ESCUELA:
TECNOLOGÍA

LUCENE, COMPASS, SEARCH

30 al 31 de Octubre de 2018

indra

icono
TRAINING CONSULTING

ÍNDICE

- Arquitectura de los sistemas de búsqueda e indexación
- Lucene, el primer paso
- Creación de documentos e índices
- Extracción de información mediante consultas

ÍNDICE

- Compass, más allá de Lucene
- Independencia de la fuente de datos
- Configuración
- Integración con Spring
- Extracción de información mediante consultas

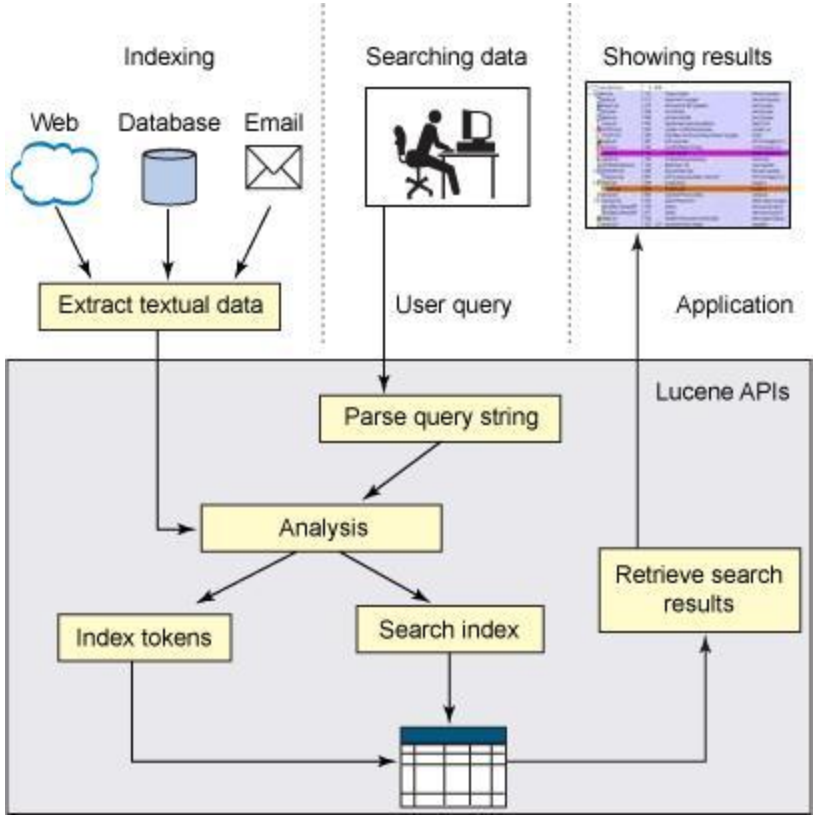
ÍNDICE

- Hibernate Search, haciendo lo que HQL no puede hacer
- Artefactos clave
- Configuración
- Creación de clases persistentes “indexables” mediante anotaciones
- Extracción de información mediante consultas

LUCENE, COMPASS, HIBERNATE SEARCH

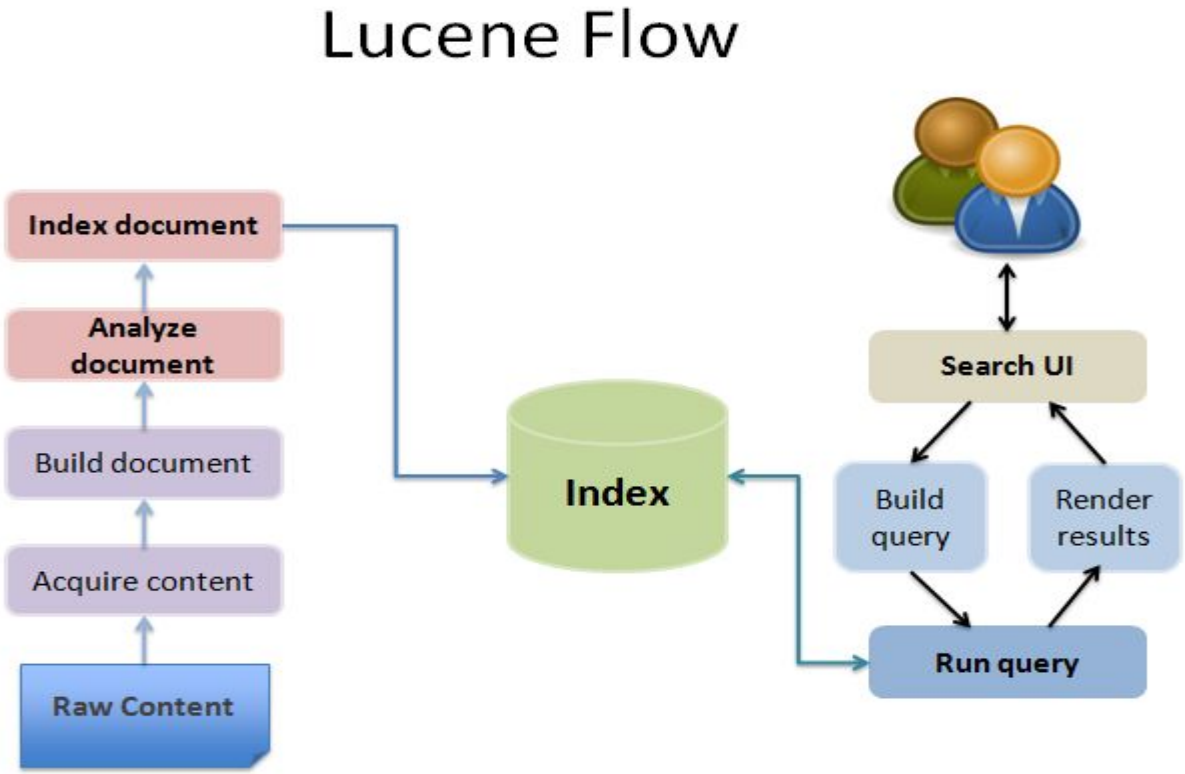
- NOMBRE APELLIDO PROFESOR
  José Mª Díaz Charcán
- VER PERFIL COMPLETO:
  linkedin.com/company/icono-training-consulting
  training@iconotc.com
- CONTACTO

