1. En la entrega de servicios tratar de solucionar el siguiente problema con las categorías: En el caso de que se añada un nuevo idioma al sistema, inicialmente todas las categorías tienen ese nuevo idioma a nulo, lo que puede dar error.

2. ¿Un handyWorker podría solicitar más de una vez una tarea? NO HAY CLASE ASOCIACIÓN. Podría interpretarse como entidad ya que puede cambiar el estado a REJECTED y volverlo a solicitar, cambiando así el estado y por lo tanto en ese caso sería una entidad. Pero esto depende de la interpretación de los requisitos.

3. Tras finalizar cada uno de los métodos anotados con @Test, se hace internamente y de manera automática un rollback en la base de datos. Tener esto en cuenta para los tests: Dentro de una misma clase de test, cuando acaba un método test y pasa al siguiente, en este último método de test la base de datos está como si estuviera recién creada, sin ningún cambio.

4. Si recuperamos un objeto de la base de datos y le aplicamos un “setAtributo”, automáticamente este objeto es actualizado en la base de datos, aunque no hayamos hecho ningún “save”. Esto no pasa cuando creamos un objeto nuevo y, antes de hacer “save”, inicializamos sus atributos con “setAtributo”.

5. La etiqueta @Transactional que se pone en los servicios es la de la librería javax.transaction.Transactional.

6. La etiqueta @Transactional que se pone al crear los test es la de la librería org.springframework.transaction.annotation.Transactional.

7. Posibles causas del error “Load Application Context”:
    a. Los métodos constructores de los servicios deben ser públicos.
    b. Alguna query de algún repositorio está mal construida.

8. Revisar la visibilidad de todos los métodos:
    a. Los métodos que vamos a usar directamente en los controladores: public.
    b. Los métodos que vayamos a reutilizar entre distintos servicios pero que no se usen directamente en controladores: protected.
    c. El resto private.

9. **No** ejecutar nunca todos los tests de golpe haciendo click derecho en la carpeta services > Run as Junit Test. Si se hace de esta forma algunos tests que están bien pueden dar error y algunos tests que están mal pueden dar como correctos.

10. Para hacer los mockups, si hay una vista en la que puedan acceder todos los actores del sistema, la url que le llega a la vista sería borrando el rol de la url, por ejemplo: **announcement/list.do,** en vez de poner **announcement/administrator/list.do.**

11. En el caso de una vista en la que puedan acceder varios actores, pero no todos, en la URL de dicha vista habría que poner los roles separados por comas. Por ejemplo: **announcement/customer,reviewer/list.do**

12. Traducción
    a. FixUpTask – Chapuza
    b. HandyWorker – Manitas
    c. Referee – Árbitro
    d. Report – Informe
    e. Endorsement – Reseña
    f. Customisation – Configuración

13. El hash de las contraseñas no debe estar en los controladores ni en los servicios, debe estar en las vistas. Esto se debe a temas de seguridad.

14. Hacer un solo formulario de registro de actor que va a servir para registrar a todo tipo de actor. Debe haber un mismo formulario donde el modelAttribute sea un objeto tipo Actor y la url de formulario debe ser única (es decir un solo formulario de edición de Actor y un solo controlador para ese formulario.). Para que esto funcione hay que quitar abstract del modelo de dominio Actor.

15. En las vistas de listados, no se debe mostrar todos los campos de las entidades, porque sino no tendría sentido la vista de display de dicha entidad.

16. En las etiquetas display:column es mejor usar el atributo titleKey en lugar de title. Por ejemplo se podría sustituir el siguiente fragmento:

```
<spring:message code="fixUpTask.ticker" var="tableTicker"/>
<display:column property="ticker" title="${tableTicker}"/>
```

Por este otro:

```
<display:column property="ticker" titleKey=" fixUpTask.ticker "/>
```

17. Los momentos que haya que inicializar con el momento de creación de la entidad, deben inicializarse en el método create y **no** en el save.

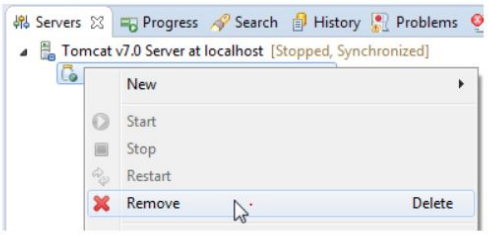
18. Si en un controlador se llaman a varios servicios hay que tener completamente claro que, si uno de los servicios peta y hace rollback, no tiene ningún conflicto con el resto de llamadas a los servicios, ya que el contexto transaccional envuelve a cada llamada por separado.

19. SOLO CON PAGINACIÓN EXTERNA: La variable **requestURI** de los listados tiene que ir siempre sin ningún parámetro GET. Si, por ejemplo, la URI para acceder a un listado es "*complaint/referee/list.do?refereeId=1212",* la **requestURI** que se le debe pasar a la tabla sería "*complaint/referee/list.do"*

20. SOLO CON PAGINACIÓN EXTERNA: Si se va a usar una query para listar y además se quiere poder ordenar por algunos de los atributos de forma dinámica (es decir, que en la misma tabla del listado puedas elegir por qué atributo quieres ordenar los registros), dicha query **tiene que cumplir** que el FROM debe ser de la tabla de los mismos objetos que se quiere listar. Si por ejemplo quiero listar Complaints, el FROM tiene que se Complaint. No se puede poner otra cosa, aunque se use JOIN o algo parecido, ya que al intentar ordenar va a petar.

21. Para el error does not exist or is not a readable directory

- probably at the end of the last line that has to be removed is `</Host>` (don't remove `</Host>` !)
- Go to Servers tab > right click on the project(s) > Remove > OK




- Right click on the project > Run As > Run on Server > Finish

Now it should work!
If you check the server.xml file, you'll see that a new correct " `<Context ... />` " line was generated.

share improve this answer