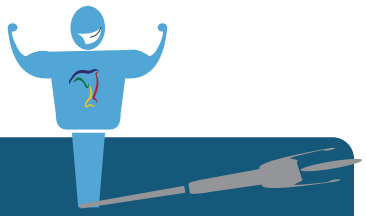
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

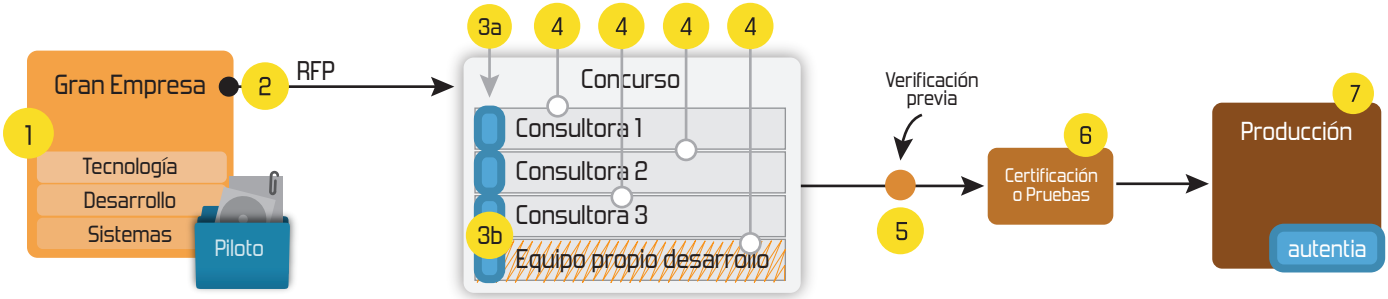
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

adictos altrabajo

autentia real business solutions
Hosting patrocinado por ENREDADOS

E-mail:
Contraseña:
Entrar
Deseo registrarme
He olvidado mis datos de acceso

Inicio Quiénes somos Tutoriales Formación Comparador de salarios Nuestro libro Charlas Más

Estás en: Inicio Tutoriales Cómo listar una entidad en wuija con prefiltrado

DESARROLLADO POR:
Alberto Barranco Ramón

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Catálogo de servicios Autentia

Anuncios Google Tutorial Excel Tutorial Criteria T SQL Tutorial

Fecha de publicación del tutorial: 2009-02-26

Share | Regístrate para votar

Cómo listar una entidad en wuija con prefiltrado

0. Índice de contenidos.

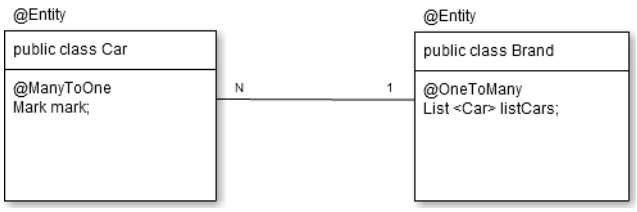
- 1. Introducción.
- 2. Añadir funcionabilidad extra y genérica sin alterar lo anterior.
- 3. Conclusiones.

1. Introducción

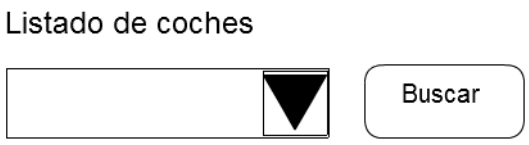
En este pequeño tutorial vamos a tratar de clarificar un poco cómo se podría prefiltrar un listado sobre una entidad en wuija.

Vamos a ponernos en contexto.

- ¿Qué tenemos hasta ahora?
  - - Una entidad (Car)anotada con @Entity persistida en base de datos. Esta entidad tiene un @ManyToOne Brand brand. Es decir un coche puede tener una marca, y una marca puede tener varios coches.



  - - Nuestro controlador (ListCars) que extiende de ListController. ListController es un controlador genérico válido para cualquier tipo de clase (ListController <T extends Object>) que tiene en su interior un Widget (CriteriaQueriedDataTable<T>) que lo que hace al inicializarse (en el init()) es pintar una tabla con todos los registros que hay en base de datos de la entidad que le hemos pasado como parámetro poniendo en la primera fila de la tabla los atributos de la entidad. Además nos pinta encima un buscador que nos permite buscar por los atributos (sean del tipo que sean) que le configuremos (configureProperties()) de la entidad y filtrar los resultados... <--- How cool is that ¿?



Modelo	Color	Puertas	Marca
Broma	Rojo	3	Joke
Scoby	Azul	5	Acme
Chachi	Blanco	3	Autentia
Sport	Rojo	5	Autentia
Tango	Blanco	5	Acme

- ¿Qué queremos?
  - - Listar una entidad (Cars)con "CriteriaQueriedDataTable" pero sólo queremos que nos salgan los coches de una marca en concreto, por ejemplo los de la marca Autentia y que sólo se pueda re-filtrar por los que se muestran en la tabla, es decir, el buscador sería un filtro sobre un resultado ya filtrado.

