Ανάπτυξη Διαδικτυακών Τόπων

Τεχνολογία JDBC

JDBC -Προεπισκόπηση

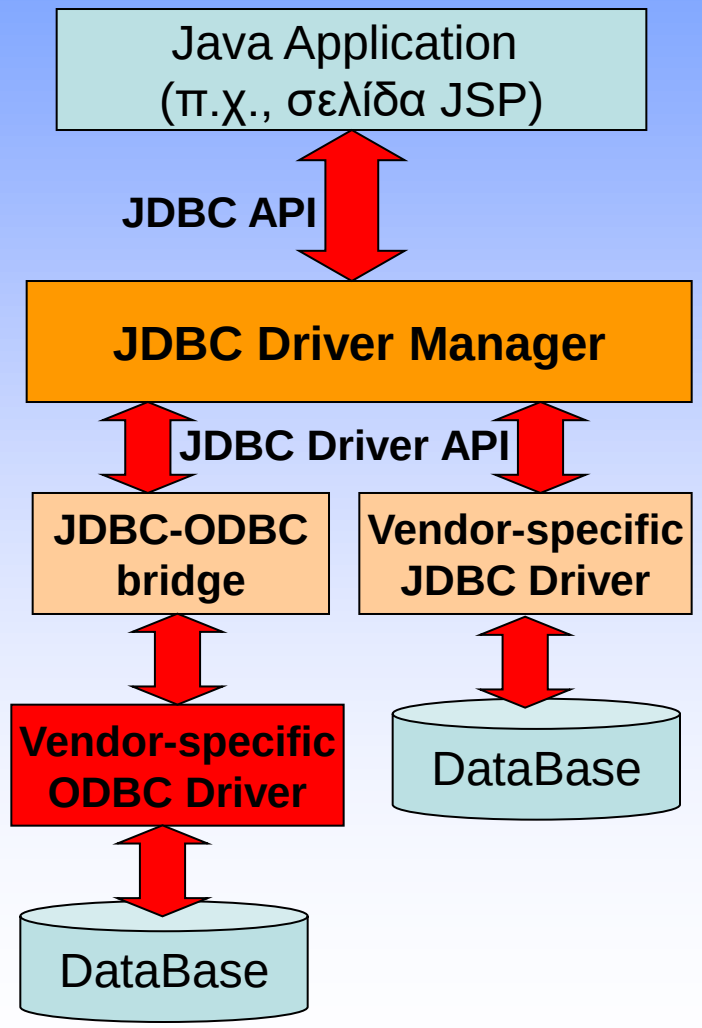
- Τι Είναι?
  - Μια βιβλιοθήκη της Sun για πρόσβαση σε Βάσεις Δεδομένων (μέσω Java)
- Τι παρέχει?
  - Ένα API το οποίο προδιαγράφει με ανοιχτό τρόπο:
    - τη διασύνδεση με οποιαδήποτε βάση δεδομένων
    - Μεθόδους για υποβολή ερωτήσεων (queries)
    - Μεθόδους για δημιουργία παραμετροποιημένων ερωτήσεων(queries)
    - Κατάλληλες δομές δεδομένων για την λήψη αποτελεσμάτων
    - Μεθόδους για την ανάγνωση μετά-πληροφορίας από τη Βάση
  - Το API δεν προδιαγράφει το συντακτικό της SQL

Υλικό στο Διαδικτυο

- Sun’s JDBC Site
  - http://java.sun.com/products/jdbc
- JDBC Tutorial
  - http://java.sun.com/docs/books/tutorial/jdbc
- Λίστα διαθέσιμων JDBC drivers
  - http://industry.java.sun.com/products/jdbc/drivers
- JDBC API
  - http://java.sun.com/j2se/1.4/docs/api/java/sql/package-summary.html

Δομή και Λειτουργία

- Το JDBC αποτελείται από 2 τμήματα:
  - JDBC API, βασισμένο πλήρως σε Java
  - JDBC Driver Manager, ο οποίος χειρίζεται την επικοινωνία με τον vendor-specific driver ο οποίος πραγματοποιεί την πραγματική επικοινωνία με την βάση



JDBC Drivers

- 4 τύποι:
  - JDBC-ODBC bridge + ODBC driver
  - Native API partly Java driver
  - JDBC-Net pure Java Driver
  - Native protocol pure Java Driver

JDBC Τύποι Δεδομένων

Είδος Δεδομένων	Τύπος JDBC	Τύπος JAVA	Προτεινόμενη μέθοδος προσπέλασης
Boolean	**BIT**	**boolean**	
Integer Data	**TINYINT**	**byte**	
	SMALLINT	**short**	**getShort()**
	INTEGER	**int**	**getInt()**
	BIGINT	**long**	
Floating-point Data	**REAL**	**float**	
	FLOAT **DOUBLE**	**double**	**getDouble()**
Binary Data	**BINARY** **VARBINARY** **LONGVARBINARY**	**byte[]**	
Textual Data	**CHAR** **VARCHAR** **LONGVARCHAR**	**String**	**getString()**
Date & Time	**DATE**	**java.sql.Date**	**getDate()**
	TIME **TIMESTAMP**	**java.sql.Time** **java.sql.Timestamp**	**getTime()** **getTimestamp()**
Other types	**NUMERIC** **DECIMAL**	**BigDecimal**	
	CLOB	**Clob***	
	BLOB	**Blob***	
	ARRAY	**Array***	
	DISTINCT	**Mapping of underlying type**	
	STRUCT	**Struct***	
	REF	**Ref***	
	JAVA_OBJECT	**Underlying java class**	

