

**Ανοικτά Ακαδημαϊκά Μαθήματα**

**Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας**

**Βάσεις Δεδομένων ΙΙ (Ε)**

**Ενότητα 10:** Σύνδεση με Βάση Δεδομένων MySql

Χ. Σκουρλάς, Α. Τσολακίδης, Δ. Βάσσης

Τμήμα Μηχανικών Πληροφορικής ΤΕ

|  |  |
| --- | --- |
| Το περιεχόμενο του μαθήματος διατίθεται με άδεια Creative Commons εκτός και αν αναφέρεται διαφορετικά | Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους. |

Περιεχόμενα

[1. Εισαγωγή 3](#_Toc401509236)

[2. Σύνδεση στη βάση δεδομένων MySQL μέσω Java 4](#_Toc401509237)

[3. Insert-delete-update-select statements μέσω Java 8](#_Toc401509238)

[Άσκηση1. Παράδειγμα χρήσης embedded select statement. 11](#_Toc401509239)

Στόχος του εργαστηρίου είναι η εξοικείωση με το JDBC API το οποίο μας παρέχει τη δυνατότητα να διαχειριστούμε μια βάση δεδομένων χρησιμοποιώντας εντολές Java. Μετά την επεξεργασία του εργαστηρίου ο ενδιαφερόμενος θα είναι σε θέση να δημιουργήσει μια εφαρμογή διαχείρισης των δεδομένων μιας βάσης κάνοντας χρήση εντολών jsp.

**Λέξεις κλειδιά**: JDBC, JSP, MySQL.

# Εισαγωγή

**Το JDBC API**

H Java παρέχει ένα προγραμματιστικό περιβάλλον (API) για σύνδεση σε βάσεις δεδομένων, γνωστό ως *Java Database Connectivity (JDBC)*. Το συγκεκριμένο API μας παρέχει τη δυνατότητα να διαχειριστούμε μια βάση δεδομένων μέσω εντολών SQL. Για να χρησιμοποιήσουμε το JDBC API απαιτούνται δυο ομάδες βιβλιοθηκών, τις οποίες πρέπει να φορτώσουμε στο πρόγραμμά μας.

* Η μια μας παρέχει τις απαραίτητες κλάσεις και μεθόδους ώστε να μπορούμε να συνδεθούμε στη συγκεκριμένη βάση δεδομένων. Η συγκεκριμένη ομάδα βιβλιοθηκών είναι ξεχωριστή για κάθε βάση δεδομένων (π.χ. Oracle, MySQL, Access) και παρέχεται από τον εκάστοτε κατασκευαστή της βάσης. Είναι γνωστή ως *JDBC driver*.
* Η δεύτερη μας παρέχει τις απαραίτητες κλάσεις και μεθόδους για να κάνουμε ενέργειες στη βάση δεδομένων, δηλαδή να γράφουμε SQL. Καθώς η SQL είναι ίδια για όλες τις βάσεις δεδομένων, έτσι και η συγκεκριμένη βιβλιοθήκη είναι κοινή, ανεξάρτητα της βάσης που θα χρησιμοποιήσουμε. Η βιβλιοθήκη αυτή περιέχεται ήδη στη Java και είναι το package *java.sql*.

Αν γράψουμε μια εφαρμογή σε Java, που διαχειρίζεται μια συγκεκριμένη βάση δεδομένων (π.χ. MySQL), η εφαρμογή αυτή, με λίγες αλλαγές, θα λειτουργεί και σε άλλη βάση δεδομένων (π.χ. Oracle), αρκεί να αλλάξουμε τον JDBC driver. Για τη MySQL, ο JDBC driver είναι ένα αρχείο jar το οποίο φορτώνουμε στην εφαρμογή μας.