Últimas Noticias

Comentando #AID2010. Agil Industrial Day 30 Nov 2010
Autentia Head Hunting - ¡¡Primeros contratos!!
XIII Charla Autentia - AOS y TDD
XII Charla Autentia - LiquiBase - Material
XI Charla Autentia - Mule - Vídeos y Material

Histórico de NOTICIAS

Últimos Tutoriales

JQuery: Paginación
Girillo TTS: Una aplicación Android para evitar el uso del móvil durante la conducción
Ejemplo básico de Spring MVC Portlet
JCaptcha: Análisis Técnico en aplicativos reales
Zen-coding: una nueva forma de escribir código HTML

Últimos Tutoriales del Autor

Construcción de componentes en wuija por composición
Cómo subir tutoriales a Adictos
Ejemplo de uso con JSF 2.0, Primefaces e Hibernate

Síguenos a través de:

Últimas ofertas de empleo

2010-10-11
Comercial - Ventas - SEVILLA.
2010-08-30
Otras - Electricidad - BARCELONA.
2010-08-24
Otras Sin catalogar - LUGO.
2010-06-25
T. Información - Analista / Programador - BARCELONA.

2. Añadir funcionabilidad extra sin alterar lo anterior

En el controlador genérico nos encontramos la siguiente lógica en el método init:

```
view plain print ?
01. final EntityCriteria criteria = new EntityCriteria(entityClass);
02. criteria.setMatchMode(MatchMode.ANY);
03. entityDataTable = new CriteriaQueriedDataTable<T>(propertiesOfList, criteria, new AdvancedQuery(
04.         propertiesOfQuery, criteria), dao, csvReportsService);
```

Podemos ver como al CriteriaQueriedDataTable y al AdvancedQuery (buscador) se le está pasando exactamente la misma criteria, que además está vacía, ya que la acabamos de crear, por lo que nos va a pintar todos los registros de la base de datos que se correspondan con la entidad.

Para añadir nuestro filtrado vamos a crear en nuestra Clase ListController un método que permita settear una precondición. Quedaría de la siguiente forma.

```
view plain print ?
01. protected void setPreCondition(EntityCriteria criteria) {
02.     preCondition = criteria;
03. }
```

Ahora es cuando en nuestra clase concreta tenemos que construir dicho filtrado o precondición. Lo vamos a hacer de la siguiente forma.

```
view plain print ?
01. private EntityCriteria getCriteria() {
02.     final EntityCriteria criteria = new EntityCriteria(Car.class, MatchMode.ALL);
03.     Junction brand = new Junction(MatchMode.ANY);
04.     brand.add(new SimpleExpression("brand", Operator.EQUALS, "Autentia" ));
05.     criteria.add(brand);
06.     return criteria;
07. }
```

Tal y como vemos tenemos un EntityCriteria para la clase Car, en donde le indicamos el MatchMode ALL. Esto quiere decir que todas las condiciones que se añadan a esta EntityCriteria van a ir separadas por un AND. Es decir, obtendrá los resultados que cumplan todas las restricciones. A continuación instanciamos un Junction con el MatchMode ANY. Esto quiere decir que se recogerán los resultados que cumplan alguna de sus condiciones (OR). En nuestro caso le vamos a fijar una condición al Junction, esto es, una SimpleExpression donde indicamos que la marca (brand) tiene que ser igual a "Autentia". Por último añadimos el Junction al Criteria.

Para que este prefiltrado esté presente en la clase genérica (ListController) vamos a utilizar el método que nos creamos anteriormente en el init de nuestra clase concreta de la siguiente forma:

```
view plain print ?
01. @Override
02. @PostConstruct
03. void init() {
04.     setPreCondition(getCriteria());
05.     super.init();
06. }
```

Ahora en el método init de ListController tendremos que preguntar si tenemos fijada ya una precondición. La lógica sería parecida a lo siguiente:

```
view plain print ?
01. if (preCondition == null){
02.             // hacemos lo que se hacía antes
03.         }
04.         else {
05.             final Junction modifiedByUserCriteria = new Junction();
06.             preCondition.add(modifiedByUserCriteria);
07.             entityDataTable = new CriteriaQueriedDataTable<T>
    (propertiesOfList, preCondition, new AdvancedQuery(
08.                     propertiesOfQuery, modifiedByUserCriteria), dao, csvReportsService);
09.         }
```

Este punto es el más importante quizás. Si nos fijamos en la línea 5 estamos instanciando un Juntion vacío. En la línea 6 se lo añadimos a la criteria (preCondition), con lo que tenemos 2 Juntion. Juntion1 AND Juntion2. Como el segundo está vacío la única restricción que tenemos es la del primero, es decir el prefiltrado por marca = Autentia. Al pasarle esta precondición al CriteriaQueriedDataTable, nos estará pintando justamente los resultados que queremos. Al buscador (AdvancedQuery), le pasamos ese Junction 2 (modifiedByUserCriteria) para que si se deseara aplicar otro filtrado, los resultados que se van a mostrar cumplirán los 2 filtrados.

3. Conclusiones

La idea de hacer funcionabilidad genérica es muy buena. Por eso cuando la completamos, no debemos vincular lo genérico a lo específico.
También es importante dejar la lógica antigua tal y como estaba. Para esto tenemos dos opciones:
1. Crearnos una nueva clase que extienda de la genérica, si es que la funcionabilidad tiene suficiente importancia como para estar en otra clase. De esta manera conservaríamos la lógica anterior y no tendríamos riesgo de haber alterado otras cosas.
2. Añadir la funcionabilidad en la propia clase genérica, ya que sigue siendo funcionabilidad genérica, teniendo mucho cuidado de no alterar lo anterior.

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**
Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

Comentarios

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2010 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS XML RSS XML ATOM