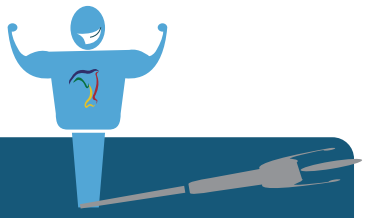
autentia

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...

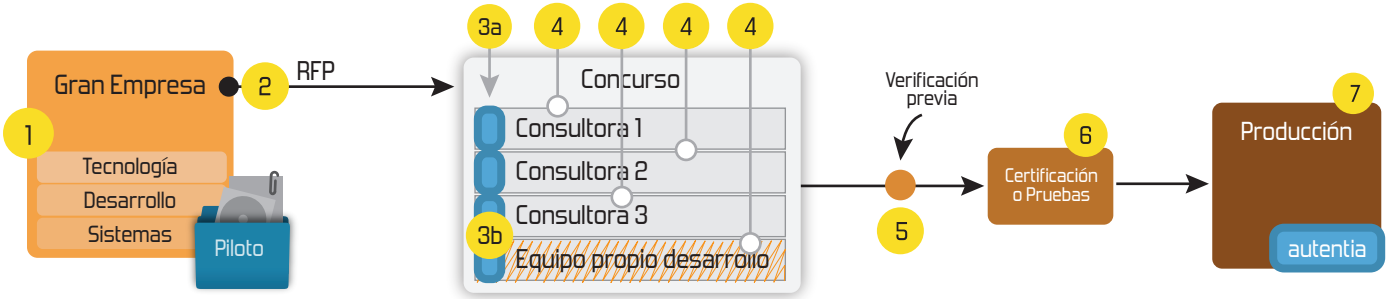
1. Desarrollo de componentes y proyectos a medida



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.

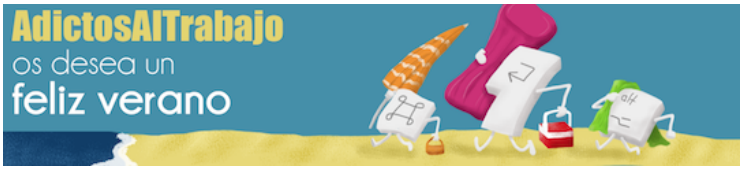


4. Cursos de formación (impartidos por desarrolladores en activo)

- 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



autentia
Soporte a desarrollo informático

Entra en Adictos a través de
E-mail
Contraseña
Entrar
Registrarme
Olvidé mi contraseña

Inicio Quiénes somos Formación Comparador de salarios Nuestros libros Más

» Estás en: Inicio Tutoriales Primeros pasos con Neo4j

Juan Alonso Ramos

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

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

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2014-07-07

Tutorial visitado 1 veces Descargar en PDF

Primeros pasos con Neo4j

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Instalación.
- 4. Lenguaje Cypher.
- 4.1 Crear el grafo.
- 4.2 Consultas sobre el grafo.
- 5. Conclusiones.

1. Introducción.

Neo4j es una base de datos orientada a grafos escrita en Java, es decir la información se almacena de forma relacionada formando un grafo dirigido entre los nodos y las relaciones entre ellos. Se integra perfectamente con múltiples lenguajes como Java, PHP, Ruby, .Net, Python, Node, Scala, etc. La base de datos está embebida en un servidor Jetty. Está especialmente indicada para modelar redes sociales y sistemas de recomendación.

Se distribuye en dos versiones: la community edition (open source) y la enterprise edition. Para hacer pruebas de concepto nos basta con la community edition pero si quieres sacarle todo el partido a Neo4j la opción enterprise es la más recomendable ya que permite ponerla en cluster, monitorización, backups en caliente y un sistema de cache de alto rendimiento, además de soporte de sus creadores.

Otra de las ventajas que tiene Neo4j es que se pueden efectuar las consultas directamente a través de un API Rest lo que hace especialmente interesante su integración con aplicaciones web.

