

# ¿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

Tareas programadas (Quartz)  
Gestor documental (Alfresco)  
Inversión de control (Spring)

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

BPM (jBPM o Bonita)  
Generación de informes (JasperReport)  
ESB (Open ESB)



E-mail:

Contraseña:

[Deseo registrarme](#)  
[He olvidado mis datos de acceso](#)

Estás en: [Inicio](#) » [Tutoriales](#) » [MyBatis Generator \(MGB\): Generador de código para MyBatis e iBATIS](#)



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



[Share](#) | [Facebook](#) [Twitter](#) [Print](#)

[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 introducidos 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 mínimas.

En este tutorial vamos a hacer una introducción de **MyBatis Generator (MGB)** un generador de código para el **framework de persistencia 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),
```

Catálogo de servicios  
Autentia



### Últimas Noticias

- [Pirineos on Tour](#)
- [VII Autentia Cycling Day](#)
- [Autentia patrocina la charla sobre Java SE 7 en Madrid](#)
- [Alfresco Day 2011](#)
- [XVII Charla Autentia - Grails - Vídeos y Material](#)

[Histórico de NOTICIAS](#)

### Últimos Tutoriales

- [Implementación de una máquina de estados](#)
- [Cómo trabajar con branch en SmartGit](#)
- [Como intentar averiguar el juego de caracteres de un archivo](#)
- [Uso del componente remoteCommand de primefaces para actualizar el contenedor de un componente de lightBox en modo iframe](#)
- [Cómo trabajar con JSF2 y el soporte de inyección de dependencias de Spring](#)

### Últimos Tutoriales del Autor

- [Android: Leer correos de Gmail](#)
- [Construcción de un control personalizado en Android](#)
- [Desarrollo de aplicaciones mixtas \(web/nativa\) en Android](#)
- [toi18n, Traduce tus aplicaciones de forma rápida Online](#)
- [Haaala! para Android: Facebook y Email con Voz](#)

Síguenos a través de:



Web2PDF

[converted by Web2PDFConvert.com](#)

```

31.     KEY companyIDX (companyid),
32.     FOREIGN KEY (companyid) REFERENCES companies(id) ON DELETE CASCADE ON UPDATE CASCADE
33. ) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;
34.
35.
36.
37. CREATE TABLE teacherNotesCategories (
38.     id INT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
39.     teacherid INT(10) UNSIGNED NOT NULL,
40.     description VARCHAR(128) DEFAULT NULL,
41.
42.     PRIMARY KEY (id),
43.     KEY teacherNotesCategoriesIDX (teacherid),
44.     FOREIGN KEY (teacherid) REFERENCES teachers(id) ON DELETE CASCADE ON UPDATE CASCADE
45. ) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;
46.
47.
48. CREATE TABLE teacherNotes (
49.     id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
50.     teacherid INT(10) UNSIGNED DEFAULT NULL,
51.     categoryid INT(3) UNSIGNED DEFAULT NULL,
52.
53.     published TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
54.     subject VARCHAR(1000) DEFAULT NULL,
55.     content MEDIUMTEXT DEFAULT NULL,
56.     PRIMARY KEY (id),
57.     KEY teacherNotesIDX (teacherid),
58.     KEY teacherCategoryIDX (categoryid),
59.     FOREIGN KEY (teacherid) REFERENCES teachers(id) ON DELETE CASCADE ON UPDATE CASCADE,
60.     FOREIGN KEY (categoryid) REFERENCES teacherNotesCategories(id) ON DELETE SET NULL ON U
61. ) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;
62.
63.
64. CREATE TABLE users (
65.     id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
66.     companyid INT(3) UNSIGNED NOT NULL,
67.     userkey VARCHAR(25) NOT NULL,
68.     pwd VARCHAR(32) NOT NULL,
69.     lastNames VARCHAR(30),
70.     name VARCHAR(20),
71.     email VARCHAR(50) NOT NULL,
72.     registerDate TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
73.     lastConnection TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
74.     state TINYINT UNSIGNED NOT NULL DEFAULT 0,
75.     downloads INT(3) UNSIGNED NOT NULL DEFAULT 0,
76.
77.
78.     PRIMARY KEY (id),
79.     UNIQUE userIDX (companyid, userkey),
80.     KEY companyIDX (companyid),
81.     FOREIGN KEY (companyid) REFERENCES companies(id) ON DELETE CASCADE ON UPDATE CASCADE
82. ) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;
83.
84. CREATE TABLE userPictures (
85.     userid INT(10) UNSIGNED NOT NULL,
86.     mimetype VARCHAR(32),
87.     picture MEDIUMBLOB,
88.     PRIMARY KEY (userid),
89.     FOREIGN KEY (userid) REFERENCES users(id) ON DELETE CASCADE ON UPDATE CASCADE
90. ) ENGINE=InnoDB CHARSET=utf8 collate=utf8_general_ci;
91.
92.

```

## 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 permitirían adaptar la generación a nuestras necesidades y que se encuentran **bastante bien documentadas** en el archivo zip de descarga.