Πώς Χρησιμοποιείται η τεχνολογία JDBC

- Εφτά Βασικά Βήματα:
  1) Φόρτωμα του Driver
  2) Ορισμός της σύνδεσης με τη Βάση
  3) Δημιουργία σύνδεσης
  4) Δημιουργία αντικειμένου ερωτήσεων
  5) Υποβολή ερώτησεων
  6) Επεξεργασία αποτελεσμάτων
  7) Κλείσιμο της σύνδεσης

Βήμα 1ο

- Φόρτωμα του driver (οδηγού)

```
try {
     Class.forName("oracle.jdbc.driver.OracleDriver");
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException error) {
     System.err.println("Error loading driver:"+error);
}
```

Βήμα 2ο

- Ορισμός της σύνδεσης με τη Βάση:
  - Γενική μορφή: jdbc:<subprotocol>:<subname>

```
//example of an oracle JDBC URL
String host="dbhost.mycompany.gr";
String dbName="MyDatabase";
int port=1234;
String oracleURL="jdbc:oracle:thin:@"+host+":"+port+":"+dbName;

//example of a standard Microsoft Access JDBC:ODBC bridged URL
String dbName=”MyOdbcSource”;
String MSAccessURL="jdbc:odbc:"+dbName;

//example of a Microsoft Access DSNless JDBC:ODBC bridged URL
String MSAccessDsnless="jdbc:odbc:DRIVER={Microsoft Access Driver
     (*.mdb)};DBQ=c:\\path\\to\\your\\Database.mdb";
```

Βήμα 3ο

- Δημιουργία σύνδεσης με τη Βάση

```
//example of an oracle JDBC connection
String username="Admin";
String password="123456";
Connection connection =
     DriverManager.getConnection(OracleURL,username,password);


//example of a standard Microsoft Access connection using JDBC:ODBC
Connection connection=DriverManager.getConnection(MSAccessURL,"","");

//example of a Microsoft Access connection using DSNless JDBC:ODBC
Connection connection=DriverManager.getConnection(MSAccessDsnless,"","");
```

Βήμα 4ο

- Δημιουργία αντικειμένου ερωτήσεων προς τη βάση

```
Statement statement = connection.createStatement();
```

Βήμα 5ο

- Υποβολή ερωτήσεων (queries) προς τη Βάση

```
//define the query using the SQL syntax
String query="SELECT col1, col2, col3 FROM table1";
String updateQuery="Update table1 SET col1="Manager" WHERE col2="Noukas";

//submit the query for execution by the DB
ResultSet resultSet=statement.executeQuery(query);

//submit an update query to the DB
int updatesPerformed=statement.executeUpdate(updateQuery);

//set a timeout interval
statement.setQueryTimeout(10);
ResultSet timedResultSet=statement.executeQuery(query);
```

Βήμα 6ο

- Επεξεργασία αποτελεσμάτων

```
//process the results

while (resultSet.next()) {
     out.println("Column1:"+resultSet.getString(1)+
          ",Column2:"+resultSet.getString(2)+
               ",Column3:"+resultSet.getString(3));
}
```

Βήμα 7ο

- Κλείσιμο της σύνδεσης στη Βάση

```
connection.close();
```

- Επειδή το κόστος του να ανοίξεις σύνδεση με τη Βάση (βήμα 3) είναι μεγάλο, το παραπάνω βήμα θα πρέπει να αποφεύγεται και να εκτελείται μόνο όταν τελειώσουν όλες οι συναλλαγές με τη βάση

