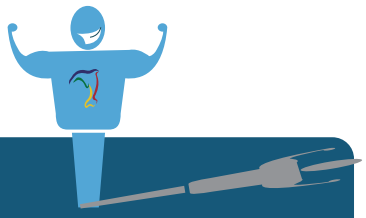
autentia

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...

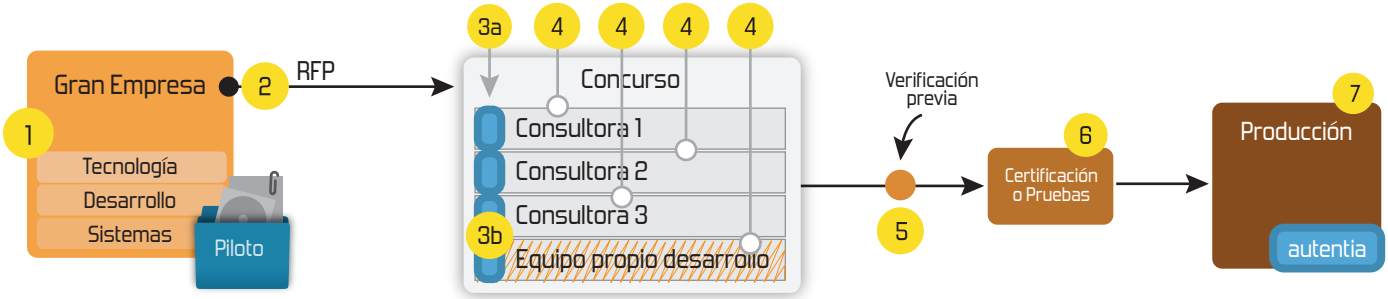
1. Desarrollo de componentes y proyectos a medida



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

DESARROLLADO POR:
Carlos García Pérez

Técnico especialista en informática de empresa (CEU).

Ingeniero Técnico en Informática de Sistemas (UPM)

Creador de MobileTest, Haaala!, Girillo, toi18n.

Charla sobre desarrollo de aplicaciones en Android.

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

Fecha de publicación del tutorial: 2011-08-15

96

Share | Regístrate para votar

MyBatis Generator (MGB): Generador de código para MyBatis e iBATIS

Introducción

En cualquier proyecto suelen haber tareas que pueden generarse de manera automática ahorrándose grandes esfuerzos y evitando los errores que podrían ser introduccidos si se hubieran realizado manualmente.

Claro está que el resultado de la generación automática tiene ser calidad y que los retoques y adaptaciones necesarias para incluirlo en nuestros proyectos deben ser mímimas.

En este tutorial vamos a hacer una introducción de **MyBatis Generator (MGB)** un generador de código para el **framework de presistencia MyBatis** desarrollado por Apache (y con licencia Apache License).

MGB se apoya en **JDBC**, para realizar una reflexión o introspección de una fuente de datos para generar automáticamente las clases del modelo, de acceso a datos y los archivos de mapeo (SQL Map XML).

MGB es una aplicación (es un JAR sin dependencias) bastante configurable a través de un archivo XML y puede ser invocado desde Ant, Maven o desde línea de comandos.

El producto y la documentación (bastante buena) puede descargarse desde página de descarga.

Un ejemplo

Para ver como funciona, partimos de una base de datos MySql ubicada en el esquema de nombre "cgarcia1".