En este tutorial vamos a ver unos primeros pasos con Neo4j, cómo instalarla en una máquina Ubuntu, crear un grafo y realizar consultas sobre el mismo.

2. Entorno.

El tutorial se ha realizado con el siguiente entorno:

- Ubuntu 12.04 64 bits
- Oracle Java SDK 1.7.0_51-b13
- Neo4j 2.1.1 - community edition

3. Instalación

Antes de instalar Neo4j comprueba que tienes como mínimo la versión 1.7 del SDK de Java. Para instalar Neo4j voy a optar por hacerlo utilizando el repositorio de paquetes de Debian. Los pasos son los siguientes:

Nos cambiamos a usuario root

```
1 sudo -s
```

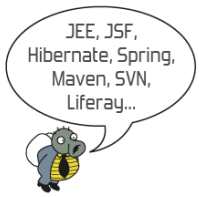
Importamos la clave pública de neo4j y aÃ±adimos el repositorio al sources.list

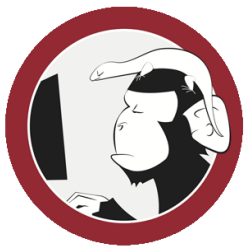
```
1 wget -O - http://debian.neo4j.org/neotechnology.gpg.key
2 apt-key add echo 'deb http://debian.neo4j.org/repo stable/' > /etc/apt/sources.list.d/neo4j
```

Actualizamos los repositorios e instalamos el paquete neo4j, versión community edition

```
1 apt-get update
```

Catálogo de servicios Autentia





Síguenos a través de:

Últimas Noticias

» Screencasts de programación narrados en Español

» Sorteo de entradas para APIdays Mediterranea

» Concurso del Día de la Madre:

» Aprende gratis ReactiveCocoa

» Checklist de Scrum de Autentia

Histórico de noticias

Últimos Tutoriales

» Introducción a WSO2 API Manager

» Introducción a Groovy y Grails con Maven: el patrón CRUD

» Menu.bat Una forma cómoda de ejecutar comandos y aplicaciones, por ejemplo para Maven

» Testing de Hadoop con MRUnit

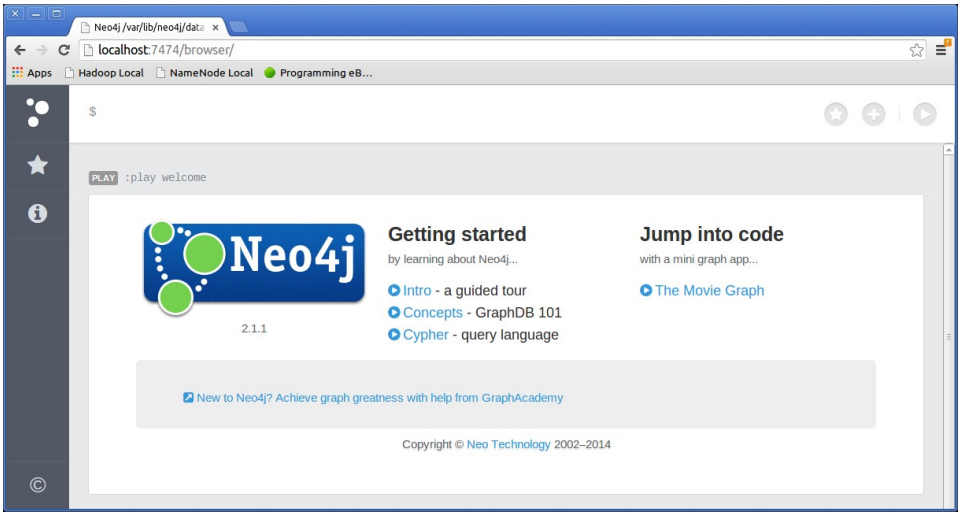
» Dart, el lenguaje de programación web del futuro creado por Google

```
apt-get install neo4j
```

Si todo ha ido bien nos saldrá el mensaje por consola:

```
Starting Neo4j Server...WARNING: not changing user
process [10310]... waiting for server to be ready..... OK.
http://localhost:7474/ is ready.
```

En la url http://localhost:7474 se levanta un cliente de Neo4j para trabajar y visualizar la base de datos de Neo4j. También tiene una ayuda muy útil para ir conociendo el interfaz del cliente web así como una pequeña guía que explica los conceptos de grafo, nodo, etiqueta, relación y unos primeros pasos con el lenguaje cypher.