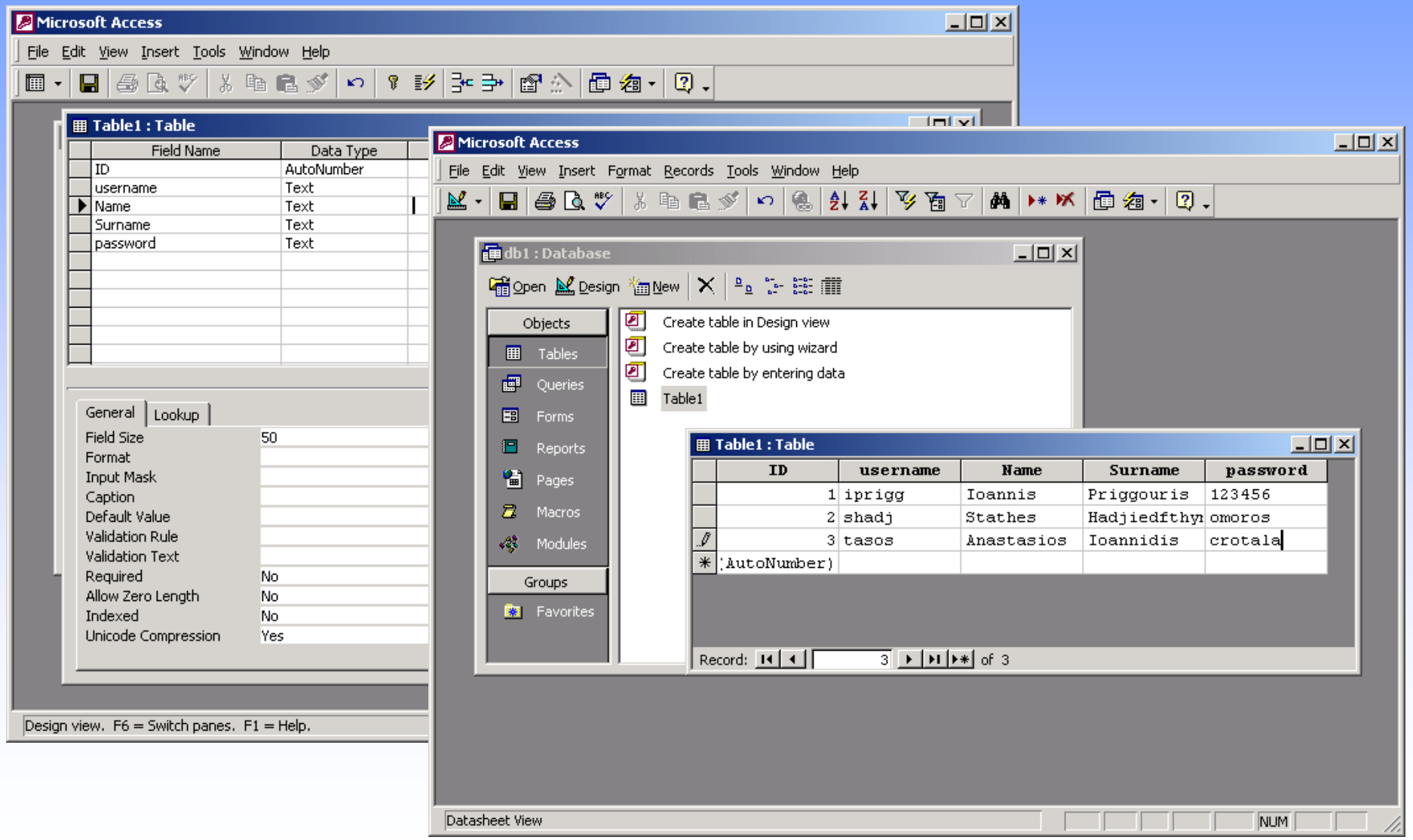
Ολοκληρωμένο παράδειγμα χρήσης (1/2)

```
<%@ page import="java.sql.*" %>

<html>
     <head> <title>Demonstrating a JDBC query</title> </head>

     <body>
     <center>
     <h2> Query Results </h2>
     <p>
<%
     //loading the jdbc:odbc driver
     try {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     } catch (ClassNotFoundException error) {
          System.err.println("Error"+error);
          out.println("Error:Unable to load jdbc:odbc Driver");
     }

     //Define the connection URL
     String dbName="testDB";
     String MSAccessURL="jdbc:odbc:"+dbName;

     //creating the connection to the DB
     Connection connection=DriverManager.getConnection(MSAccessURL,"","");

     //creating the statement object
     Statement statement = connection.createStatement();

     //performing a test query
     String query="SELECT * FROM User";
     ResultSet resultSet=statement.executeQuery(query);
%>
```

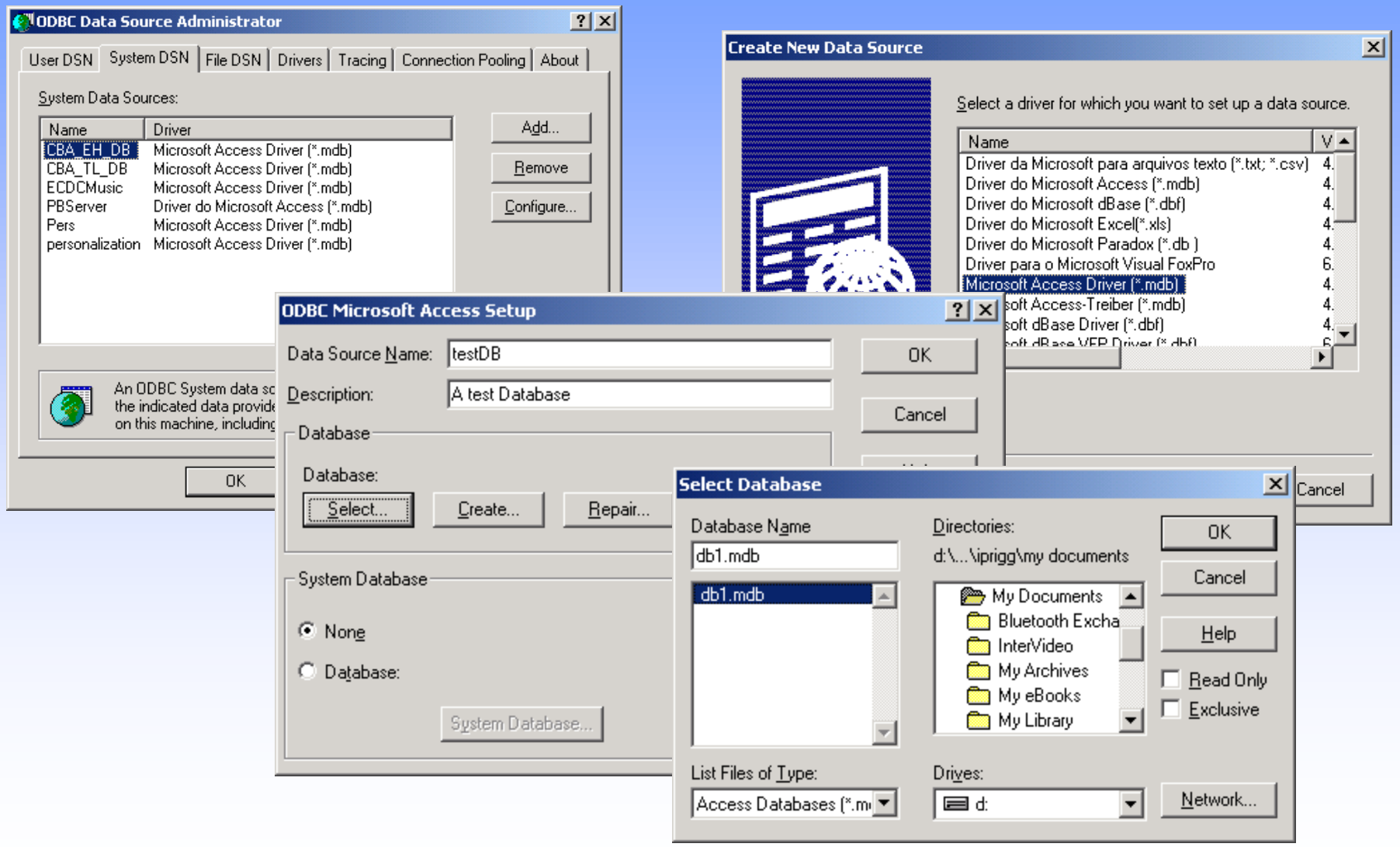
Ολοκληρωμένο παράδειγμα χρήσης (2/2)

```
<table border="2">
    <tr bgcolor="cyan">
        <td>ID</td>
        <td>UserName</td>
        <td>Name</td>
        <td>Surname</td>
        <td>Password</td>
    </tr>
<%
    //print the results of the query in table format
    while (resultSet.next()) {
%>
    <tr>
        <td><%=resultSet.getInt("ID")%></td>
        <td><%=resultSet.getString("username")%></td>
        <td><%out.println(resultSet.getString("Name"));%></td>
        <td><%out.println(resultSet.getString("Surname"));%></td>
        <td><%=resultSet.getString("Password")%></td>
    </tr>
<%
    }
%>
    </table>
    <body>
</html>
```