**Εμβάθυνση στην embedded SQL**

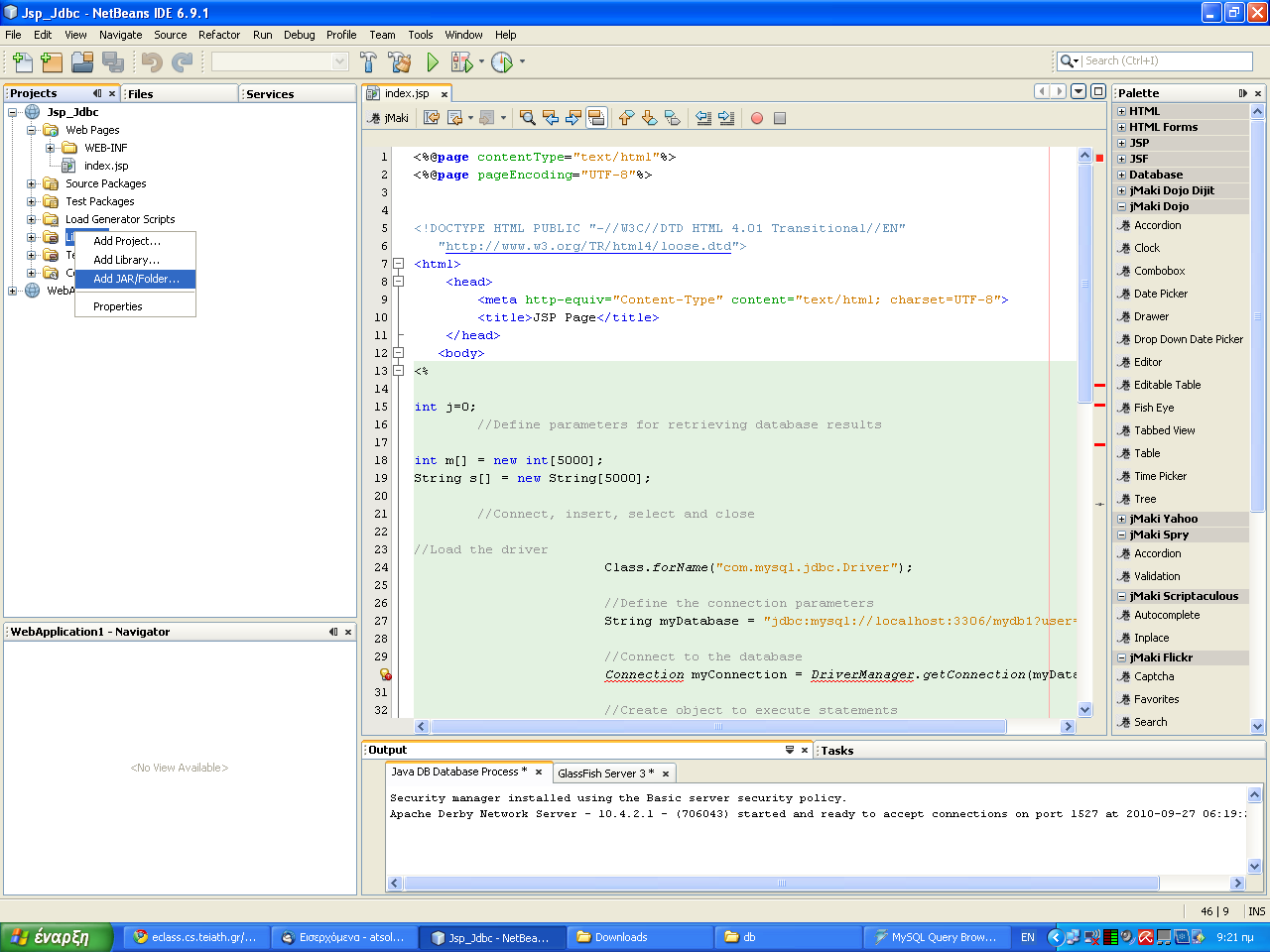
Στο παρόν κεφάλαιο θα εμβαθύνουμε στις ειδικές συναρτήσεις που μας παρέχει η Java για να γράφουμε εντολές SQL. Συγκεκριμένα, θα ασχοληθούμε με την εντολή myStatement.methodName(sqlString) και με τους διάφορους τρόπους που μας παρέχει για να διαχειριζόμαστε πληροφορία από βάση δεδομένων.