Arquitectura de los sistemas de búsqueda e indexación (Lucene)



Origen: http://www.knstek.com/full-text-search-using-apache-lucene-part/

Arquitectura de los sistemas de búsqueda e indexación (Lucene)



Origen: http://www.knstek.com/full-text-search-using-apache-lucene-part-iii/

APACHE LUCENE (ver carpeta “docs” de lucene-7.5.0.zip)

- Lucene es una herramienta pensada para facilitar el acceso flexible, cercano al lenguaje natural, a la información existente en documentos con formatos muy dispares
- Sus características principales son:
  - Escalable
  - Algoritmos sofisticados de búsqueda
  - Lenguaje de consultas muy potente
  - Integración con Solr
  - Integración con Elasticsearch
  - Proyecto Apache Tika

TAREAS ESTÁNDAR

- Creación de índices
  - El analizador: StandardAnalyzer, SpanishAnalyzer,...
  - Dónde guardamos el índice: RAMDirectory, FSDirectory, ...
  - Configurar el índice (IndexWriterConfig) y crear un IndexWriter
- Añadir documentos
  - Crear documento (Document)
  - Añadir campos (TextField, StringField, …)
  - Incluir el documento en el índice (IndexWriter)

TAREAS ESTÁNDAR

- Crear consultas en el lenguaje de Lucene
  - QueryParser clásico u otros
  - Acceder a donde se encuentra el índice: DirectoryReader
- Realizar la búsqueda
  - Buscar mediante el índice: IndexSearcher.read -> TopDocs
  - Obtener los documentos con puntuación más alta (scoreDocs)
- Mostrar resultados
  - Obtener lo que necesitamos doc.get y mostrarlo

COMPASS

- Construido sobre Lucene
- Simplifica la programación, adoptando un enfoque tipo ORM, con un lenguaje de consultas
- Es posible trabajar con datos en diferentes formatos, y mapearlos para que Compass sepa cómo tratarlos: anotaciones (OSEM), XML (XSEM), JSON (JSEM), recursos genéricos (RSEM)
- Las operaciones que realizamos pueden ser transaccionales
- Compass se integra con ORMs: JPA, Hibernate y otros. Así, los índices se actualizan de forma automática. También con Spring Framework
- Distribuir índices de Lucene guardados en base de datos o con productos Data Grid

TAREAS ESTÁNDAR

- Escribir una configuración (compass-cfg.xml)
- Programáticamente crear un objeto de tipo Compass y, a menudo, un CompassTemplate
- Decidir si mantenemos el índice existente o lo recreamos
- Indexar objetos con una CompassSession y una CompassTransaction
- Realizar consultas que pueden o no devolver resultados, mediante (normalmente por comodidad) un CompassTemplate
- El lenguaje que empleamos en las consultas es el de Lucene
- Ejecutar operaciones de búsqueda, actualización y borrado sobre objetos específicos

HIBERNATE SEARCH

- Integración con Lucene y Elastic Search
- El principal propósito de la herramienta es ayudarnos consultar una base de datos usando un lenguaje (el de Lucene) mucho más potente, más cercano al lenguaje natural que SQL o HQL
- Búsqueda sofisticada orientada a textos, normal o difusa, agrupamiento de los resultados por varios criterios, diversas herramientas para crear consultas (texto plano, QueryBuilder)
- Al cambiar el estado de las entidades, cambia automáticamente el del índice
- Los resultados se obtienen clasificados por relevancia

HIBERNATE SEARCH

- Los índices pueden distribuirse empleando Data Grids como Inifinispan y/o ElasticSearch
- También es posible indexar entidades geolocalizadas mediante anotaciones
- La estrategia de Search es apoyarse en Lucene por defecto, y en Hibernate para tratar las entidades cuyos datos acabarán en las tablas.
- Para nosotros, resulta muy cómodo que una operación persistente sobre una entidad se proyecte automáticamente sobre el índice

TAREAS ESTÁNDAR

- Escribir una configuración (hibernate.cfg.xml)
- En ella, definir en qué medio va a crearse el índice y la base del mismo
- Anotar las clases persistentes (@Entity) con las anotaciones de Search
- Las esenciales son: @Indexed, @Field, @IndexedEmbedded si hay relaciones entre clases, y los “puentes” (@DateBridge)
- Debe respetarse el contrato estándar de Hibernate. Fechas, métodos equals y hashCode, etc
- Es necesario crear, o recrear el índice
- Hemos de escoger el método para definir las consultas