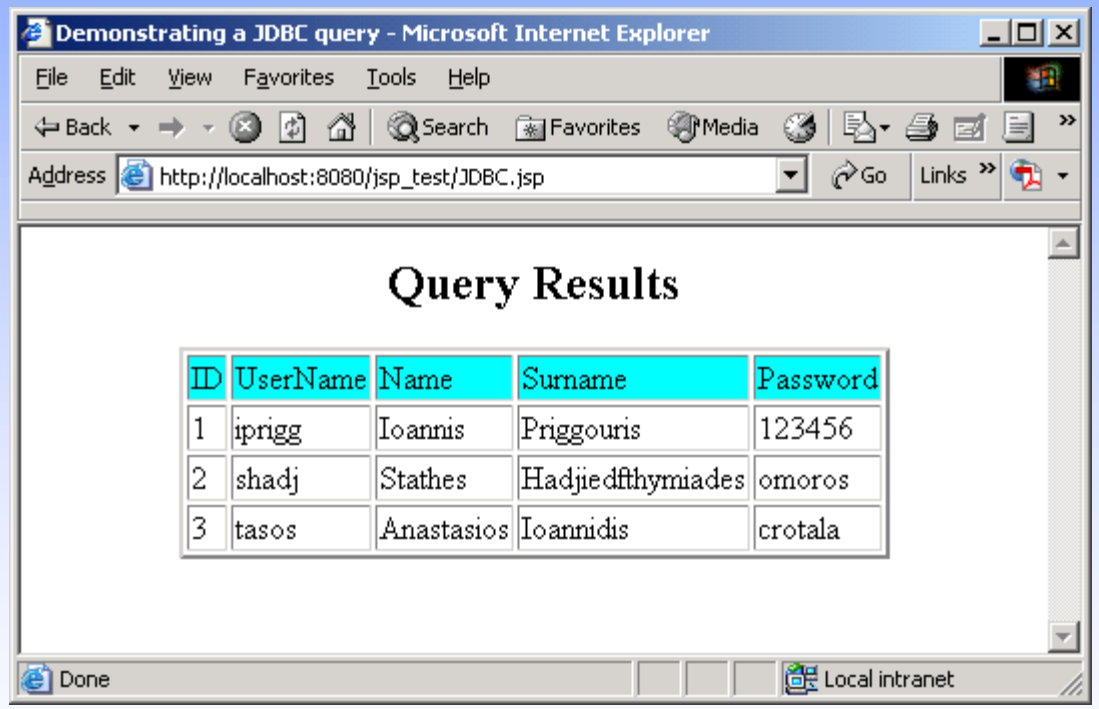
Creating a Microsoft Access DataBase



Configuring Microsoft Access ODBC DataSource



Running the example



Interface Connection

- Περιγραφή
  - Ανοίγει μια σύνδεση (session) με τη βάση
  - Ανήκει στο java.sql package
- Βασικές μέθοδοι:
  - **Statement createStatement()**
    - Δημιουργεί ένα αντικείμενο **Statement** για την διενέργεια queries προς τη βάση
  - **PreparedStatement prepareStatement(String sql)**
    - Δημιουργεί ένα αντικείμενο **PreparedStatement** για την διενέργεια pre-compiled queries προς τη βάση
  - **boolean getAutoCommit() / setAutoCommit(boolean autoCommit)**
    - Επιστρέφει/Ενεργοποιεί την αυτόματη διαδικασία commit() για την βάση. Εάν υπάρχουν δοσοληψίες που εκρεμούν τις κάνει πρώτα commit.
  - **commit()**
    - Κάνει μόνιμες όλες τις αλλαγές που πραγματοποιήθηκαν από τη τρέχουσα σύνδεση ως τώρα
  - **rollback()**
    - Ακυρώνει όλες τις αλλαγές που πραγματοποιήθηκαν από το τρέχουσα σύνδεση (από το τελευταίο commit() και μετά)

Εισαγωγή στην SQL

- Βασικές εντολές
  - CREATE TABLE
  - ALTER
  - DROP
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - ROLLBACK
  - COMMIT