# Σύνδεση στη βάση δεδομένων MySQL μέσω Java

Για να συνδεθούμε στη βάση δεδομένων MySQL και να γράψουμε εντολές SQL απαιτούνται οι εξής ενέργειες:

*Δήλωση του JDBC driver στο classpath*

 O JDBC driver για MySQL παρέχεται στη σελίδα της MySQL. Είναι ένα μικρό αρχείο jar με όνομα mysql-connector-java- 5.1.13 -bin.jar ή mysql.jar(e-class). Το αρχείο αυτό πρέπει να το βάλουμε στους καταλόγους jre\_Home/lib/ext και jdk\_Home/jre/lib/ext ή να το δηλώσουμε στην εφαρμογής μας *libraries>δεξί κλικ> Add Jar/Folder* και επιλέγουμε το αρχείο *mysql-connector-java- 5.1.13 -bin.jar ή mysql.jar*



.Συνίσταται η τελευταία λύση, καθώς έτσι θα το βλέπει οποιαδήποτε εφαρμογή το χρειάζεται.

*Εισαγωγή του package java.sql στο πρόγραμμά μας*

Η εισαγωγή ενός package σε πρόγραμμα java γίνεται με την εντολή

import packageName.className ή import packageName.\* (για όλες τις κλάσεις).

Στη δική μας περίπτωση θα πρέπει να γράψουμε την εντολή

import java.sql.\*;

**Φόρτωμα του JDBC driver**

Το φόρτωμα του JDBC driver γίνεται φορτώνοντας την κλάση Driver από το

package com.mysql.jdbc. Το φόρτωμα μιας κλάσης γίνεται μέσω της εντολής

Class.forName(“ClassName”);

Στην περίπτωσή μας πρέπει να γράψουμε την εντολή:

Class.forName(“com.mysql.jdbc.Driver”);

Παρατηρήστε ότι γράψαμε ολόκληρο το path της κλάσης Driver, καθώς αυτή περιέχεται σε package. Θα μπορούσαμε να γράψουμε μόνο το όνομα της κλάσης, αλλά να είχαμε πριν συμπεριλάβει το αντίστοιχο package στο πρόγραμμά μας, μέσω της εντολής

import com.mysql.jdbc

**Σύνδεση στη βάση δεδομένων**

Η σύνδεση στη βάση δεδομένων γίνεται δημιουργώντας ένα αντικείμενο τύπου *Connection*. Μέσω του αντικειμένου αυτού θα μπορούμε να γράφουμε εντολές SQL. Η δημιουργία ενός αντικειμένου τύπου Connection (π.χ. *myConnection*) γίνεται με την παρακάτω εντολή:

Connection myConnection = DriverManager.getConnection(myDatabase);

 MyDatabase είναι μια συμβολοσειρά (string) τη μορφής:

“jdbc:mysql://ServerName:ServerPort/DatabaseName?user=myUsername&password=myPassword";

* *ServerName* είναι το όνομα ή η διεύθυνση ΙΡ του database server.
* *ServerPort* είναι η θύρα επικοινωνίας του database server MySQL (default=3306).
* *DatabaseName* είναι το όνομα της βάσης δεδομένων που θα χρησιμοποιήσουμε.
* *myUsername* και *myPassword* είναι το username και το password που χρησιμοποιούμε για να συνδεθούμε στη mySQL.

