

Explorando el Repositorio de Contenidos Java Jackrabbit

CONEJO VELOZ



saul F. Fotolia

Con Jackrabbit ahorramos tiempo en el desarrollo web. Se trata de una implementación en código abierto del estándar Java Content Repository.

POR CARSTEN ZIEGELER

Las bases de datos, al igual que las estructuras de datos convencionales, no siempre son la solución perfecta para el desarrollo web. Una aplicación web, como puede ser un portal de servicios para el personal de una empresa, por ejemplo, necesita una serie de requisitos sobre cómo y cuándo pueden los usuarios acceder a los datos. Cada usuario tiene sus propias necesidades con respecto a los datos. Por ejemplo, a uno puede interesarle la búsqueda de determinados contenidos, mientras que otro puede querer que se le notifiquen ciertos eventos, pero ambos necesitan privilegios para la modificación de contenidos.

En resumen, no basta con ofrecer contenidos. El usuario de hoy día – así como el desarrollador de hoy día – espera que el contenido venga acompañado de una serie de servicios. Por ejemplo, las aplicaciones web suelen implementar control de acceso, funciones de búsqueda, revisiones y, aunque el desarrollador podría implementar todas estas funciones desde cero en la aplicación, es mucho más conveniente

la utilización de un método más eficiente.

La idea tras la API JCR para la tecnología Java (Java Content Repository) consiste en abstraer los servicios relacionados con los datos de la aplicación subyacente empleando una API estándar para el acceso a dichos servicios. Con un repositorio de contenidos evitamos la necesidad de reimplementar continuamente los servicios de datos con cada aplicación. En lugar de eso, la aplicación simplemente llama a una función a través de la API del repositorio.

El repositorio de contenidos combina algunas ventajas de los sistemas de archivos y de las bases de datos. De los sistemas de archivos, el repositorio adopta el almacenamiento jerárquico de archivos sin estructura y los permisos para el control de acceso. En lo que concierne a las bases de datos, el repositorio también soporta almacenamiento de datos estructurados, consultas, transacciones y comprobaciones de integridad. Los repositorios de contenidos soportan además funcionalidades

tales como el control de revisiones o el historial de cambios (Figura 1).

El Repositorio de Contenidos

La especificación completa del estándar para el Repositorio de Contenidos es un punto de partida excelente a la hora de familiarizarnos con la API del Repositorio de Contenidos de Java [1]. La idea es que la definición de un repositorio es independiente de las fuentes de datos subyacentes, los protocolos y la arquitectura. La API se divide en dos niveles. El nivel 1 ofrece la funcionalidad básica necesaria para el acceso de lectura, mientras que el nivel 2 soluciona los problemas relativos a la modificación de los datos almacenados.

La implementación de la referencia del JCR fue creada por Day Software, de cuyo relevo se encargó la *Apache Software Foundation*. Desde entonces, esta implementación se ha convertido en un exitoso proyecto de código abierto con el nombre de Apache Jackrabbit [2]. Junto al proyecto se ha creado también una importante comunidad, que conti-

núa impulsando su desarrollo. El repositorio de Jackrabbit es una rica implementación del estándar, con un completo juego de funciones de nivel 1 y nivel 2. Jackrabbit añade además funcionalidades extra, como la posibilidad de instalar un cluster del repositorio.

Con Jackrabbit se incluye una aplicación web que nos guía en los primeros pasos necesarios para la definición de un repositorio de contenidos. Dicho de otro modo, con esta aplicación disponemos de una interfaz que nos permite instalar nuevos repositorios.

Apache Jackrabbit soporta el acceso al repositorio a través de *WebDAV*, que hace más sencillo su montaje, la copia de cualquier tipo de archivos, la creación de directorios y la gestión de los contenidos del repositorio.

La Figura 2 muestra una perspectiva general del modelo del repositorio: Presenta una estructura jerárquica simple en forma de árbol con *n* niveles. La instancia principal es el repositorio, que puede contener una o más áreas de trabajo. A su vez, cada espacio de trabajo contiene una serie de elementos en



Figura 1: El Repositorio de contenidos auna las ventajas de las bases de datos y las de los sistemas de archivos.

forma de árbol, donde cada elemento es o bien un nodo o una propiedad. Cada nodo puede o no tener nodos hijos, así como de 0 a *n* propiedades para el almacenamiento de datos (ver el cuadro de texto *Tipos*). Una propiedad es de un tipo dado y contiene un tipo de dato

(cadena de caracteres, número, cadena binaria, etc).