4. Lenguaje Cypher.

Cypher es un lenguaje declarativo para trabajar con grafos. Permite crear la estructura de un grafo, nodos y relaciones así como realizar consultar filtrando con argumentos, ordenar, etc. Para aprender a utilizar este lenguaje voy a utilizar el ejemplo del grafo de películas y actores que viene con la instalación de Neo4j llamada 'The Movie Graph'.

4.1 Crear el grafo.

Con **CREATE** estamos creando un nodo del grafo (se sabe porque va entre paréntesis) con etiqueta 'Movie' que se guarda en la variable 'TheMatrix' (utilizada para posteriormente referirnos al nodo) con las propiedades 'title', 'released' y 'tagline' con sus respectivos valores en formato JSON.

Un trozo del grafo para la película Matrix con sus actores principales, directores y productor.

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (AndyW:Person {name:'Andy Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
  (AndyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),
  (JoelS)-[:PRODUCED]->(TheMatrix)

...
```

A continuación se crean unos cuantos nodos más etiquetados con 'Person' y las propiedades 'name' y 'born' de forma similar al anterior. Estos nodos de momento no forman un grafo, al menos un grafo que tenga cierto sentido para nosotros, por ello es necesario crear las relaciones.

Las relaciones se crean de forma muy similar a los nodos, mediante **CREATE** indicamos a través de las variables que identifican los nodos las relaciones con otros nodos. La relación puede tener nombre y propiedades. Si intentamos interpretar la primera, el lenguaje declarativo de Cypher nos indica que una persona identificada como 'Keanu' actuó con el rol de 'Neo' en 'TheMatrix'.

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
...
```

Como ves el lenguaje es bastante sencillo de entender. Para el resto de actores las relaciones son similares, cambiando su rol en la película pero el tipo de relación es el mismo 'ACTED_IN'. Esto es importante para luego las consultas dentro del grafo. Otras relaciones del grafo son para los dos directores y el productor.

4.2 Consultas sobre el grafo.

Una vez creado el grafo vamos a realizar algunas consultas sobre él.

Si queremos buscar algo en el grafo debemos buscar con **MATCH**. Para consultar todo el grafo.

```
MATCH (n) RETURN n
```

Desde el cliente de Neo4j podemos abrir los distintos nodos y ver sus relaciones. También podemos ver las propiedades

Últimos Tutoriales del Autor

» Testing de Hadoop con MRUnit

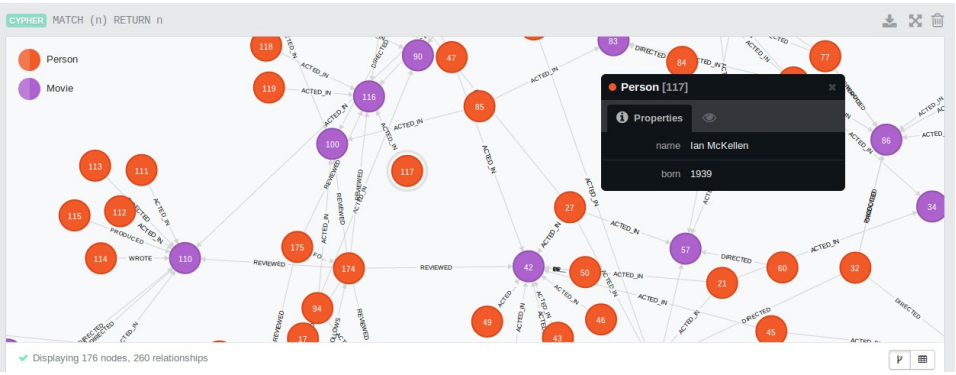
» Introducción a Spring Data Hadoop

» Implementar una función UDF de Apache Pig

» Primeros pasos con Apache Pig

» Implementando tu propio Writable en Hadoop

de cada nodo con sus atributos.