Παραδείγματα εντολών SQL (1/4)

- **Δημιουργία Δομών**
  - CREATE TABLE
    - Δημιουργεί ένα καινούργιο πίνακα
    - Σύνταξη: **CREATE TABLE tableName (column_element [, column_element] …)**
    - Παράδειγμα: **CREATE Table User (Id NUMBER NOT NULL PRIMARY KEY, Name CHAR(30), Surname CHAR(30), Username CHAR(10))**
  - ALTER TABLE
    - Αλλάζει τη δομή ενός υπάρχοντος πίνακα
    - Σύνταξη: **ALTER TABLE tableName [ADD (column_element [, column_element] …)] [MODIFY (column_element [, column_element]…)]**
    - Παράδειγμα: **ALTER Table User ADD (Password CHAR(10))**
  - DROP TABLE
    - Καταστρέφει έναν πίνακα με όλες τις εγγραφές του
    - Σύνταξη: **DROP TABLE tableName**
    - Παράδειγμα: **DROP TABLE User**

Παραδείγματα εντολών SQL (2/4)

- **Αναζήτηση**
  - SELECT
    - Επιστρέφει έναν πίνακα από εγγραφές που συμφωνούν με τις παραμέτρους που ορίστηκαν
    - Σύνταξη: :**SELECT [ALL|DISTINCT] { * | tableName.* | expr [, { tableName.* | expr}]… FROM tableName [t_alias] [, tableName [t_alias]]…[WHERE condition] …**
    - Παράδειγμα: **SELECT DISTINCT * FROM User u WHERE u.Name='Ioannis'**

Παραδείγματα εντολών SQL (3/4)

- **Εισαγωγή και Τροποποίηση Εγγραφών**
  - INSERT
    - Εισάγει μια νέα εγγραφή στην βάση
    - Σύνταξη: **INSERT INTO tableName [(column [, column] …)] {VALUES (value [,value]…) | query }**
    - Παράδειγμα: **INSERT INTO User (Id, Name, Surname, Username) VALUES (1, 'ioannis', 'Priggouris', 'iprigg')**
  - UPDATE
    - Ενημερώνει μια υπάρχουσα εγγραφή με νέα στοιχεία
    - Σύνταξη: **UPDATE tableName SET column=expr [,column=expr] … [WHERE condition]**
    - Παράδειγμα: **UPDATE User SET Password='123456' WHERE Username='iprigg'**
  - DELETE
    - Διαγράφει μια εγγραφή από την Βάση
    - Σύνταξη:**DELETE [FROM] tableName [WHERE condition]**
    - Παράδειγμα: **DELETE FROM User WHERE username='iprigg'**

Παραδείγματα εντολών SQL (4/4)

- **Άλλες Εντολές**
  - ROLLBACK
    - Ακυρώνει όλες τις εργασίες που πραγματοποιήθηκαν στη τρέχουσα δοσοληψία
    - Σύνταξη: **ROLLBACK [WORK] [TO [SAVEPOINT] savepoint]**
    - Παράδειγμα: **ROLLBACK**
  - COMMIT
    - Κάνει μόνιμες όλες τις αλλαγές που πραγματοποιήθηκαν στην βάση κατά την διάρκεια της τελευταίας δοσοληψίας
    - Σύνταξη:**COMMIT [WORK]**
    - Παράδειγμα: **COMMIT**

Interface Statement

- Περιγραφή
  - Χρησιμοποιείται για την υποβολή ερωτήσεων προς τη Βάση
  - Ανήκει στο java.sql package
- Βασικές μέθοδοι:
  - **boolean execute(String sql)**
    - Υποβολή οποιοδήποτε ερώτησης
  - **ResultSet executeQuery(String sql)**
    - Υποβολή ερωτήσεων που αφορούν SELECT
  - **int executeUpdate(String sql)**
    - Υποβολή μόνο ερωτήσεων που τροποποιούν εγγραφές (INSERT, UPDATE, DELETE) και δομές (CREATE TABLE, ALTER TABLE, DROP TABLE)
  - **ResultSet getResultSet()**
    - Λήψη αποτελεσμάτων που προέκυψαν από την εκτέλεση μιας **execute()**
  - **Connection getConnection():**
    - Επιστρέφει το connection object που είναι συνδεδεμένο με το συγκεκριμένο Statement;

Interface PreparedStatement

- Περιγραφή
  - Χρησιμοποιείται για την υποβολή pre-compiled ερωτήσεων προς τη Βάση
  - Ανήκει στο java.sql package
- Βασικές μέθοδοι:
  - Κληρονομεί όλες τις μεθόδους του Statement
- Πλεονεκτήματα:
  - Μεγαλύτερη αποδοτικότητα για πολλαπλές ερωτήσεις του ίδιου τύπου
  - Το query αποθηκεύεται σε compiled μορφή στη βάση
  - Σε κάθε νέα χρήση οι ορισμένες παράμετροι αντικαθίστανται από καινούργιες με την χρήση των setXxx μεθόδων
  - Όλες οι μετατροπές τύπων γίνονται από την ίδια την βάση

Statement Vs PreparedStatement

```
//example of a standard statement
String sqlQuery=”INSERT INTO User VALUES ( 10, ‘Ioannis’, ‘Priggouris’, ‘iprigg’ )”;
Statement statement=connection.createStatement();
statement.executeUpdate(sqlQuery);
```

```
//example of a prepared Statement
connection.prepareStatement(“INSERT INTO User VALUES ( ?, ?, ?, ? )”);
statement.setInt(1,10);
statement.setString(2, “ioannis”);
statement.setString(3, “Priggouris”);
statement.setString(4, “iprigg”);
statement.executeUpdate():
```