```

view plain print ?
01. <?xml version="1.0" encoding="UTF-8"?>
02. <!DOCTYPE generatorConfiguration
03. PUBLIC "-//mybatis.org/DTD MyBatis Generator Configuration 1.0//EN"
04. "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
05.
06. <generatorConfiguration>
07.     <!-- Ubicación del JAR JDBC del gestor de Base de datos a usar, en este caso MySQL -->
08.     <classPathEntry location="mysql-connector-java-5.1.14.jar" />
09.
10.     <!-- Generaremos para MyBatis versión 3 -->
11.     <context id="DB1" targetRuntime="MyBatis3">
12.
13.         <!-- No generamos los comentarios -->
14.         <commentGenerator>
15.             <property name="suppressAllComments" value="true"/>
16.         </commentGenerator>
17.
18.         <!-- Parámetros de conexión a la bd -->
19.         <jdbcConnection driverClass="com.mysql.jdbc.Driver"
20.             connectionURL="jdbc:mysql://localhost:3306/cgarcia1"
21.             userId="invitado"
22.             password="invitado">
23.         </jdbcConnection>

```

## Últimas ofertas de empleo

2011-07-06

 Otras Sin catalogar - LUGO.

2011-06-20

 Comercial - Ventas - SEVILLA.

2011-05-24

 Contabilidad - Especialista Contable - BARCELONA.

2011-05-14

 Comercial - Ventas - TARRAGONA.

2011-04-13

 Comercial - Ventas - VALENCIA.

```

24.
25.     <javaModelGenerator targetPackage="es.carlogarcia.dao.model" targetProject="salida">
26.         <property name="enableSubPackages" value="true" />
27.     </javaModelGenerator>
28.
29.     <sqlMapGenerator targetPackage="es.carlogarcia.dao.sqlmap" targetProject="salida">
30.         <property name="enableSubPackages" value="true" />
31.     </sqlMapGenerator>
32.
33.     <!-- También podríamos indicar el tipo ANNOTATEDMAPPER -->
34.     <javaClientGenerator type="XMLMAPPER" targetPackage="es.carlogarcia.dao" targetProject="salida">
35.         <property name="enableSubPackages" value="true" />
36.     </javaClientGenerator>
37.
38.     <!--
39.     Generamos el código fuente de todas las tablas evitando la generación varios métodos -->
40.     <table tableName="%" enableCountByExample="false" enableDeleteByExample="false"
41.         enableUpdateByExample="false" selectByPrimaryKeyQueryId="false"
42.         selectByExampleQueryId="false" enableSelectByExample="false" modelType="flat">
43.         <property name="useActualColumnNames" value="true"/>
44.     </table>
45.
46. </context>
47. </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.carlogarcia.dao.model.Companies

```

view plain print ?
01. package es.carlogarcia.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.carlogarcia.dao.CompaniesMapper.java

```
view plain print ?
```

```

01. package es.carlosgarcia.dao;
02.
03. import es.carlosgarcia.dao.model.Companies;
04.
05. public interface CompaniesMapper {
06.     int deleteByPrimaryKey(Integer id);
07.     int insert(Companies record);
08.     int insertSelective(Companies record);
09.     Companies selectByPrimaryKey(Integer id);
10.     int updateByPrimaryKeySelective(Companies record);
11.     int updateByPrimaryKey(Companies record);
12. }

```

es.carlosgarcia.dao.sqlmap.CompaniesMapper.xml

```

view plain print ?
01. <?xml version="1.0" encoding="UTF-8"?>
02. <!DOCTYPE mapper PUBLIC "-
03. //mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
04. <mapper namespace="es.carlosgarcia.dao.CompaniesMapper" >
05.     <resultMap id="BaseResultMap" type="es.carlosgarcia.dao.model.Companies" >
06.         <id column="id" property="id" jdbcType="INTEGER" />
07.         <result column="companykey" property="companykey" jdbcType="VARCHAR" />
08.         <result column="name" property="name" jdbcType="VARCHAR" />
09.         <result column="url" property="url" jdbcType="VARCHAR" />
10.         <result column="publicity" property="publicity" jdbcType="BIT" />
11.         <result column="userlicenses" property="userlicenses" jdbcType="INTEGER" />
12.         <result column="teacherlicenses" property="teacherlicenses" jdbcType="INTEGER" />
13.         <result column="email" property="email" jdbcType="VARCHAR" />
14.         <result column="state" property="state" jdbcType="BIT" />
15.         <result column="messages" property="messages" jdbcType="BIT" />
16.         <result column="notes" property="notes" jdbcType="BIT" />
17.         <result column="autoRegisterPwd" property="autoRegisterPwd" jdbcType="VARCHAR" />
18.         <result column="language" property="language" jdbcType="VARCHAR" />
19.         <result column="hddMbSpace" property="hddMbSpace" jdbcType="INTEGER" />
20.     </resultMap>
21.     <sql id="Base_Column_List" >
22.         id, companykey, name, url, publicity, userlicenses, teacherlicenses, email, state,
23.         messages, notes, autoRegisterPwd, language, hddMbSpace
24.     </sql>
25.     <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Integer" >
26.         select 'false' as QUERYID,
27.         <include refid="Base_Column_List" />
28.         from companies
29.         where id = #{id,jdbcType=INTEGER}
30.     </select>
31.     <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer" >
32.         delete from companies
33.         where id = #{id,jdbcType=INTEGER}
34.     </delete>
35.     <insert id="insert" parameterType="es.carlosgarcia.dao.model.Companies" >
36.         insert into companies (id, companykey, name,
37.         url, publicity, userlicenses,
38.         teacherlicenses, email, state,
39.         messages, notes, autoRegisterPwd,
40.         language, hddMbSpace)
41.         values (#{id,jdbcType=INTEGER}, #{companykey,jdbcType=VARCHAR}, #{name,jdbcType=VARCHAR},
42.         #{url,jdbcType=VARCHAR}, #{publicity,jdbcType=BIT}, #{userlicenses,jdbcType=INTEGER},
43.         #{teacherlicenses,jdbcType=INTEGER}, #{email,jdbcType=VARCHAR}, #{state,jdbcType=BIT},
44.         #{messages,jdbcType=BIT}, #{notes,jdbcType=BIT}, #{autoRegisterPwd,jdbcType=VARCHAR},
45.         #{language,jdbcType=VARCHAR}, #{hddMbSpace,jdbcType=INTEGER})
46.     </insert>
47.     <insert id="insertSelective" parameterType="es.carlosgarcia.dao.model.Companies" >
48.         insert into companies
49.         <trim prefix="(" suffix=")" suffixOverrides="," >
50.             <if test="id != null" >
51.                 id,
52.             </if>
53.             <if test="companykey != null" >
54.                 companykey,
55.             </if>
56.             <if test="name != null" >
57.                 name,
58.             </if>
59.             <if test="url != null" >
60.                 url,
61.             </if>
62.             <if test="publicity != null" >
63.                 publicity,
64.             </if>
65.             <if test="userlicenses != null" >
66.                 userlicenses,
67.             </if>
68.             <if test="teacherlicenses != null" >
69.                 teacherlicenses,
70.             </if>
71.             <if test="email != null" >
72.                 email,
73.             </if>
74.             <if test="state != null" >
75.                 state,
76.             </if>
77.             <if test="messages != null" >
78.                 messages,

```

```

78.         </if>
79.         <if test="notes != null" >
80.             notes,
81.         </if>
82.         <if test="autoRegisterPwd != null" >
83.             autoRegisterPwd,
84.         </if>
85.         <if test="language != null" >
86.             language,
87.         </if>
88.         <if test="hddMbSpace != null" >
89.             hddMbSpace,
90.         </if>
91.     </trim>
92. <trim prefix="values (" suffix=")" suffixOverrides="," >
93.     <if test="id != null" >
94.         #{id,jdbcType=INTEGER},
95.     </if>
96.     <if test="companykey != null" >
97.         #{companykey,jdbcType=VARCHAR},
98.     </if>
99.     <if test="name != null" >
100.         #{name,jdbcType=VARCHAR},
101.     </if>
102.     <if test="url != null" >
103.         #{url,jdbcType=VARCHAR},
104.     </if>
105.     <if test="publicity != null" >
106.         #{publicity,jdbcType=BIT},
107.     </if>
108.     <if test="userlicenses != null" >
109.         #{userlicenses,jdbcType=INTEGER},
110.     </if>
111.     <if test="teacherlicenses != null" >
112.         #{teacherlicenses,jdbcType=INTEGER},
113.     </if>
114.     <if test="email != null" >
115.         #{email,jdbcType=VARCHAR},
116.     </if>
117.     <if test="state != null" >
118.         #{state,jdbcType=BIT},
119.     </if>
120.     <if test="messages != null" >
121.         #{messages,jdbcType=BIT},
122.     </if>
123.     <if test="notes != null" >
124.         #{notes,jdbcType=BIT},
125.     </if>
126.     <if test="autoRegisterPwd != null" >
127.         #{autoRegisterPwd,jdbcType=VARCHAR},
128.     </if>
129.     <if test="language != null" >
130.         #{language,jdbcType=VARCHAR},
131.     </if>
132.     <if test="hddMbSpace != null" >
133.         #{hddMbSpace,jdbcType=INTEGER},
134.     </if>
135. </trim>
136. </insert>
137. <update id="updateByPrimaryKeySelective" parameterType="es.carlosgarcia.dao.model.Companie
138. update companies
139. <set >
140.     <if test="companykey != null" >
141.         companykey = #{companykey,jdbcType=VARCHAR},
142.     </if>
143.     <if test="name != null" >
144.         name = #{name,jdbcType=VARCHAR},
145.     </if>
146.     <if test="url != null" >
147.         url = #{url,jdbcType=VARCHAR},
148.     </if>
149.     <if test="publicity != null" >
150.         publicity = #{publicity,jdbcType=BIT},
151.     </if>
152.     <if test="userlicenses != null" >
153.         userlicenses = #{userlicenses,jdbcType=INTEGER},
154.     </if>
155.     <if test="teacherlicenses != null" >
156.         teacherlicenses = #{teacherlicenses,jdbcType=INTEGER},
157.     </if>
158.     <if test="email != null" >
159.         email = #{email,jdbcType=VARCHAR},
160.     </if>
161.     <if test="state != null" >
162.         state = #{state,jdbcType=BIT},
163.     </if>
164.     <if test="messages != null" >
165.         messages = #{messages,jdbcType=BIT},
166.     </if>
167.     <if test="notes != null" >
168.         notes = #{notes,jdbcType=BIT},
169.     </if>
170.     <if test="autoRegisterPwd != null" >
171.         autoRegisterPwd = #{autoRegisterPwd,jdbcType=VARCHAR},
172.     </if>
173.     <if test="language != null" >
174.         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 cubrirán 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)

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

## COMENTARIOS



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

Copyright 2003-2011 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

[W3C XHTML 1.0](#)
[W3C CSS](#)
[XML RSS](#)
[XML ATOM](#)