view plain print ?

```
01.  CREATE TABLE companies (
02.    id INT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
03.    companykey VARCHAR(25) NOT NULL,
04.    name VARCHAR(100) NOT NULL,
05.    url VARCHAR(255),
06.    publicity TINYINT(1) UNSIGNED NOT NULL DEFAULT 0,
07.    userlicenses INT(10) UNSIGNED NOT NULL DEFAULT 20,
08.    teacherlicenses INT(10) UNSIGNED NOT NULL DEFAULT 1,
09.    email VARCHAR(50) NOT NULL,
10.    state TINYINT(1) UNSIGNED NOT NULL DEFAULT 1,
11.    messages TINYINT(1) UNSIGNED NOT NULL DEFAULT 1,
12.    notes    TINYINT(1) UNSIGNED NOT NULL DEFAULT 1,
13.    autoRegisterPwd VARCHAR(15),
14.    language VARCHAR(2) NOT NULL DEFAULT 'es',
15.    hddMbSpace INT(5) NOT NULL DEFAULT 5,
16.    PRIMARY KEY  (id),
17.    UNIQUE companyKeyIDX (companykey)
18.  ) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;
19.
20.  CREATE TABLE teachers (
21.    id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
22.    companyid INT(3) UNSIGNED NOT NULL,
23.    teacherkey VARCHAR(25) NOT NULL,
24.    pwd   VARCHAR(50) NOT NULL,
25.    name  VARCHAR(70) NOT NULL,
26.    email VARCHAR(50) NOT NULL,
27.    notes VARCHAR(255),
28.    testlicenses INT(3) UNSIGNED DEFAULT 100,
29.    PRIMARY KEY  (id),
30.    UNIQUE teacherIDX (companyid, teacherkey),
```

```
  KEY companyIDX (companyid),
  FOREIGN KEY  (companyid) REFERENCES companies(id) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;


CREATE TABLE teacherNotesCategories (
  id        INT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  teacherid INT(10) UNSIGNED NOT NULL,
  description   VARCHAR(128) DEFAULT NULL,

  PRIMARY KEY  (id),
  KEY teacherNotesCategoriesIDX (teacherid),
  FOREIGN KEY (teacherid) REFERENCES teachers(id) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;


CREATE TABLE  teacherNotes (
  id           INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  teacherid  INT(10) UNSIGNED DEFAULT NULL,
  categoryid INT(3)  UNSIGNED DEFAULT NULL,

  published  TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
  subject    VARCHAR(1000) DEFAULT NULL,
  content    MEDIUMTEXT DEFAULT NULL,
  PRIMARY KEY (id),
  KEY         teacherNotesIDX    (teacherid),
  KEY         teacherCategoryIDX (categoryid),
  FOREIGN KEY (teacherid)  REFERENCES teachers(id)  ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (categoryid) REFERENCES teacherNotesCategories(id)    ON DELETE SET NULL  ON U
) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;


CREATE TABLE users (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  companyid INT(3) UNSIGNED NOT NULL,
  userkey VARCHAR(25) NOT NULL,
  pwd       VARCHAR(32)  NOT NULL,
  lastNames   VARCHAR(30),
  name  VARCHAR(20),
  email VARCHAR(50)  NOT NULL,
  registerDate TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
  lastConnection TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
  state    TINYINT UNSIGNED NOT NULL DEFAULT 0,
  downloads INT(3) UNSIGNED NOT NULL DEFAULT 0,


  PRIMARY KEY  (id),
  UNIQUE userIDX (companyid, userkey),
  KEY companyIDX (companyid),
  FOREIGN KEY  (companyid) REFERENCES companies(id) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;

CREATE TABLE userPictures (
  userid  INT(10) UNSIGNED NOT NULL,
  mimetype VARCHAR(32),
  picture MEDIUMBLOB,
  PRIMARY KEY (userid),
  FOREIGN KEY (userid) REFERENCES users(id) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;


```

generatorConfig.xml

A continuación exponemos el archivo de configuración necesario para generar el código fuente de nuestra aplicación.

Existen una gran **cantidad de parámetros opcionales** que nos permitirián adaptar la generación a nuestras necesidades y que se encuentran **bastante bien documentadas** en el archivo zip de descarga.

view plain print ?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
  "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>
    <!-- Ubicación del JAR JDBC del gestor de Base de datos a usar, en este caso MySQL  --
>
  <classPathEntry location="mysql-connector-java-5.1.14.jar" />

  <!--  Generaremos para MyBatis versión 3 -->
  <context id="DB1" targetRuntime="MyBatis3">

    <!-- No generamos los comentarios -->
    <commentGenerator>
      <property name="suppressAllComments" value="true"/>
    </commentGenerator>

      <!--  Parámetros de conexión a la bd -->
    <jdbcConnection driverClass="com.mysql.jdbc.Driver"
        connectionURL="jdbc:mysql://localhost:3306/cgarcia1"
        userId="invitado"
        password="invitado">
    </jdbcConnection>
```