Los nodos permiten el almacenamiento en forma jerárquica de, a modo de ejemplo, fotografías digitales bajo un nodo *fotos*. Otros nodos del repositorio representan los álbumes de fotos, que pueden a su vez contener subálbumes. Por ejemplo, todas la fotografías tomadas en Amsterdam en el año 2008 podrían residir bajo */fotos/2008/Amsterdam*. Se puede acceder a cada elemento, ya se trate de un nodo o de una propiedad, a través de una ruta que comienza por la raíz del repositorio. Por debajo del nodo *Amsterdam* hay fotografías, y cada foto tiene su propio nodo. Pero un repositorio es más que una simple colección de archivos y directorios. Las propiedades de cada nodo individual pueden incluir parámetros útiles para la aplicación web, como el flujo binario de la imagen, o parámetros como la fecha de captura de la fotografía y su localización. Es responsabilidad del desarrollador estructurar los datos en el repositorio de la mejor forma posible para la aplicación. La documentación del repositorio Jackrabbit, así como su wiki [3], nos ofrecen trucos y consejos para el modelaje de contenidos.

La API de Java

Para interactuar con el repositorio se requiere la realización de un par de pasos previos: Los desarrolladores de la aplicación deben definir primero una

Listado 1: Acceso al Repositorio

```
01 InitialContext jndiContext = ...
02 Repository repositorio =
    (Repository)jndiContext.lookup("MiRepositorio");
03
04 // Crear las credenciales
05 Credentials credenciales = new SimpleCredentials(
06     "carsten", "passwordsecreto".toCharArray());
07
08 // Abrir una sesión
09 Session sesion = repositorio.login(credenciales, "Área de Trabajo
    A");
```

Listado 2: Leyendo y Escribiendo en el Repositorio

```
01 // Tomar el álbum "/fotos/2008"
02 Node knoten = (Node)sesion.getRootNode().getNode("fotos/2008");
03 // Crear un nuevo álbum
04 Node album = knoten.addNode("Amsterdam");
05 // Añadir una propiedad
06 album.setProperty("Descripción", "Fotos de Amsterdam");
07 album.setProperty("publico_lectura", true);
08
09 // y guardar
10 sesion.save();
```

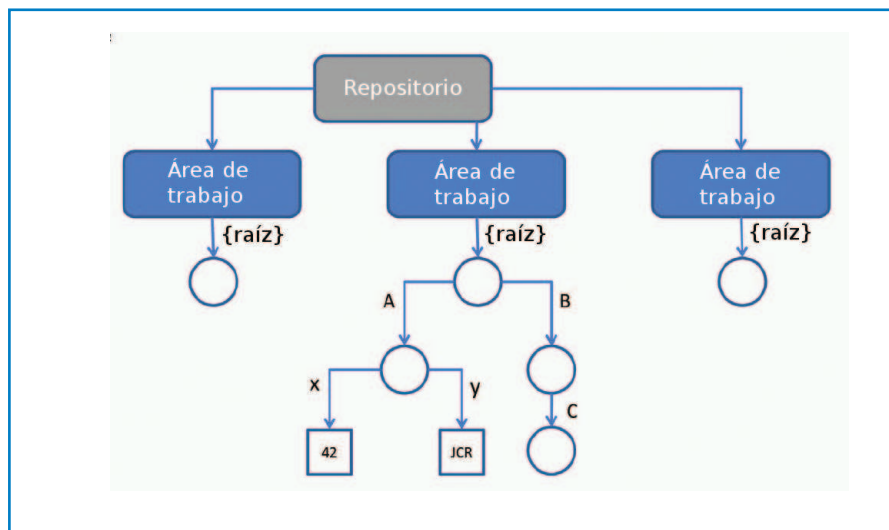


Figura 2: El modelo de repositorio: El área de trabajo contiene los nodos A, B y C. El nodo A tiene las propiedades x e y.

conexión al repositorio, para crear después una sesión para cada usuario individual. El resto de las acciones usarán esa sesión. El primer paso a dar, necesario para el acceso al repositorio, no está definido en el estándar. Dependiendo de cómo esté implementada la infraestructura, Apache Jackrabbit nos ofrece varios métodos. Uno de ellos usa JNDI (Java Naming and Directory Interface). El código del Listado 1 crea la sesión de un usuario ficticio.

El desarrollador puede hacer uso de la sesión para la consulta y la modificación de nodos. Cuando se producen los cambios, no es el repositorio quien los guarda; en lugar de eso, se envía un mensaje a la sesión, lo cual significa que se pueden guardar múltiples cambios de una sola vez. Por encima de esto hay transacciones explícitas. El

código del Listado 2 consulta un nodo (por ejemplo, /fotos/2008) de un repositorio. Por debajo de éste, crea un nuevo nodo *Amsterdam*, asigna propiedades y guarda los cambios.

La API soporta una función de búsqueda a través de *SQL* o *XPath* para la realización de consultas complejas. Para mantener un registro de los cambios producidos en el repositorio se referencia el *EventListeners* registrado con la sesión. El usuario puede especificar qué partes del repositorio monitorizar durante el proceso de registro y restringir la notificación a cambios y tipos determinados. Con esta última funcionalidad es sencillo ejecutar un flujo de trabajo determinado con un tipo específico de contenido del repositorio. Los componentes de las distintas aplicaciones, o las aplicaciones mismas, pueden

reaccionar las unas con las otras mientras permanecen conectadas. Por ejemplo, puede haber muchas aplicaciones que guarden fotos en un mismo álbum.

El nuevo framework Apache Sling [4] está basado en un repositorio de contenidos basado a su vez en *REST*. Cada petición se empareja con un contenido del repositorio, para luego seleccionar un script que se encarga de mostrar dicho contenido.

Desarrollo: JSR 283

La versión 2 del estándar se encuentra actualmente en desarrollo bajo la denominación *JSR 283*. El nuevo estándar, que se planea esté listo este mismo año, incluirá varias extensiones. El principal objetivo es la mejora de la gestión del repositorio y de las funciones de administración. Por ejemplo, al estándar actual no le preocupan ni el control del acceso ni la gestión de los tipos de los nodos. *JSR 283* se ocupará de estos aspectos [5].

Futuro

El proyecto *JCR* ha convertido la palabra *JCR* en sinónimo de *repositorio de contenidos*. Un repositorio de contenidos se concibe como una infraestructura con unos servicios definidos. El usuario sólo tiene que preocuparse de la aplicación, mientras que una sola aplicación puede acceder fácilmente a varios repositorios de contenidos.

La flexibilidad de *JCR* ayuda sobremedida al programador, que puede aprenderse una sola API y aplicarla a una variedad de casos distintos. Además, con Jackrabbit no dependemos de ninguna marca comercial y reducimos el coste de la migración. ■

Tipos

Cada nodo tiene exactamente un tipo primario. Este tipo define la estructura del nodo, por ejemplo, especificando qué propiedades o hijos puede contener. Además del tipo primario, un nodo puede tener un número indeterminado de *mixins*. Un *mixin* es también una definición de tipo que puede añadir propiedades a un nodo cualquiera. Cada aplicación puede definir sus propios tipos. La combinación de la herencia múltiple y los *mixins* posibilita una definición de tipos extremadamente flexible y precisa. El estándar define varios tipos, como por ejemplo *nt:unstructured*, con el que se permiten arbitrariamente árboles de nodos y propiedades.

Normalmente conviene usar los tipos que ya hay y adaptarlos luego a nuestras propias necesidades. Para el álbum de fotos esto significaría usar algún tipo como *nt:folder*, que describe un directorio, con nuestro propio *mixin* aportando información adicional sobre el álbum. A su vez, cada imagen puede ser de tipo *nt:file*, con un *mixin* que contenga datos específicos de imagen. El uso de tipos estándar ofrece también la ventaja de simplificar la interacción con aplicaciones de terceros y el manejo que éstas hagan de los datos.

RECURSOS

- [1] JSR 170: <http://www.jcp.org/jsr/detail/170.jsp>
- [2] Apache Jackrabbit: <http://jackrabbit.apache.org>
- [3] Wiki de Apache Jackrabbit: <http://wiki.apache.org/jackrabbit/FrontPage>
- [4] Proyecto Apache Sling: <http://incubator.apache.org/sling>
- [5] JSR 283: <http://jcp.org/en/jsr/detail?id=283>
- [6] REST explicado: <http://www.xfront.com/REST-Web-Services.html>