 Αν χρησιμοποιούσαμε μια βάση δεδομένων με όνομα test, ο MySQL server έτρεχε τοπικά στο σύστημά μας και συνδεόμασταν με username: teiath και password:aigaleo, τότε οι εντολές για σύνδεση θα είχαν ως εξής:

String testDatabase = “jdbc:mysql://localhost:3306/mydb1?user=teiath&password=aigaleo";

Connection myConnection = DriverManager.getConnection(testDatabase);

**ΠΡΟΣΟΧΗ:** Μη συγχέετε το username και το password με τα οποία συνδεόμαστε στη βάση δεδομένων με το username και password που πιθανόν να απαιτείται να δώσει ο χρήστης για να χρησιμοποιήσει την εφαρμογή που φτιάχνουμε. Τα πρώτα χρησιμοποιούνται από το ίδιο το πρόγραμμα, και μόνο από αυτό, για να συνδεθεί το πρόγραμμα στη βάση δεδομένων. Τα δεύτερα χρησιμοποιούνται από το χρήστη, για τη δική μας εφαρμογή, και τα ορίζουμε εμείς, ή βάζουμε το χρήστη να τα ορίσει.

**Δημιουργία αντικειμένου για να γράψουμε SQL statements**

 Έχοντας το αντικείμενο myConnection, το οποίο στην ουσία είναι μια ανοιχτή σύνδεση με τη βάση δεδομένων, μπορούμε να δημιουργήσουμε αντικείμενο τύπου Statement (και όνομα, π.χ. myStatement) μέσω του οποίου θα γράφουμε εντολές SQL. Αυτό γίνεται με την εντολή:

Statement myStatement = myconnection.createStatement();

**Εγγραφή και εκτέλεση SQL statements**

 Στη συνέχεια γράφουμε την εντολή SQL που θέλουμε να εκτελεστεί. Η συγκεκριμένη εντολή γράφεται σε μορφή string, όπως ακριβώς θα τη γράφαμε αν την εκτελούσαμε από το περιβάλλον της mySQL. Για παράδειγμα:

String sqlString = "INSERT INTO USERS VALUES ('user1', 'password1')";

Για να εκτελέσουμε την εντολή που γράψαμε παραπάνω χρησιμοποιούμε το αντικείμενο myStatement που δημιουργήσαμε, το οποίο μας παρέχει διάφορες μεθόδους, ανάλογα με το τι κάνει η εντολή SQL που δίνουμε. H σύνταξη της εντολής που γράφουμε για να εκτελέσουμε την SQL statement είναι:

myStatement.methodName(sqlString);

Για την εντολή *insert* που γράψαμε παραπάνω, η μέθοδος θα είχε ως εξής:

myStatement.executeUpdate(sqlString);

Αν η sqlString αφορούσε μια SQL statement τύπου select, η μέθοδος θα ήταν:

myStatement.executeQuery(sqlString);

Τις διάφορες μεθόδους των αντικειμένων τύπου Connection για να εκτελούμε SQL statements θα τις δούμε αναλυτικά σε επόμενα κεφάλαια.

**Κλείσιμο του αντικειμένου εγγραφής SQL statements**

Αφού έχουμε τελειώσει με την εκτέλεση των SQL statements πρέπει να κλείσουμε το αντικείμενο εγγραφής SQL statements. Αυτό γίνεται απλά με την εντολή

myStatement.close();

**Κλείσιμο της σύνδεσης**

Τέλος, για να κλείσουμε τη σύνδεση με τη βάση δεδομένων, γράφουμε την εντολή

myConnection.close();

# Insert-delete-update-select statements μέσω Java

Στο προηγούμενο κεφάλαιο ασχοληθήκαμε με τη σύνδεση σε μια βάση δεδομένων MySQL. Μάθαμε ότι, για να γράψουμε μια εντολή SQL, χρησιμοποιούμε τη μέθοδο

myStatement.methodName(sqlString);

όπου myStatement είναι ένα αντικείμενο τύπου Statement και methodName είναι μια από τις πολλές μεθόδους που παρέχει η κλάση Statement. Θα ασχοληθούμε λίγο παραπάνω με τις διάφορες μορφές της μεθόδου methodName.