```
    <javaModelGenerator targetPackage="es.carlosgarcia.dao.model" targetProject="salida">
      <property name="enableSubPackages" value="true" />
    </javaModelGenerator>

    <sqlMapGenerator targetPackage="es.carlosgarcia.dao.sqlmap"  targetProject="salida">
      <property name="enableSubPackages" value="true" />
    </sqlMapGenerator>

    <!-- También podríamos indicar el tipo ANNOTATEDMAPPER -->
    <javaClientGenerator type="XMLMAPPER" targetPackage="es.carlosgarcia.dao"  targetProject
      <property name="enableSubPackages" value="true" />
    </javaClientGenerator>

    <!--
 Generamos el código fuente de todas las tablas evitando la generación varios métodos -->
    <table tableName="%" enableCountByExample="false" enableDeleteByExample="false"
           enableUpdateByExample="false" selectByPrimaryKeyQueryId="false"
           selectByExampleQueryId="false" enableSelectByExample="false" modelType="flat">
      <property name="useActualColumnNames" value="true"/>
    </table>

  </context>
</generatorConfiguration>
```

Suponiendo que el archivo de configuración `generatorConfig.xml`, el driver JDBC y el JAR del generador de código están en el mismo directorio, ejecutamos el comando:

```
java -jar mybatis-generator-core-1.3.1.jar -configfile generatorConfig.xml -overwrite
```

Captura de pantalla del resultado de la generación de código:

Nombre	Fecha de modificación	Tamaño	Clase
CompaniesMapper.java	Hoy, 20:01	4 KB	Java Source File
model	Hoy, 20:01	--	Carpeta
Companies.java	Hoy, 20:01	4 KB	Java Source File
Teachernotes.java	Hoy, 20:01	4 KB	Java Source File
Teachernotescategories.java	Hoy, 20:01	4 KB	Java Source File
Teachers.java	Hoy, 20:01	4 KB	Java Source File
Userpictures.java	Hoy, 20:01	4 KB	Java Source File
Users.java	Hoy, 20:01	4 KB	Java Source File
sqlmap	Hoy, 20:01	--	Carpeta
CompaniesMapper.xml	Hoy, 20:01	8 KB	XML Source File
TeachernotescategoriesMapper.xml	Hoy, 20:01	4 KB	XML Source File
TeachernotesMapper.xml	Hoy, 20:01	8 KB	XML Source File
TeachersMapper.xml	Hoy, 20:01	8 KB	XML Source File
UserpicturesMapper.xml	Hoy, 20:01	4 KB	XML Source File
UsersMapper.xml	Hoy, 20:01	8 KB	XML Source File
TeachernotescategoriesMapper.java	Hoy, 20:01	4 KB	Java Source File
TeachernotesMapper.java	Hoy, 20:01	4 KB	Java Source File
TeachersMapper.java	Hoy, 20:01	4 KB	Java Source File
UserpicturesMapper.java	Hoy, 20:01	4 KB	Java Source File
UsersMapper.java	Hoy, 20:01	4 KB	Java Source File

es.carlosgarcia.dao.model.Companies

```
view plain  print  ?
01. package es.carlosgarcia.dao.model;
02.
03. public class Companies {
04.     private Integer id;
05.     private String companykey;
06.     private String name;
07.     private String url;
08.     private Boolean publicity;
09.     private Integer userlicenses;
10.     private Integer teacherlicenses;
11.     private String email;
12.     private Boolean state;
13.     private Boolean messages;
14.     private Boolean notes;
15.     private String autoRegisterPwd;
16.     private String language;
17.     private Integer hddMbSpace;
18.
19.     // ......................
20.     // Getters/Setters
21.     // ......................
22.   }
```

es.carlosgarcia.dao.CompaniesMapper.java

```
view plain  print  ?
```

```java
package es.carlosgarcia.dao;

import es.carlosgarcia.dao.model.Companies;

public interface CompaniesMapper {
    int deleteByPrimaryKey(Integer id);
    int insert(Companies record);
    int insertSelective(Companies record);
    Companies selectByPrimaryKey(Integer id);
    int updateByPrimaryKeySelective(Companies record);
    int updateByPrimaryKey(Companies record);
}
```

es.carlosgarcia.dao.sqlmap.CompaniesMapper.xml

