Diseño e Implementación de un sistema de URL-Shortening compuesto de un servidor de REST-WS sobre CloudFoundry y una aplicación cliente desarrollada en GWT

Informe de decisiones técnicas y alternativas consideradas

Autor: Carlos Yagüe Martín-Lunas

1. Metodología usada durante el desarrollo del JAR **url-shortener-core**

    - Se ha usado TDD (Test Driven Development) para probar las clases desarrolladas.
        i. Hay un test para probar el `UrlShortenerWsDAO`
        ii. Y otro para el `UrlShortableManager` cuyo DAO accede a la BBDD MongoDB.
    - Ambos tests contienen la anotación `org.junit.Ignore` para deshabilitar ambos tests y no fallen durante la compilación Maven. En en el caso de que el servidor de WS de CloudFoundry o el SGBD
    - Ambos tests son configurables desde el archivo `/src/test/resources/test.properties`

2. Metodología usada durante el desarrollo del WAR **url-shortener-ws**

    - Durante el desarrollo se ha utilizado un servidor Apache Tomcat 6 para probar su uso junto con el plugin HTTP4e de Eclipse para probar las llamadas a los servicios REST.

3. Metodología usada durante el desarrollo del WAR **url-shortener-web**

    - Se ha ido generando dicho módulo a partir del arquetipo Maven del plugin GWT Maven Plugin en comparativa con los proyectos que crea el Google Suite Plugin de Eclipse. De esta forma se ha conseguido generar una aplicación con soporte Maven y plenamente funcional con los plugins de GWT de Eclipse.

4. Despliegue de aplicaciones en CloudFoundry.com

    - Se ha partido de la aplicación de ejemplo *hello-spring-mongodb*: https://github.com/SpringSource/cloudfoundry-samples/wiki/Spring-Hello-MongoDB-Sample-Application
    -

5. Despliegue de aplicaciones en Google App Engine

    - Para los despliegues en dicha plataforma se ha usado el plugin de Eclipse Google Suite Plugin mediante la opción Deploy App Engine Project.

6. ¿Por qué no se ha utilizado Hibernate?

Se pensó en utilizar en la aplicación `url-shortener-ws`, pero al no estar implementado un dialecto Hibernate para tratar con el SGBD MongoDB se ha desestimado su uso. Se ha optado por usar el módulo de Spring SPRING-DATA-MONGODB como wrapper de dialecto con el driver java del SGBD MongoDB.

7. ¿Por qué se ha usado REST en vez de SOAP?

    - La sencillez y la rapidez de la arquitectura REST se adaptaba perfectamente al caso de uso de unos servicios de URL-Shortening ya que no requiere el fuerte tipado que ofrece SOAP (sin contrato).
    - Otros sistemas existentes y fiables usan esta misma tecnología. Este es el caso del servicio de URL Shortening de Google: http://code.google.com/intl/es-ES/apis/urlshortener/v1/getting_started.html

8. ¿Por qué se ha usado JERSEY como implementación de REST?

    - REST es una arquitectura que define la forma de hacer WS, pero no la implementación. Consultadas las implementaciones existentes, se barajaron las siguientes:
        i. JBoss RESTEasy: http://www.jboss.org/resteasy
        ii. JERSEY: http://jersey.java.net/

    - Ambas implementaciones tienen soporte Spring y eran fiables *a priori*.
    - La primera que se probó fue Jersey al ser una de las implementaciones del proyecto Glassfish, auspiciado por Oracle (antigua Sun) y al funcionar correctamente, se dio por válida.