**Εισαγωγή δεδομένων (insert statement)**

Όπως είδαμε στο προηγούμενο μάθημα, για να κάνουμε εισαγωγή δεδομένων στη

βάση, χρησιμοποιούμε τη μέθοδο executeUpdate μέσω της εντολής:

myStatement.executeUpdate(sqlString);

όπου sqlString είναι μια συμβολοσειρά της μορφής:

String sqlString = "INSERT INTO MYTABLE VALUES ('value1', 'value2')";

**Διαγραφή δεδομένων (delete statement)**

Για διαγραφή δεδομένων χρησιμοποιούμε πάλι τη μέθοδο executeUpdate μέσω της

εντολής:

myStatement.executeUpdate(sqlString);

όπου τώρα sqlString είναι της μορφής:

String sqlString = "DELETE FROM MYTABLE WHERE PARAM = 'paramValue'";

**Ανανέωση δεδομένων (update statement)**

Για ανανέωση δεδομένων χρησιμοποιούμε τη μέθοδο executeUpdate μέσω της

εντολής:

myStatement.executeUpdate(sqlString);

όπου τώρα sqlString είναι της μορφής:

String sqlString = "UPDATE MYTABLE SET PARAM1='value1' WHERE PARAM2='value2'";

**Επιλογή δεδομένων (select statement)**

Για επιλογή δεδομένων χρησιμοποιούμε τη μέθοδο executeQuery μέσω της

εντολής:

ResultSet rs = myStatement.executeQuery(sqlString);

όπου sqlString είναι μια συμβολοσειρά της μορφής:

String sqlString = "SELECT \* FROM MYTABLE WHERE PARAM = 'paramValue'";

Η μέθοδος *executeQuery* μας δίνει τη δυνατότητα να τραβήξουμε από τη βάση κάποια δεδομένα. Τα δεδομένα αυτά όμως πρέπει να τα αποθηκεύσουμε προσωρινά σε κάποια δομή στο πρόγραμμά μας, ώστε να τα χρησιμοποιήσουμε. Η Java μας δίνει μια εύχρηστη δομή μέσω της κλάσης *ResultSet*. Πιο συγκεκριμένα, η μέθοδος *ExecuteQuery* μας επιστρέφει ένα αντικείμενο τύπου *ResultSet*. Μέσα στο αντικείμενο αυτό αποθηκεύονται προσωρινά όλες οι εγγραφές (γραμμές πίνακα) από την ερώτηση που κάναμε στη βάση δεδομένων. Το συγκεκριμένο αντικείμενο μας παρέχει και έναν δείκτη ο οποίος αρχικά δείχνει ΠΡΙΝ από την πρώτη εγγραφή που είναι αποθηκευμένη στο αντικείμενο. Μέσω του αντικειμένου αυτού μπορούμε να επεξεργαστούμε τις εγγραφές στο πρόγραμμά μας. Ας ονομάσουμε το συγκεκριμένο αντικείμενο *rs*. Παρακάτω φαίνεται σχηματικά η διαδικασία που περιγράψαμε προηγουμένως.

Το rs (ως αντικείμενο της κλάσης ResultSet) μας παρέχει μεθόδους για να επεξεργαζόμαστε τις εγγραφές που τραβήξαμε από τη βάση δεδομένων. Οι πιο σημαντικές μέθοδοι ναφέρονται παρακάτω:





Για να πάρουμε τώρα πληροφορία που είναι καταχωρημένη στο αντικείμενο rs,

πρέπει να κάνουμε δυο πράγματα:

* Να μετακινήσουμε τη δείκτη στην εγγραφή που θέλουμε να πάρουμε (ή να τον μετακινούμε διαδοχικά αν θέλουμε να τις πάρουμε όλες). Αυτό γίνεται με μια από τις μεθόδους next(), previous(), first(), last() που είδαμε παραπάνω.
* Να χρησιμοποιήσουμε μια συνάρτηση που θα πάρουμε πληροφορία από μια στήλη. Αυτό γίνεται με τις συναρτήσεις getString, getInt, getFloat getDate κ.τ.λ.

Άσκηση1. Παράδειγμα χρήσης embedded select statement.