```
view plain  print  ?
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-
//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="es.carlosgarcia.dao.CompaniesMapper" >
  <resultMap id="BaseResultMap" type="es.carlosgarcia.dao.model.Companies" >
    <id column="id" property="id" jdbcType="INTEGER" />
    <result column="companykey" property="companykey" jdbcType="VARCHAR" />
    <result column="name" property="name" jdbcType="VARCHAR" />
    <result column="url" property="url" jdbcType="VARCHAR" />
    <result column="publicity" property="publicity" jdbcType="BIT" />
    <result column="userlicenses" property="userlicenses" jdbcType="INTEGER" />
    <result column="teacherlicenses" property="teacherlicenses" jdbcType="INTEGER" />
    <result column="email" property="email" jdbcType="VARCHAR" />
    <result column="state" property="state" jdbcType="BIT" />
    <result column="messages" property="messages" jdbcType="BIT" />
    <result column="notes" property="notes" jdbcType="BIT" />
    <result column="autoRegisterPwd" property="autoRegisterPwd" jdbcType="VARCHAR" />
    <result column="language" property="language" jdbcType="VARCHAR" />
    <result column="hddMbSpace" property="hddMbSpace" jdbcType="INTEGER" />
  </resultMap>
  <sql id="Base_Column_List" >
    id, companykey, name, url, publicity, userlicenses, teacherlicenses, email, state,
    messages, notes, autoRegisterPwd, language, hddMbSpace
  </sql>
  <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Integer
    select 'false' as QUERYID,
    <include refid="Base_Column_List" />
    from companies
    where id = #{id,jdbcType=INTEGER}
  </select>
  <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer" >
    delete from companies
    where id = #{id,jdbcType=INTEGER}
  </delete>
  <insert id="insert" parameterType="es.carlosgarcia.dao.model.Companies" >
    insert into companies (id, companykey, name,
      url, publicity, userlicenses,
      teacherlicenses, email, state,
      messages, notes, autoRegisterPwd,
      language, hddMbSpace)
    values (#{id,jdbcType=INTEGER}, #{companykey,jdbcType=VARCHAR}, #{name,jdbcType=VARCHAR}
      #{url,jdbcType=VARCHAR}, #{publicity,jdbcType=BIT}, #{userlicenses,jdbcType=INTEGER},
      #{teacherlicenses,jdbcType=INTEGER}, #{email,jdbcType=VARCHAR}, #{state,jdbcType=BIT},
      #{messages,jdbcType=BIT}, #{notes,jdbcType=BIT}, #{autoRegisterPwd,jdbcType=VARCHAR},
      #{language,jdbcType=VARCHAR}, #{hddMbSpace,jdbcType=INTEGER})
  </insert>
  <insert id="insertSelective" parameterType="es.carlosgarcia.dao.model.Companies" >
    insert into companies
    <trim prefix="(" suffix=")" suffixOverrides="," >
      <if test="id != null" >
        id,
      </if>
      <if test="companykey != null" >
        companykey,
      </if>
      <if test="name != null" >
        name,
      </if>
      <if test="url != null" >
        url,
      </if>
      <if test="publicity != null" >
        publicity,
      </if>
      <if test="userlicenses != null" >
        userlicenses,
      </if>
      <if test="teacherlicenses != null" >
        teacherlicenses,
      </if>
      <if test="email != null" >
        email,
      </if>
      <if test="state != null" >
        state,
      </if>
      <if test="messages != null" >
        messages,
```

```
      </if>
      <if test="notes != null" >
        notes,
      </if>
      <if test="autoRegisterPwd != null" >
        autoRegisterPwd,
      </if>
      <if test="language != null" >
        language,
      </if>
      <if test="hddMbSpace != null" >
        hddMbSpace,
      </if>
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides="," >
      <if test="id != null" >
        #{id,jdbcType=INTEGER},
      </if>
      <if test="companykey != null" >
        #{companykey,jdbcType=VARCHAR},
      </if>
      <if test="name != null" >
        #{name,jdbcType=VARCHAR},
      </if>
      <if test="url != null" >
        #{url,jdbcType=VARCHAR},
      </if>
      <if test="publicity != null" >
        #{publicity,jdbcType=BIT},
      </if>
      <if test="userlicenses != null" >
        #{userlicenses,jdbcType=INTEGER},
      </if>
      <if test="teacherlicenses != null" >
        #{teacherlicenses,jdbcType=INTEGER},
      </if>
      <if test="email != null" >
        #{email,jdbcType=VARCHAR},
      </if>
      <if test="state != null" >
        #{state,jdbcType=BIT},
      </if>
      <if test="messages != null" >
        #{messages,jdbcType=BIT},
      </if>
      <if test="notes != null" >
        #{notes,jdbcType=BIT},
      </if>
      <if test="autoRegisterPwd != null" >
        #{autoRegisterPwd,jdbcType=VARCHAR},
      </if>
      <if test="language != null" >
        #{language,jdbcType=VARCHAR},
      </if>
      <if test="hddMbSpace != null" >
        #{hddMbSpace,jdbcType=INTEGER},
      </if>
    </trim>
  </insert>
  <update id="updateByPrimaryKeySelective" parameterType="es.carlosgarcia.dao.model.Companie
    update companies
    <set >
      <if test="companykey != null" >
        companykey = #{companykey,jdbcType=VARCHAR},
      </if>
      <if test="name != null" >
        name = #{name,jdbcType=VARCHAR},
      </if>
      <if test="url != null" >
        url = #{url,jdbcType=VARCHAR},
      </if>
      <if test="publicity != null" >
        publicity = #{publicity,jdbcType=BIT},
      </if>
      <if test="userlicenses != null" >
        userlicenses = #{userlicenses,jdbcType=INTEGER},
      </if>
      <if test="teacherlicenses != null" >
        teacherlicenses = #{teacherlicenses,jdbcType=INTEGER},
      </if>
      <if test="email != null" >
        email = #{email,jdbcType=VARCHAR},
      </if>
      <if test="state != null" >
        state = #{state,jdbcType=BIT},
      </if>
      <if test="messages != null" >
        messages = #{messages,jdbcType=BIT},
      </if>
      <if test="notes != null" >
        notes = #{notes,jdbcType=BIT},
      </if>
      <if test="autoRegisterPwd != null" >
        autoRegisterPwd = #{autoRegisterPwd,jdbcType=VARCHAR},
      </if>
      <if test="language != null" >
        language = #{language,jdbcType=VARCHAR},
```

```
175.        </if>
176.        <if test="hddMbSpace != null" >
177.          hddMbSpace = #{hddMbSpace,jdbcType=INTEGER},
178.        </if>
179.      </set>
180.      where id = #{id,jdbcType=INTEGER}
181.    </update>
182.    <update id="updateByPrimaryKey" parameterType="es.carlosgarcia.dao.model.Companies" >
183.      update companies
184.      set companykey = #{companykey,jdbcType=VARCHAR},
185.        name = #{name,jdbcType=VARCHAR},
186.        url = #{url,jdbcType=VARCHAR},
187.        publicity = #{publicity,jdbcType=BIT},
188.        userlicenses = #{userlicenses,jdbcType=INTEGER},
189.        teacherlicenses = #{teacherlicenses,jdbcType=INTEGER},
190.        email = #{email,jdbcType=VARCHAR},
191.        state = #{state,jdbcType=BIT},
192.        messages = #{messages,jdbcType=BIT},
193.        notes = #{notes,jdbcType=BIT},
194.        autoRegisterPwd = #{autoRegisterPwd,jdbcType=VARCHAR},
195.        language = #{language,jdbcType=VARCHAR},
196.        hddMbSpace = #{hddMbSpace,jdbcType=INTEGER}
197.      where id = #{id,jdbcType=INTEGER}
198.    </update>
199.  </mapper>
200.
201.
```

Conclusiones

Aunque este tipo de herramientas no cubrán el 100% de vuestras necesidades (por ejemplo, joins entre tablas), pueden ahorrarnos mucho tiempo y evitarnos las tareas repetitivas y desmotivadoras de los proyectos.

Bueno después de esta introducción, me despido esperando que os haya resultado de utilidad y dejándoos a vosotros profundizar en sus posibilidades y limitaciones.

Un saludo.
Carlos García.

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

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

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

COMENTARIOS