Interface ResultSet (1/4)

- Περιγραφή
  - Ανήκει στο πακέτο java.sql
  - Ένα αντικείμενο τύπου ResultSet περιέχει τα αποτελέσματα ενός διενεργηθέντος SQL query
    - Επιστρέφεται ως αποτέλεσμα της κλήσης της ακόλουθης συνάρτησης: **statement.executeQuery(sqlString);**
    - Αναπαριστά έναν πίνακα από γραμμές και στήλες
    - Για την λήψη της 1ης γραμμής πρέπει να κληθεί η μέθοδος: **next()**
    - Για την λήψη κάθε μιας εκ των υπολοίπων γραμμών καλείται επίσης η **next()**
    - Μετακίνηση προς τα πίσω (για την λήψη προηγούμενης γραμμής) είναι εφικτή (JDBC 2.0 +)

Interface ResultSet (2/4)

- Χρήσιμες μέθοδοι
  - close()
    - Αποδεσμεύει το JDBC connection και τους πόρους της βάσης
    - Ένα ResultSet κλείνει αυτόματα κάθε φορά που το αντίστοιχο αντικείμενο Statement εκτελεί ένα νέο query.
  - ResultSetMetaData getMataDataObject()
    - Επιστρέφει ένα αντικείμενο ResultSetMetaData που περιέχει πληροφορίες για τις στήλες του ResultSet
  - boolean next()
    - Μετακίνηση στην επόμενη γραμμή του πίνακα του ResultSet:
      - Επιστρέφει **true** αν βρεθεί επόμενη γραμμή, αλλιώς **false**
      - Η πρώτη κλήση θέτει τον δείκτη στην πρώτη γραμμή (εάν υπάρχει)

Interface ResultSet (3/4)

- Χρήσιμες μέθοδοι
  - int findColumn(String columnName)
    - Επιστρέφει την αντίστοιχη ακέραια τιμή (δείκτη) που αντιστοιχεί στο όνομα της στήλης
    - Παρατήρηση: δεν υπάρχει 1:1 αντιστοιχία ανάμεσα στις στήλες του ResultSet και αυτές του πίνακα
  - Xxx getXxx(int index)
  - Xxx getXxx(String columnName)
    - Επιστρέφει την τιμή της στήλης που υποδεικνύει το όρισμα ως τύπο **Xxx** της java
    - Επιστρέφει **0** ή **NULL** όταν η τιμή είναι τύπου **SQL NULL**
    - Κυριότεροι Αποδεκτοί **Xxx** τύποι:

double	byte	int	long	Date	String	Blob
float	short	long	Time	TimeStamp		Object

  - wasNULL
    - Ελέγχει αν η τελευταία **getXxx** κλήση ήταν επέστρεψε **SQL NULL**

Interface ResultSet (4/4)

- Χρήσιμες μέθοδοι
  - first()
    - Μετακινεί τον δείκτη στη πρώτη γραμμή
  - last()
    - Μετακινεί τον δείκτη στη τελευταία γραμμή
  - isFirst()
    - Επιστρέφει **true** εάν η τρέχουσα γραμμή είναι η πρώτη
  - isLast()
    - Επιστρέφει **true** εάν η τρέχουσα γραμμή είναι η τελευταία
  - previous()
    - Μετακινεί τον δείκτη στη προηγούμενη γραμμή
  - Statement getStatement()
    - Επιστρέφει το Statement object που δημιούργησε το συγκεκριμένο ResultSet

Άσκηση 1η

- Να φτιαχτεί web εφαρμογή:
  - η οποία να διαβάζει μέσω μια φόρμας τα ακόλουθα στοιχεία και να τα τυπώνει στην οθόνη:
    - Name (String)
    - Surname (String)
    - Telephone (Number)
    - Username (String)
    - Password (String)

Άσκηση 2η

- Να τροποποιηθεί η εφαρμογή της άσκησης 1 ώστε:
  - Αντί απλά να τυπώνει τα στοιχεία στην οθόνη να τα καταχωρεί και σε κατάλληλη βάση δεδομένων
  - Εάν η καταχώρηση είναι επιτυχής να τυπώνει μια επιβεβαίωση για τα στοιχεία που καταχωρήθηκαν
  - Εάν η καταχώρηση δεν είναι επιτυχής να τυπώνει κατάλληλο μήνυμα ανεπιτυχούς καταχώρησης (π.χ. τον λόγο της αποτυχίας)
  - Να φτιαχτούν 2 εκδόσεις του προγράμματος:
    - Με χρήση απλού Statement
    - Με χρήση PreparedStatement

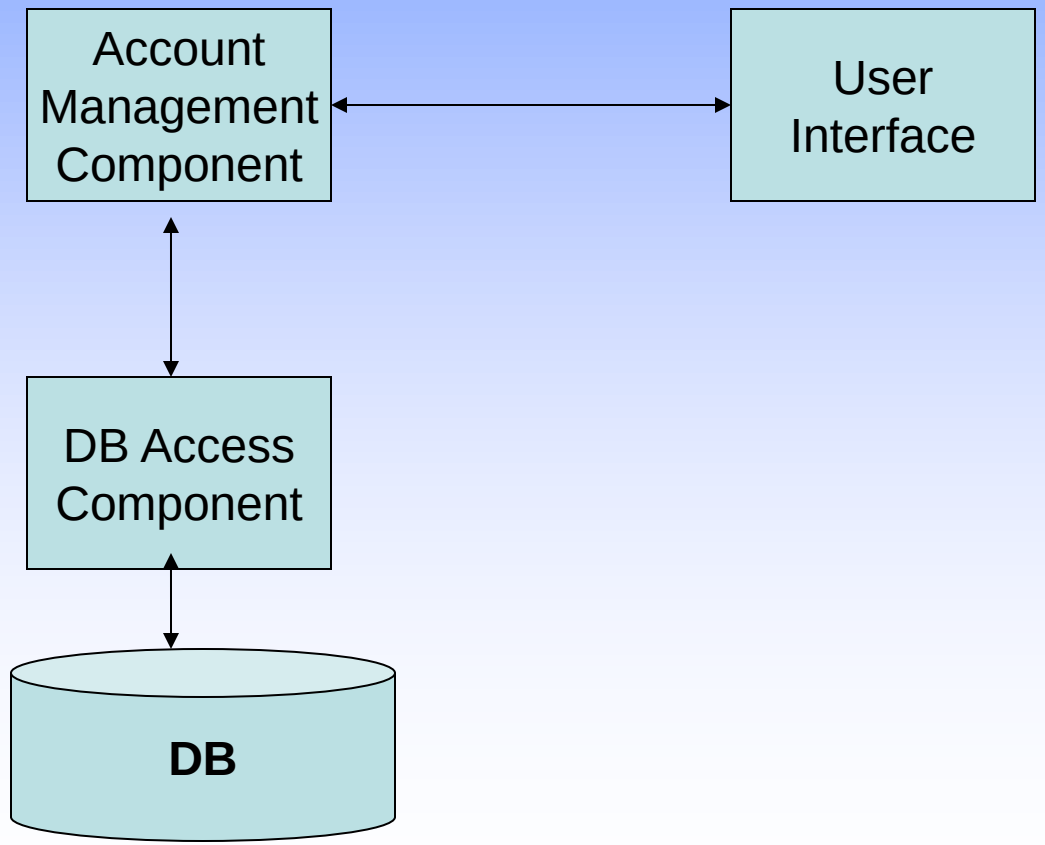
Άσκηση 3η

- Να τροποποιηθεί η άσκηση 2 ώστε:
  - Να δίνει την δυνατότητα στον χρήστη να αναιρέσει τις καταχωρήσεις που έκανε εφόσον το επιθυμεί
  - Να επιβεβαιώσει τις καταχωρήσεις του
  - Να κλείνει την σύνδεση με τη βάση.