Θα δούμε ένα τυπικό παράδειγμα χρήσης της executeQuery, καθώς και του τρόπου που χρησιμοποιούμε τις διάφορες μεθόδους της κλάσης ResultSet.

Ξεκινήστε τη MySQL αν δεν έχει ήδη ξεκινήσει με τα windows, ανοίγοντας μια γραμμή εντολών και γράφοντας

net start mySQL

Συνδεθείτε στη βάση δεδομένων γράφοντας την εντολή

mysql –u root

Συνδεθείτε στη βάση mydb1:

use mydb1

Δημιουργήστε τον πίνακα myTable,

|  |  |
| --- | --- |
| ID | *Int* |
| Name | *Text* |

δείτε τη δομή του και το τι περιέχει, γράφοντας τις εντολές:

describe myTable

select \* from myTable

Έστω ότι έχουμε τον παραπάνω πίνακα με όνομα myTable:

Δημιουργείστε ένα αρχείο test.jsp και γράψτε τον παρακάτω κώδικα:

<[%@page import="java.sql.\*" %](mailto:%25@page%20import=%22java.sql.*%22%20%25)>

<%

int j=0;

//Define parameters for retrieving database results

int m[] = new int[5000];

String s[] = new String[5000];

//Load the driver

Class.forName("com.mysql.jdbc.Driver");

//Define the connection parameters

String myDatabase = "jdbc:mysql://localhost:3306/mydb1?user=root";

//Connect to the database

Connection myConnection = DriverManager.getConnection(myDatabase);

//Create object to execute statements

Statement myStatement = myConnection.createStatement();

//Select everything from the database

String sqlString = "SELECT \* FROM myTable ";

ResultSet rs=myStatement.executeQuery(sqlString);

//Store the results in vectors in order to pass them to the JSP

while(rs.next()){

m[j]=rs.getInt("id");

s[j]=rs.getString("name");

j++;

}

//Close the connection to the database

myStatement.close();

myConnection.close();

for(int i=0; i<j; i++)

{

out.println(i+"\_\_\_"+m[i]+"<br>");

out.println(i+"\_\_\_"+s[i]+"<br>");

}

%>

Στη συνέχεια τρέξτε το πρόγραμμα.

Τα βήματα είναι τα παρακάτω.

1. Ορίζουμε δυο παραμέτρους.
2. Στις παραμέτρους αυτές θα καταχωρήσουμε τις τιμές των δυο πεδίων του πίνακα mytable1.
3. Φορτώνουμε το driver και προετοιμάζουμε τη συμβολοσειρά για σύνδεση.
4. Συνδεόμαστε τοπικά, με username root και κενό password, στη βάση δεδομένων mydb1.
5. Συνδεόμαστε στη βάση δεδομένων, δημιουργούμε το αντικείμενο τύπου statement και προετοιμάζουμε την SQL string.
6. Εκτελούμε το SQL Statement.
7. Μέχρι στιγμής έχουμε καταφέρει όλο τον πίνακα να τον καταχωρήσουμε στο αντικείμενο rs. Μένει να πάρουμε τα στοιχεία του rs. Αυτό γίνεται με έναν βρόχο while.
8. Η εντολή while(rs.next()) έχει διπλό ρόλο:Εκτελεί την εντολή rs.next().Ελέγχει αν rs.next()=true. Αυτό σημαίνει ότι, αν rs.next()=true, τότε ο βρόχος θα εκτελεστεί. Αν rs.next()=false τότε ο βρόχος δε θα εκτελεστεί και το πρόγραμμα θα προχωρήσει παρακάτω. Βλέποντας τον πίνακα με τις μεθόδους της resultSet, φαίνεται ότι η rs.next() επιστρέφει true αν υπάρχει επόμενη εγγραφή για να διαβάσει. Αν δεν υπάρχει επόμενη εγγραφή, δηλαδή έχει φτάσει στο τέλος, τότε επιστρέφει false. Με τον τρόπο αυτό, αν υπάρχει επόμενη εγγραφή, ο βρόχος εκτελείται, και σταματά να εκτελείται όταν οι εγγραφές τελειώσουν. Ο βρόχος κάνει το εξής:
   1. Διαβάζει, τα πεδία της τρέχουσας εγγραφής και τα καταχωρεί στις αντίστοιχες μεταβλητές του προγράμματος (m[j], (m[j],), μέσω της μεθόδου getString() ή getInt().
   2. Τυπώνει στην οθόνη τις παραπάνω μεταβλητές.

