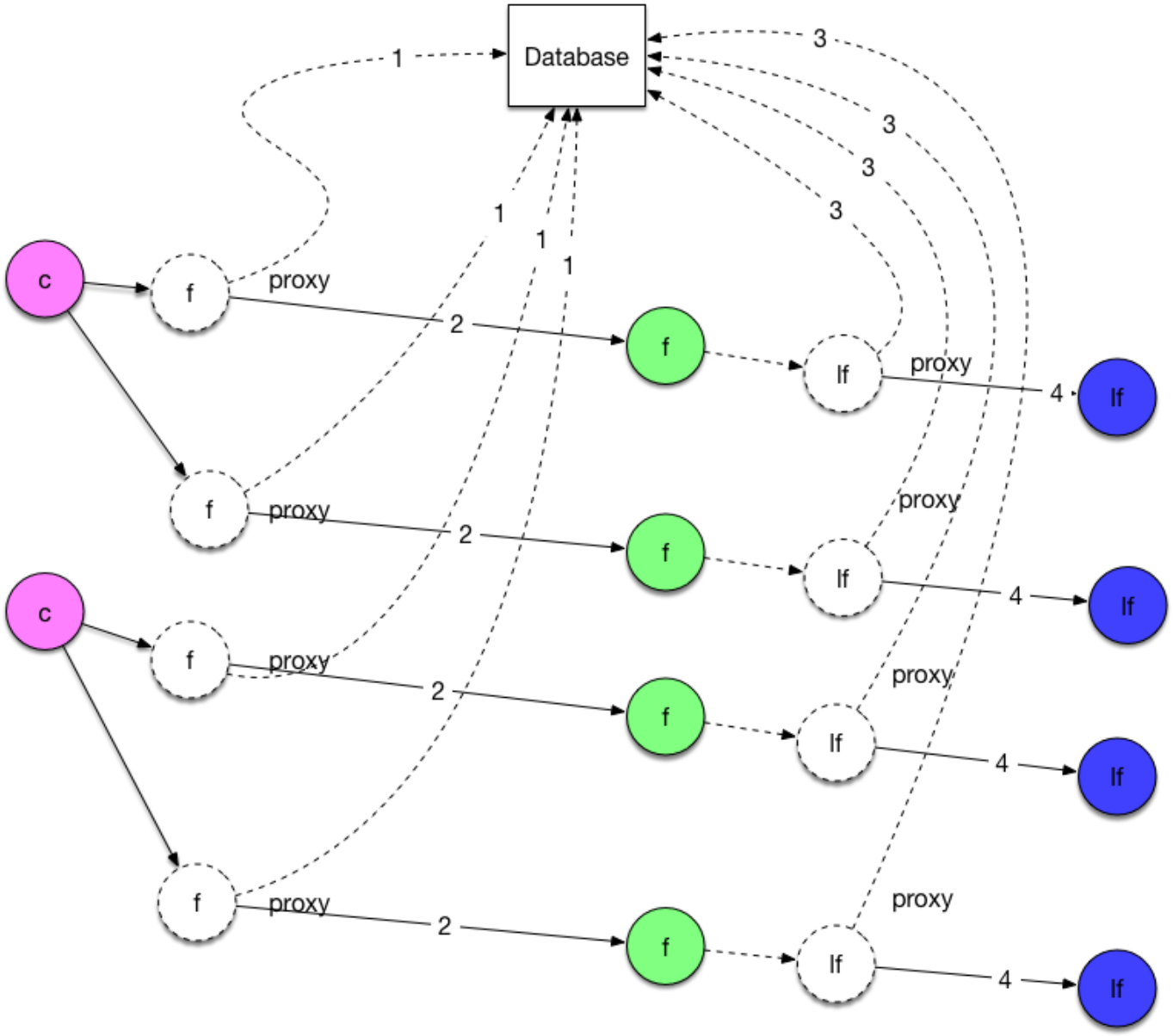
proxies para cargar el segundo nivel.



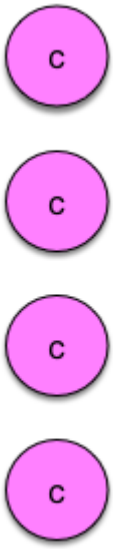

Este quiere decir que las facturas no están cargadas a nivel del grafo , lo que tenemos son una serie de objetos “proxy que simulan ser las facturas”. Cuando en algún momento le solicitemos al framework el acceso a las facturas este se encargará de cargar las facturas dese la base de datos.



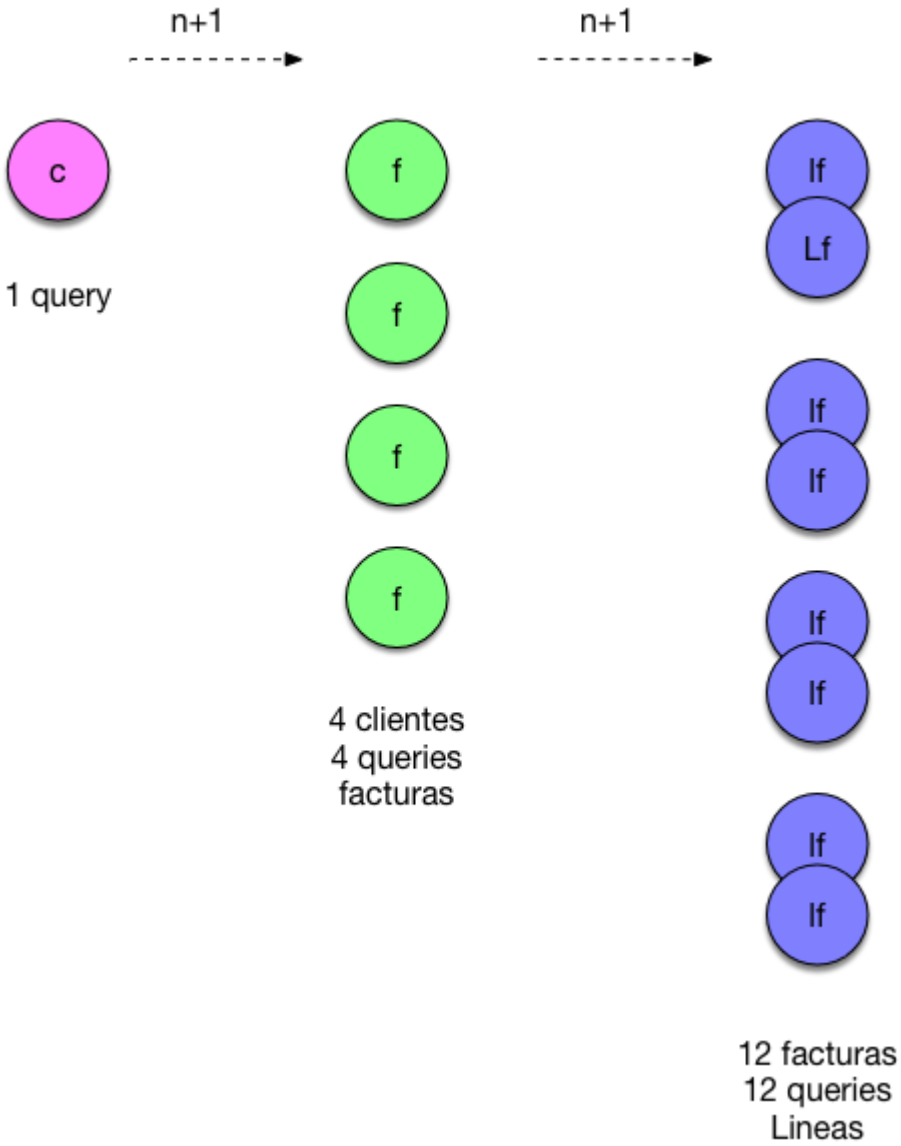

Lo mismo pasará cuando desde las facturas deseemos acceder a los datos de las lineas de factura.




En estas situaciones hay que tener cuidado ya que sino configuramos correctamente las relaciones y las queries que ejecuta el framework de persistencia podemos encontrarnos con un problema de n+1 queries. Es decir alguien solicita un listado de clientes




Eso en primer momento es una sola consulta. Pero si accedemos a las facturas de cada cliente y tenemos n clientes, entonces el framework realizará una consulta por cada cliente para obtener sus facturas .Lo mismo sucederá para las facturas al obtener las lineas.

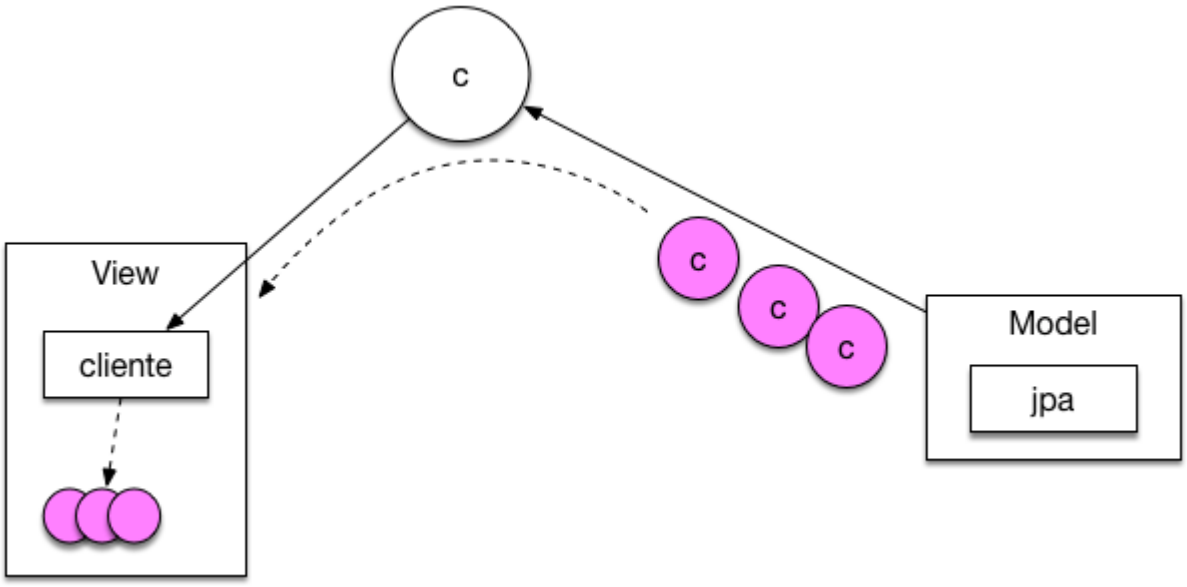



## OpenSessioninView y MVC

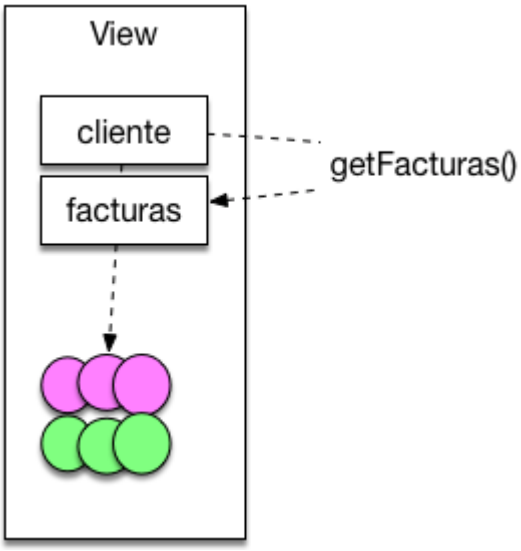
Muchas veces el problema viene de las personas que desarrollan la capa de presentación en un modelo MVC. Supongamos que la capa de backend realiza una consulta como la siguiente.

select c from Cliente c;

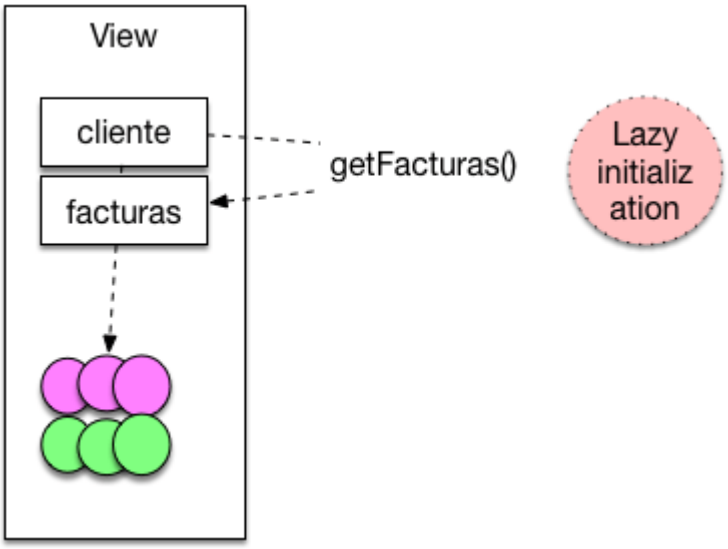
En principio en el modelo MVC podremos tener una capa de modelo que seleccione los clientes y los muestre en una vista.



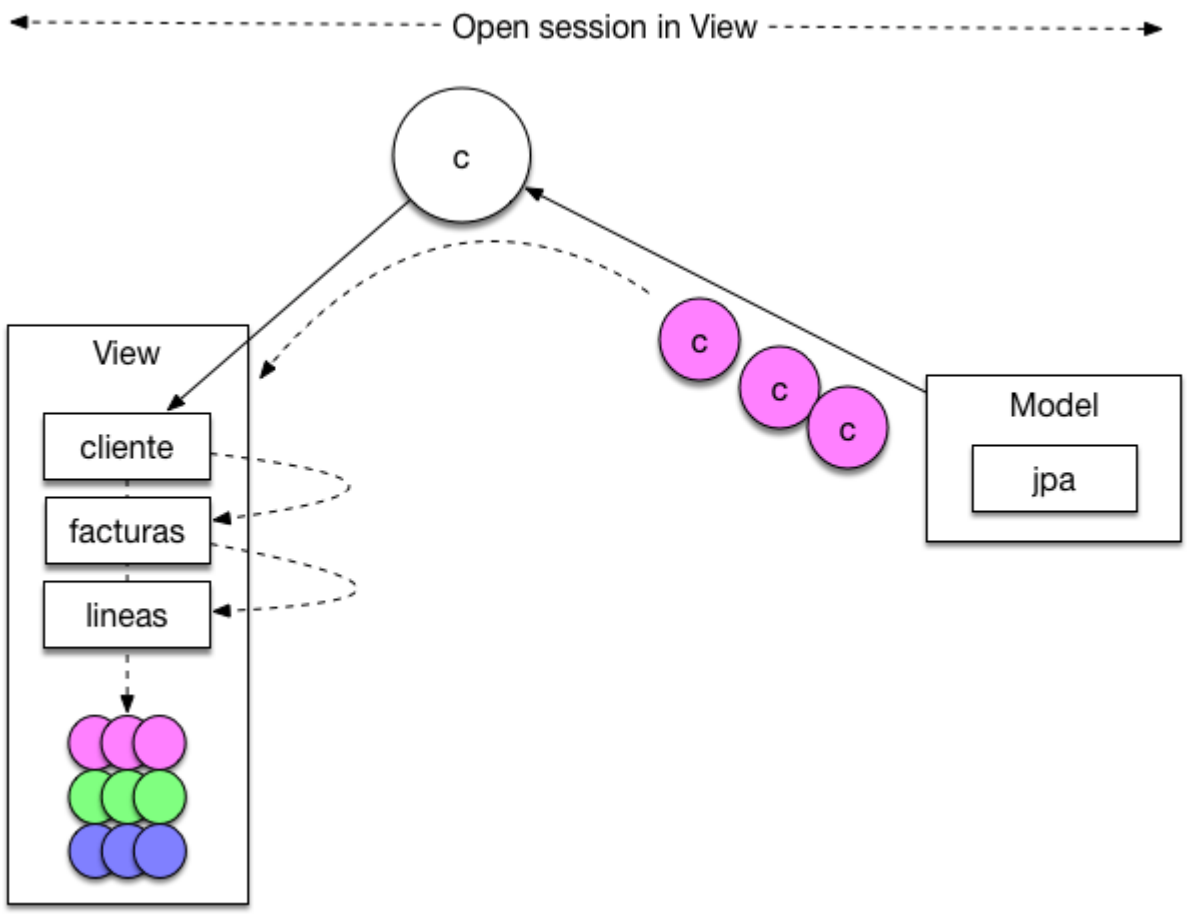

El problema surge cuando alguien que construye la vista no solo quiere mostrar los clientes. Sino que quiere mostrar ademas las facturas.Entonces es típico que en la vista intentemos algo como lo siguiente:




Aquí surgirá un problema ya que el cliente tiene proxies como facturas y ya hace tiempo que se desacoplo de Hibernate o JPA.




Para evitar este problema se opta por el patrón OpenSessionInView que permite tener la session de hibernate o el entityManager de JPA abierto hasta que la vista se resuelve.




Esto permitirá a un desarrollador de capa de presentación mantener la session abierta y generar un problema de n+1 queries sin practicamente darse cuenta. Por ejemplo un listado que imprima 100 clientes con sus facturas y sus lineas generará:

- 1 consulta para los clientes
- 100 consultas para las facturas de cada cliente  (esto devuelve 1000 facturas por ejemplo)
- 1000 consultas para las lineas de factura.

Si por ejemplo tenemos 10 usuarios que concurrentemente solicitan esta pagina JSP estaremos lanzando mas de 10.000 consultas a la base de datos para mostrar una vista. El patrón OpenSessionInView es un antipatrón y hay que evitar su uso.

Otros artículos relacionados

1. JPA Lazy fetching proxies y rendimiento
2. Un ejemplo de JPA Entity Graph
3. JPA Criteria API , un enfoque diferente
4. JPA