Si queremos buscar un nodo en concreto dentro del grafo. Si existe un nodo con el atributo 'name' igual a 'Keanu Reeves' nos lo devolverá.

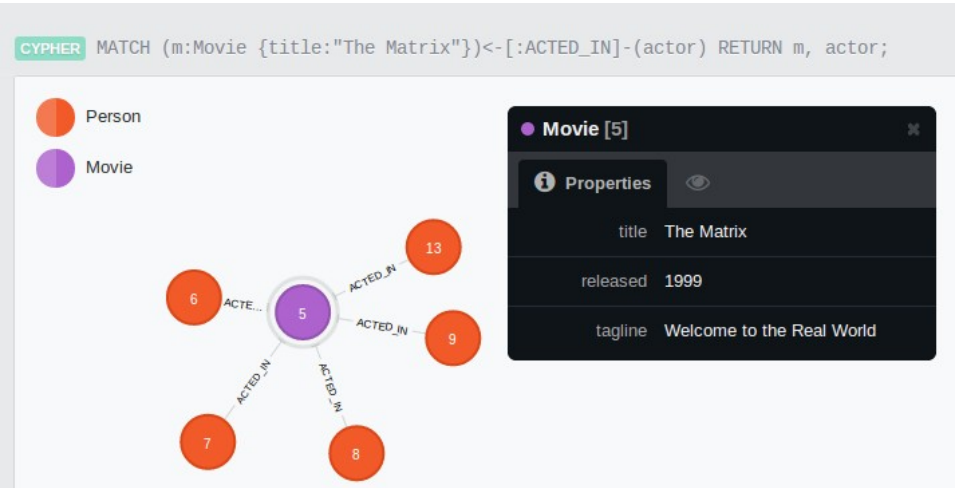
```
MATCH (actor {name: "Keanu Reeves"}) RETURN actor
```

Más búsquedas, por ejemplo consultar todas las películas en las que ha actuado un actor:

```
MATCH (n {name: "Keanu Reeves"})-[:ACTS_IN]->(d) RETURN n, COUNT(d)
```

Todos los actores de una película:

```
MATCH (m:Movie {title:"The Matrix"})<-[:ACTED_IN]-(actor) RETURN m, actor;
```



Listado de las películas filmadas en 2008.

```
MATCH (nineties:Movie) WHERE nineties.released = 2008 RETURN nineties.title
```

Listado de todas las películas y sus actores

```
MATCH (m:Movie)<-[:ACTED_IN]-(actor)-[:ACTED_IN]->(movie)
RETURN movie.title, collect(actor.name),
count(*) as count
ORDER BY count desc;
```

Como se puede ver la creación de un grafo en Neo4j es muy sencilla así como realizar las consutas que necesitemos para extraer la información que necesitemos.

5. Conclusiones.

En un primer vistazo hemos podido ver cómo funciona Neo4j, la creación de un grafo y sus consultas a través del lenguaje Cypher. Este lenguaje está basado en SQL por lo que cuando empiezas con él su sintaxis te resulta muy familiar. Las consultas ya dependen de las necesidades de tu negocio.

Os dejo alguna referencia para profundizar en Neo4j:

Guía de referencia: http://docs.neo4j.org/chunked/milestone/preface.html

Chuleta de neo4j: http://docs.neo4j.org/refcard/2.1/

Recursos de aprendizaje http://www.neo4j.org/learn

Charla de Michael Huger - Neo4j for Java Developers: https://www.youtube.com/watch?v=FCp68iGo0pY

Espero que te haya sido de ayuda.

Un saludo.

Juan

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 8+1 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

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

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

IMPULSA Impulsores Comunidad ¿Ayuda?

---- sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by karmacracy

Copyright 2003-2014 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

W3C XHTML 1.0 W3C CSS XML RSS XML ATOM