Ejercicios sesión 3: Mapeado entidad-relación, tablas

Índice

En esta sesión de ejercicios vamos a practicar con el mapeado de entidades en tablas y columnas. El mapeado de relaciones lo dejamos para la sesión siguiente.

Vamos a continuar con el ejemplo anterior en el que definíamos autores y mensajes. En esta sesión de ejercicios añadiremos algunas funcionalidades más, introduciendo la posibilidad de que la aplicación que estamos construyendo sirva para anotar la autoría de diversos tipos de recursos y etiquetarlos. Este escenario nos servirá para practicar el mapeado de las relaciones de herencia, introduciendo dos tipos de recursos: páginas HTML y documentos PDF.

1. Creación de la nueva entidad Tag

Vamos a añadir nuevas entidades a las que ya tenemos. Vamos a definir recursos y etiquetas. En la próxima sesión relacionaremos los autores con los recursos, definiendo una relación uno-a-muchos entre autores y recursos y una relación muchos-a-muchos entre recursos y tags.

La idea es definir las bases para una aplicación web con la que los autores pueden crear mensajes y recursos, y añadir etiquetas a estos últimos.

1. Comencemos por crear el nuevo proyecto `jsf-sesion3-web` copiando y pegando el proyecto del ejercicio anterior. Recuerda del ejercicio 2 que hay que tocar el fichero `.settings/org.eclipse.wst.common.component` y modificar a mano el nombre del proyecto en un par de elementos XML.

2. Una vez creado el nuevo proyecto, tenemos que crear la entidad `Tag`. Las instancias de esta entidad van a servir para etiquetar mensajes y recursos con una cadena de texto. La entidad debe tener los siguientes atributos:

- `int id`: identificador de la etiqueta (autogenerado)
- `String cadena`: cadena de texto que representa la etiqueta. Puede contener espacios.
- `java.util.Date fCreacion`: fecha en la que se creo por primera vez la etiqueta
- `int ocurrencias`: número de documentos etiquetados con ese tag

Algunos ejemplos de cadenas de etiquetas: `'curso'`, `'java'`, `'tutorial'`

Define la clase entidad `Tag` con los atributos anteriores. Define un constructor vacío `protected` y los setters y getters necesarios. En la aplicación que ya tienes montada del ejercicio anterior, crea una función privada `addTag()` (sin parámetros) que cree una nueva `Tag` y que la haga persistente en la base de datos. Despliega la aplicación web, ejecuta la nueva acción y comprueba la nueva tabla con el administrador de MySQL. ¿Qué tipos de columna se han definido?

3. Define la clase `TagEAO` en el mismo paquete con los métodos que vamos a usar para trabajar con la entidad persistente:

- `Tag createTag(String cadena)`: crea una etiqueta, la inicializa con la fecha actual

del sistema, le pone el numero de ocurrencias a 0 y la hace persistente.
- `Tag cambiaCadena(Tag tag, String cadena): sustituye la etiqueta de la cadena por una nueva.`
- `List<Tag> listaTags()`: devuelve todas las etiquetas creadas

2. Creación de la nueva entidad Recurso

Vamos ahora a realizar la construcción de la entidad con la que representamos los recursos.

1. Crea un tipo embebido `DatosFichero` con los siguientes atributos:

- `String nombre`: nombre del fichero (incluyendo su ruta y extensión)
- `Double size`: tamaño en Kbytes del fichero
- `java.util.Date fCreacion`: fecha de creación del fichero

2. Crea la nueva entidad `Recurso` con los siguientes atributos:

- `int idRecurso`: identificador del recurso (autogenerado)
- `DatosFichero fichero`: datos del fichero
- `String resumen`: descripción larga con el contenido del recurso. Debe ser un tipo LOB.
- `int visitas`: número de veces que se ha consultado el recurso

Define un constructor vacío y los getters y setters necesarios. En la aplicación que ya tienes montada, crea una función `addRecurso()` (sin parámetros) que cree un nuevo `Recurso` y que lo haga persistente en la base de datos. Despliega la aplicación web, ejecuta la nueva acción y comprueba la nueva tabla con el administrador de MySQL. ¿Qué tipos de columna se han definido?

3. Creación de las entidades hijas

Crea las nuevas entidades `PaginaHtml` y `FicheroPDF`, subentidades de `Recurso`. Utiliza el método de la tabla única, y escoge tu mismo la columna y el valor discriminante.

En la entidad `PaginaHTML` define los siguientes atributos:

- `String contenido`: Contenido HTML de la página. Debe ser un tipo LOB
- `boolean imagenes`: True o false, dependiendo de si la página enlaza a imágenes

En la entidad `FicheroPDF` define los siguientes atributos adicionales:

- `int páginas`: número de páginas
- `String titulo`: título del PDF

1. Implementa ambas clases y modifica la función `addRecurso` para comprobar la tabla creada en el mapeado.

2. Implementa una clase `RecursoEAO` que implemente los siguientes métodos de acceso a las entidades persistentes:

- Recurso createRecurso(String nombreFichero, String resumen, TipoRecurso tipo)
- void updatePaginaHTML(PaginaHTML pagina, String contenido, boolean imagenes)
- void updateFicheroPDF(FicheroPDF pdf, int paginas, String titulo)
- Recurso findRecursoById(int idRecurso)
- Collection<Recurso> findRecursoByName(String name)

4. Interfaz web para etiquetas

Define un servlet `doActionTag`, similar al implementado en la sesión 2, con el que podamos probar las distintas funcionalidades del `TagEAO`.

Define una página HTML con la que podamos introducir los parámetros necesarios para el servlet. Debes pedir la cadena del tag y una acción a realizar.

Implementa como mínimo las acciones:

- `add-tag`: añade una etiqueta
- `listar-tags`: lista todas las etiquetas
- `buscar-tag`: busca una etiqueta

Implementa una página JSP que muestre el resultado de cada acción.

5. Interfaz web para recursos

Haz lo mismo para el `RecursoEAO`.

Define una páginas HTML con las que podamos introducir nuevos recursos, llamando al método `createRecurso` del RecursoEAO. Implementa como mínimo las acciones:

- `add-pagina`: añade una página
- `buscar-pagina`: busca una página por el nombre

Implementa una página JSP que muestre el resultado de cada acción.