|  |
| --- |
| **Ανοικτά Ακαδημαϊκά Μαθήματα**  **Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας** |
| **Τέλος Ενότητας** |
| **Χρηματοδότηση**   * Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα. * Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Αθήνας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού. * Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους. |

**Σημειώματα**

**Σημείωμα Αναφοράς**

Copyright ΤΕΙ Αθήνας, Χ. Σκουρλάς, Α. Τσολακίδης, Δ. Βάσσης 2014. Χ. Σκουρλάς, Α. Τσολακίδης, Δ. Βάσσης. «Βάσεις Δεδομένων ΙΙ (Ε). Ενότητα 10: Σύνδεση με Βάση Δεδομένων MySql». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: [ocp.teiath.gr](https://ocp.teiath.gr/).

**Σημείωμα Αδειοδότησης**

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό. Οι όροι χρήσης των έργων τρίτων επεξηγούνται στη διαφάνεια «Επεξήγηση όρων χρήσης έργων τρίτων».

Τα έργα για τα οποία έχει ζητηθεί άδεια αναφέρονται στο «Σημείωμα Χρήσης Έργων Τρίτων».

[](file:///C:\Users\pantelis\Downloads\%5b1%5d%20http:\creativecommons.org\licenses\by-nc-sa\4.0\)

[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

* που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
* που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
* που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

**Επεξήγηση όρων χρήσης έργων τρίτων**

|  |  |
| --- | --- |
| © | Δεν επιτρέπεται η επαναχρησιμοποίηση του έργου, παρά μόνο εάν ζητηθεί εκ νέου άδεια από το δημιουργό. |
| διαθέσιμο με άδεια CC-BY | Επιτρέπεται η επαναχρησιμοποίηση του έργου και η δημιουργία παραγώγων αυτού με απλή αναφορά του δημιουργού. |
| διαθέσιμο με άδεια CC-BY-SA | Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού, και διάθεση του έργου ή του παράγωγου αυτού με την ίδια άδεια. |
| διαθέσιμο με άδεια CC-BY-ND | Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η δημιουργία παραγώγων του έργου. |
| διαθέσιμο με άδεια CC-BY-NC | Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η εμπορική χρήση του έργου. |
| διαθέσιμο με άδεια CC-BY-NC-SA | Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού και διάθεση του έργου ή του παράγωγου αυτού με την ίδια άδεια. Δεν επιτρέπεται η εμπορική χρήση του έργου. |
| διαθέσιμο με άδεια CC-BY-NC-ND | Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η εμπορική χρήση του έργου και η δημιουργία παραγώγων του. |
| διαθέσιμο με άδεια CC0 Public Domain | Επιτρέπεται η επαναχρησιμοποίηση του έργου, η δημιουργία παραγώγων αυτού και η εμπορική του χρήση, χωρίς αναφορά του δημιουργού. |
| διαθέσιμο ως κοινό κτήμα | Επιτρέπεται η επαναχρησιμοποίηση του έργου, η δημιουργία παραγώγων αυτού και η εμπορική του χρήση, χωρίς αναφορά του δημιουργού. |
| χωρίς σήμανση | Συνήθως δεν επιτρέπεται η επαναχρησιμοποίηση του έργου. |

**Διατήρηση Σημειωμάτων**

* Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
* το Σημείωμα Αναφοράς
* το Σημείωμα Αδειοδότησης
* τη δήλωση Διατήρησης Σημειωμάτων
* το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους.