Integración simultánea de tecnologías ebean y jpa en un proyecto playframework java

Simultaneous integration of ebean and jpa technologies on a playframework java project

Armando Méndez Morales, Gloria del Carmen Córdoba Hernández,
Andrés Donaciano Martínez Guillén
División de Tecnologías de la Información y Comunicación, Universidad Tecnológica de la Selva (UTSelva). Entronque Toniná Km. 0.5 Carretera Ocosingo-Altamirano, Ocosingo, Chiapas, México, C.P. 29950.
{amendezm2009, gloriacrdb}@gmail.com, andres_mtz_g@hotmail.com

PALABRAS CLAVE:

Playframework, ebean, jpa, orm.

RESUMEN

En este trabajo se presenta una forma de implementar las tecnologías orm ebean y jpa en el desarrollo de una aplicación web empleando playframework, la integración de ambas tecnologías permitirá aprovechar sus mejores características realizando operaciones con la base de datos, por lo que ebean se utilizará para operaciones create, update y delete con registros de una sola tabla, mientras que JPA se utilizará para operaciones con varias tablas mediante vistas o procedimientos almacenados que retornan registros.

KEYWORDS:

Playframework, ebean, jpa, orm.

ABSTRACT

This work presents a way of implementing ORM ebean and jpa technologies in the development of a web application using playframework, the integration of both technologies will let improve their best characteristics making operations over database, therefore ebean will be used to create, update and delete operations with records of one unique table, whereas that JPA will be used for operations with several tables through views or store procedures that return records.

Recibido: 4 de Julio del 2017 • Aceptado: 9 de febrero del 2018 • Publicado en línea: 28 de agosto del 2018z

1. INTRODUCCIÓN

El mapeo objeto-relacional es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos relacional. [1]

Playframework es un framework de aplicaciones web de java y scala de alta productividad que integra los componentes y APIs que necesita para el desarrollo de aplicaciones web. Play está basado en una arquitectura ligera, sin estados y web-amigable y con características predecibles y mínimos consumo de recursos (CPU, memoria, hilos) para aplicaciones altamente escalables gracias a su modelo reactivo basado en Akka Streams. [2].

En el desarrollo de aplicaciones web utilizando la tecnología de playframework, para la capa de acceso a datos se recomienda el uso de la tecnología de tipo ORM que puede ser slick, anorm, eban, jpa; sin embargo, para algunos contextos puede ser necesaria la combinación de algunas tecnologías, para aprovechar sus mejores características. En este artículo se presentan configuraciones y código para lograr la integración entre los orms ebean y jpa, debido a que en la literatura existen muy pocos recursos que especifican la forma en que esto debe realizarse, se toma como referencia el proyecto que utiliza el orm ebean, disponible en https://github.com/playframework/play-ebean-example.

2. PLANTEAMIENTO DEL PROBLEMA.

En el sitio de playframework [3] se muestran las configuraciones que deben aplicarse para poder emplear cada tecnología de tipo orm. Petrella [4] muestra un ejemplo de desarrollo de aplicaciones utilizando la tecnología ebean; Nicolas y Sietse [5] también explican detalladamente un ejemplo con ebean, sin embargo, respecto a la tecnología jpa solamente muestran la configuración básica para emplearla. Por otra parte, en github.com/playframework [5] se encuentra disponible para descarga un proyecto de ejemplo donde utiliza específicamente jpa. Sin embargo, en la literatura existen muy pocos proyectos que presentan un ejemplo detallado sobre la forma en la que se pueden integrar ambas tecnologías.

3. DESARROLLO

La tecnología Ebean permite que los objetos, de las clases modelos (de modelo-vista-controlador) que extienden a la clase ebean.Model, puedan adoptar el estado y comportamiento soportados por Ebean, el comportamiento traducido a funcionalidades consisten en la realización de consultas específicas sobre los mismos modelos, por ejemplo al realizar una consulta a través de un campo id, o bien, para que se obtengan todos los registros encontrados en el propio modelo, como se aprecia en la tabla 1.

Tabla 1. Extensión de métodos de la clase ebean. Model para clases correspondientes a modelos

```
package models;
import java.util.*;
import javax.persistence.*;
import com.avaje.ebean.*;
import play.data.format.*;
import play.data.validation.*;
@Entity
public class Computer extends Model{
@Id
public Long id;
      public String name;
public Computer(Long id) {
   this.id = id;
}
      public static Find<Long,Computer> find = new Find-
<Long,Computer>(){};

     public static PagedList<Computer> page(int page, int
pageSize, String sortBy, String order, String filter){
return find.where()
  .ilike("name", "%" + filter + "%")
  .orderBy(sortBy + " " + order)
  .fetch("company")
  .findPagedList(page, pageSize);
   }
}
```

La tecnología Ebean es muy práctica cuando se desean ejecutar operaciones en la base de datos de tipo insert, update y delete, para registros individuales tal y como se aprecia en la tabla 2.

Tabla 2. Operaciones con objetos para buscar, borrar, agregar o actualizar registros en la base de datos

```
package controllers;
import models.*;
import play.mvc.*;
import javax.inject.Inject;
import javax.persistence.PersistenceException;
import play.db.jpa.JPAApi;

public class HomeController extends Controller {
    public static Result actionsEbean(){
// Búsquedas específicas por campos
      Computer obj = Computer.find.byId (7L);
// Borrado de objetos (registros de la base de datos)
obj.delete();
// Creación de objetos (registros en la base de datos)
obj = new Computer(7L);
obj.save();
// Actualización de datos hacia la base de datos
obj.name = "PC from Universidad Tecnológica de la Selva";
obj.update();
return ok(views.html.mipagscala.render(obj));
    }
}
```

No obstante, el mecanismo que utiliza Ebean, se vuelve impráctico cuando se necesita que la información consultada, corresponde a un conjunto de campos correspondientes a diferentes tablas o vistas de bases de datos, inclusive resultantes de ejecutar procedimientos almacenados que en su interior ejecutan operaciones contenidas en una o varias transacciones donde puedan ser incluidas desde una a múltiples tablas, como se requiere en la consulta encontrada en la tabla 3.

Tabla 3. Procedimiento almacenado que retorna registros que serán mapeados en modelos a ser empleados en la capa de presentación

```
drop procedure if exists computervip_sp;
delimiter $$
create procedure computervip_sp ()
Begin
   /* Uso:
          call computervip_sp();
   */
   # asegurarse de que al ejecutarse este sp, se obligue a
   # insertar el valor de company en 1 para todos aquellos
   # registros que no tengan la compañia establecida
    update computer set company_id = 1 where company_id
is null;
   # Finalmente retornar la información actualizada con la
   # aplicación de las reglas del negocio
   Select c.id, c.name, n.id company, n.name companyname,
c.introduced, c.discontinued
   , case when c.company_id  is null then 'Invalid' else 'OK'
end as validated
          from computer c
           inner join (
             Select * from company where name in
          ('Apple Inc.', 'IBM', 'Nokia', 'Sony'
          , 'Xerox', 'ASUS', 'Samsung Electronics')
           ) n on c.company_id = n.id;
End
$$
```

Tomando todo esto en cuenta, se creó el método ConnBD.getRegistros el cual retorna una lista de objetos correspondientes a los registros obtenidos de ejecutar una instrucción sql indicada. Este método necesita dos parámetros, uno correspondiente a una instrucción sql y el otro corresponde a la clase de código con la que se mapearán los resultados obtenidos por esta instrucción sql. El método se puede observar en la tabla 4.

Tabla 4. Encapsulado de la funcionalidad de JPA para retornar objetos

```
package modelsjpa;
import com.avaje.ebean.*;
import play.Logger;
import java.lang.reflect.Field;
import java.util.*;
import play.db.jpa.JPAApi;

public class ConnBD extends Model{
    public static List getRegistros(JPAApi jpaApi, String sql-
Given, Class classNameToReturn){
        List lista = new ArrayList();
        lista = jpaApi.withTransaction(entityManager -> {
            return entityManager.createNativeQuery(sqlGiven,-
classNameToReturn).getResultList();
        });
       return lista;
    }
}
```

Tabla 5. Estructura de campos a ser utilizados en la capa de presentación

```
package modelsjpa;
import play.data.format.Formats;
import javax.persistence.*;
import java.util.Date;
@Entity
public class ComputerJPA  {
    private static final long serialVersionUID = 1L;
    @Id
    public Long id;
    public String name;
@Formats.DateTime(pattern="yyyy-MM-dd")
    public Date introduced;
    @Formats.DateTime(pattern="yyyy-MM-dd")
    public Date discontinued;
    public String company;
    public String companyname;
    public String validated;
}
```

De esta forma queda disponible la funcionalidad genérica para pasar cualquier instrucción sql válida, que realice una llamada directamente a una instrucción sql que tenga como objetivo ejecutar un conjunto de instrucciones y que retorne un conjunto de registros, los cuales serán mapeados a objetos acorde al parámetro classNameToReturn. En la tabla 5 se observa la estructura de la clase de objetos a los que se adaptará el resultado de ejecutar el procedimiento almacenado mostrado en la tabla 3.

Finalmente, en la tabla 6 se muestra el método listjpa encontrado en HomeController, y que emplea la tecnología jpa a través de la llamada del método getRegistros (expuesta en la tabla 4).

Tabla 6. Empleo de la funcionalidad encapsulada de JPA para consultas sql que retornan registros de múltiples objetos de base de datos ya sean procedimientos almacenados, vistas o tablas.

```
@Inject
private JPAApi jpaApi;
public Result listjpa(int page, String sortBy, String order,
String filter) {
    List lista = new ArrayList<>();
    String sqlString = "call computervip_sp();";
    // Con JPA, se ejecutan instrucciones complejas encapsu-
ladas en store procedures
    // y se obtienen los correspondientes registros mapeados a
la estructura del modelo indicado
    lista = modelsjpa.ConnBD.getRegistros(jpaApi, sqlString,
modelsjpa.ComputerJPA.class);
     return ok(views.html.listjpa.render(lista,sortBy, order, fil-
ter));
}
```

4. RESULTADOS Y CONCLUSIONES

Con el desarrollo de este trabajo se aprovechan las ventajas de ambas tecnologías ebean y jpa; Con ebean se evitan las consultas JDBC puras y que los registros obtenidos tengan que ser mapeados a objetos de forma explícita por el desarrollador, permitiendo ejecutar operaciones create, update y delete en registros unitarios mediante objetos, mientras que al utilizar jpa es posible ejecutar operaciones de tipo sql haciendo la recomendación que se realicen llamadas a procedimientos almacenados, y que sean estos lo que contengan las consultas y operaciones complejas con la base de datos que puedan incluir múltiples transacciones y operaciones sql.

Por último, este trabajo se ha podido aplicar en proyectos realizados del cuerpo académico en tecnologías de la información, el más sobresaliente con apoyo de financiamiento para el proyecto "SEVADE Sistema de Entorno Virtual de Aprendizaje Dirigido a la Evaluación" de la convocatoria PRODEP [7] "Fortalecimiento de Cuerpos Académicos" del año 2015.

5. ANEXOS

Tabla 7. Configuraciones parciales relacionadas a ebean y jpa en los archivos del proyecto playframework

Archivo	*Configuraciones principales*
build.sbt	*name := "play-java-ebean-example"* *version := "0.0.1-SNAPSHOT"* *scalaVersion := "2.11.11"* *lazy val root = (project in file(".")).enablePlugins(PlayJava, PlayEbean)* *libraryDependencies += jdbc* *libraryDependencies += javaJpa* *libraryDependencies += "com.adrianhurt" %% "play-bootstrap" % "1.0-P25-B3"* *libraryDependencies += "mysql" % "mysql-connector-java" % "5.1.21"* *libraryDependencies += "org.eclipse.persistence" % "eclipselink" % "2.6.0"*
project/plugins.sbt	*resolvers += "Typesafe repository" at "http://repo.typesafe.com/typesafe/releases/"* *addSbtPlugin("com.typesafe.play" % "sbt-plugin" % "2.5.15")* *addSbtPlugin("com.typesafe.sbt" % "sbt-play-ebean" % "3.0.1")*
project/build.properties	*sbt.version=0.13.15*
conf/application.conf	*application.name=computer-database* *akka.log-level = "WARN"* *db.default.driver=com.mysql.jdbc.Driver* *db.default.url="mysql://vip:vip123@localhost:3306/edufacil"* *db.default.jndiName=DefaultDS* *jpa.default=defaultPersistenceUnit* *evolutions {* *enabled = false* *}* *ebean.default="models.** *# Assets configuration* *#~~~~~* *"assets.cache./public/stylesheets/bootstrap.min.css"="max-age=3600"*

REFERENCIAS

1. Naranjo, F. P. (2016). Herramienta de mapeo objeto relacional y la productividad de la empresa interfases software group. Journal of Science and Research: Revista Ciencia e Investigación, 34.
2. Lightbend. (26 de 09 de 2017). playframework. Obtenido de playframework: https://www.playframework.com/
3. playframework. (s.f.). https://www.playframework.com. Recuperado el 20 de 06 de 2017, de https://www.playframework.com/documentation/2.5.x/JavaEbean
4. Petrella, A. (2013). Learning PlayFramework 2. Birmingham: Packt Publishing.
5. Leroux, N., & de Kaper, S. (2014). Play for Java Covers play 2. Manning Publications.
6. github.com/playframework. (s.f.). github.com. Recuperado el 21 de 06 de 2017, de https://github.com/playframework/play-java-jpa-example
7. gob.mx. (s.f.). Recuperado el 01 de 11 de 2017, de Programa para el Desarrollo Profesional Docente, para el Tipo Superior (PRODEP): http://www.dgesu.ses.sep.gob.mx/Documentos/ultimosbenef/FORTA2015-1.pdf

Acerca de los autores

Armando Méndez Morales nació en el Distrito Federal, hoy Ciudad de México. Se graduó como Licenciado en Informática en el Instituto Tecnológico de Comitán y como Maestro en Ciencias en Ciencias de la Computación con especialidad de Ingeniería de Software en el Centro Nacional de Investigación y Desarrollo Tecnológico. Trabajó como desarrollador web independiente y por contrato en Nyssen Consultores, como Sysadmin y Dba en el Instituto de Investigaciones Eléctricas y actualmente se desempeña como Profesor de Tiempo Completo en la UTselva, donde es Representante del Cuerpo Académico de Tecnologías de la Información ante PRODEP, entre sus campos de interés la programación web y móvil, iniciando recientemente en el concepto de Internet de las Cosas programando con placas de arduino y raspberry.

Gloria del Carmen Córdoba Hernández nació en Ocosingo Chiapas, tiene un título de Licenciatura en Informática con Maestría en Ciencias de la Computación con especialidad en Bases de Datos. Se desempeña como Profesora de Tiempo Completo en la División de Tecnologías de la Información y Comunicación en la Universidad Tecnológica de la Selva desde el año 2002, en la Ciudad de Ocosingo, Chiapas, México; ha participado en el diseño de los planes y programas educativos de las ingenierías en competencias profesionales para el Subsistema de Universidades Tecnológicas, ha obtenido el reconocimiento a Perfil deseable en dos ocasiones por el Programa para el Desarrollo Profesional Docente para el tipo superior, es miembro del Cuerpo Académico en Tecnologías de la Información, reconocido ante el PRODEP con Número de registro (UTSEL-6).

Andrés Donaciano Martínez Guillén nació en Tuxtla Gutiérrez Chiapas, se graduó como Licenciado en Informática en la Universidad Autónoma de Chiapas y tiene una Maestría en Desarrollo de Software. Se desempeña como Profesor de Tiempo Completo en la División de Tecnologías de la Información y Comunicación en la Universidad Tecnológica de la Selva desde el año 2001, en la Ciudad de Ocosingo, Chiapas, México; actualmente es docente en el Instituto de Estudios Superiores de Chiapas en la modalidad semi-escolarizado en la capital del estado de Chiapas, colaboró de en la UTSelva como Representante Institucional ante el Programa para el Desarrollo Profesional Docente para el tipo superior, es miembro activo del Cuerpo Académico en Tecnologías de la Información, reconocido ante el PRODEP con Número de registro (UTSEL-6).