Ανάπτυξη Διαδικτυακών Τόπων

JSP Components

Component-based μοντέλο



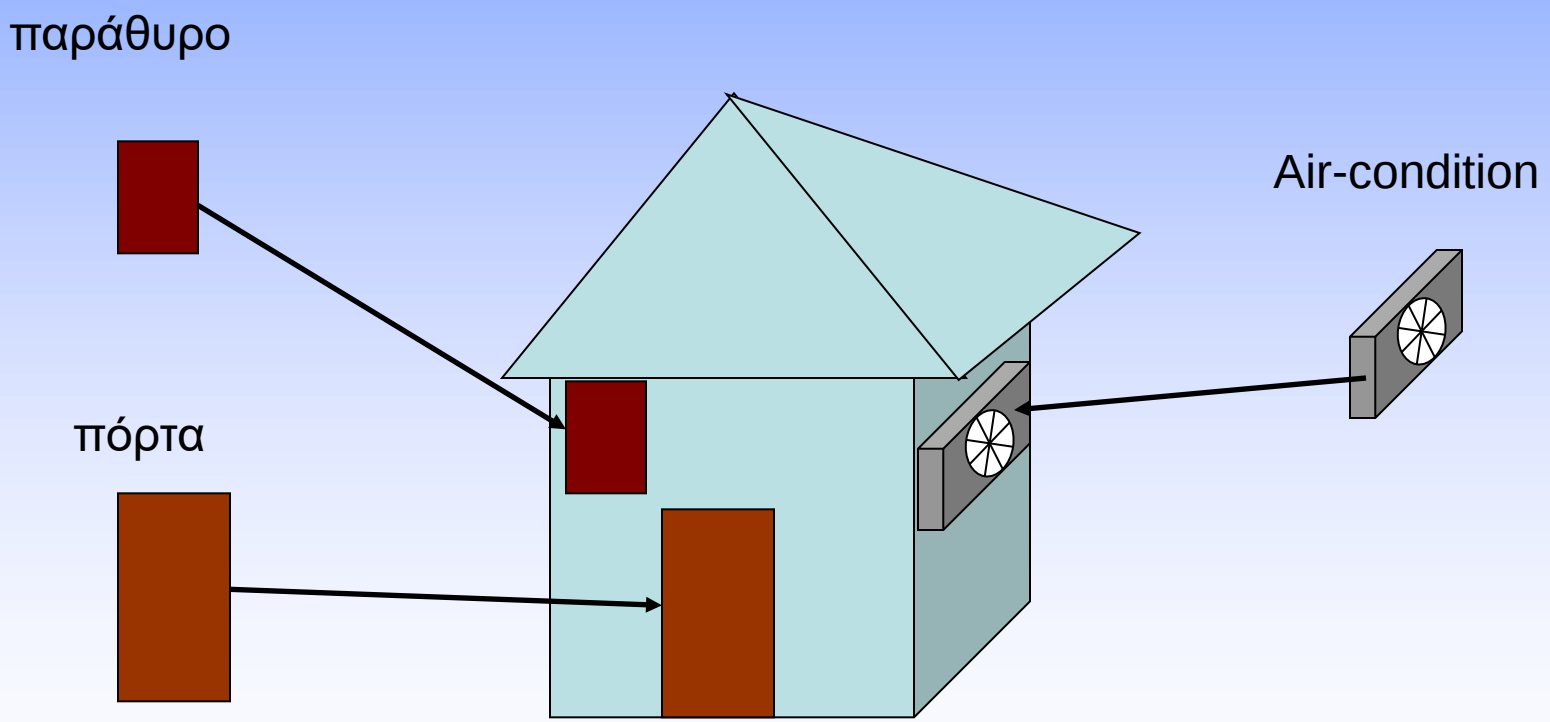
Components

- Τι είναι το component?
  - Αυτόνομο κομμάτι software με συγκεκριμένη λειτουργικότητα:
    - Επεξεργασία δεδομένων
    - Αποθήκευση συγκεκριμένου τύπου πληροφορίας
  - Κύριο γνώρισμα:
    - Δυνατότητα επαναχρησιμοποίησης (reusability)

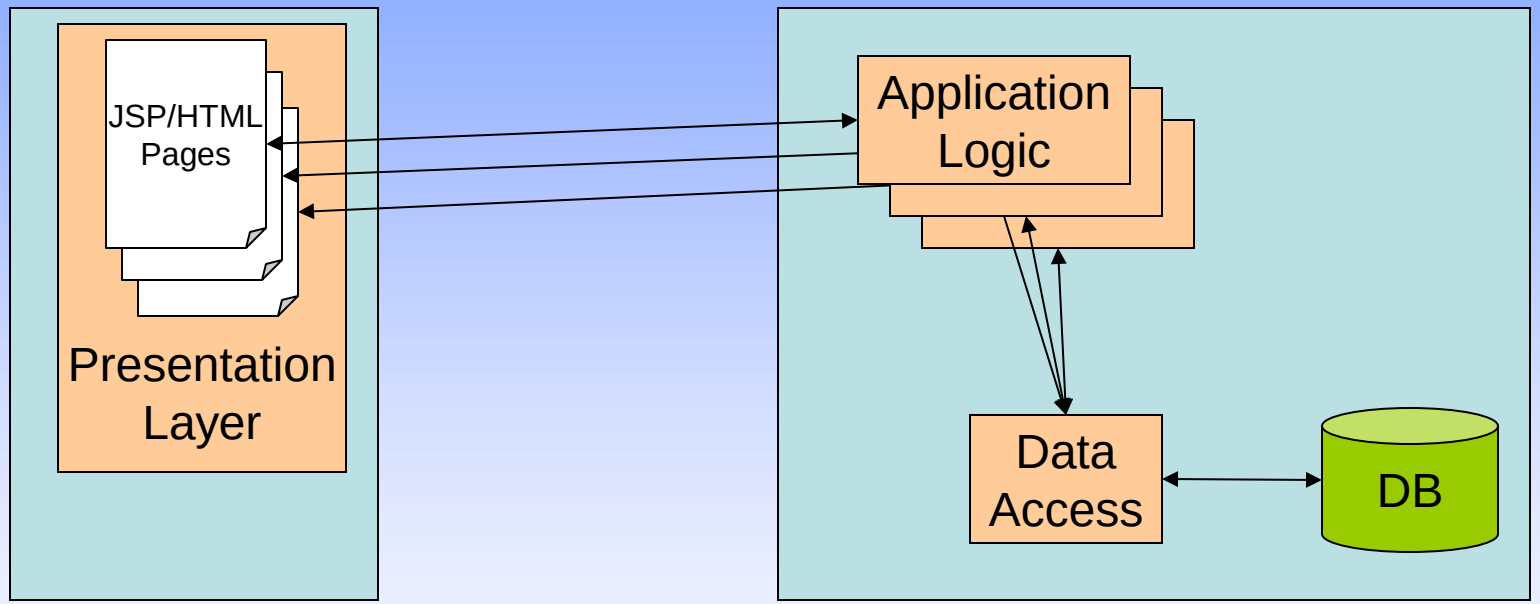
Component-Based Architectures

- Πλεονεκτήματα αρχιτεκτονικών:
  - Αυστηρή διάκριση λειτουργιών
  - Μείωση πολυπλοκότητας
  - Επαναχρησιμοποιησημότητα
  - Γρήγορη ανάπτυξη μεγάλων εφαρμογών

Παράδειγμα από την καθημερινή ζωή



Εφαρμογή στην ανάπτυξη Web εφαρμογών



Web designer's domain

Developer's domain

JavaBeans (1/5)

- Βασικά Χαρακτηριστικά:
  - Συμπεριφορά (behavior)
  - Κατάσταση (State)
- Χρήση σε web εφαρμογές:
  - Μέσω ειδικών tags
- Πλεονεκτήματα:
  - Ανάπτυξη μιας JSP εφαρμογής σχεδόν χωρίς καθόλου συγγραφή κώδικα Java (εντός της σελίδος)

JavaBeans (2/5)

- Βασικές έννοιες:
  - Bean container
  - Bean properties
    - Trigger & Linked properties
    - Indexed properties
  - Property Data types
    - Text
    - Number

JavaBeans (3/5)

- Τύποι JavaBeans:
  - Visual Components Beans
  - Data Beans
  - Service (Worker) Beans

JavaBeans (4/5)

- Μια κλάση της Java με συγκεκριμένες προδιαγραφές:
  - Κάθε property απεικονίζεται σε ένα member field
  - getter και setter μεθόδους για κάθε member field που περιέχει
  - Έναν default constructor

JavaBeans (5/5)

```
public class UserBean {

      String name;
      String surname;
      String username;
      String password;

      public User() {
            name="";
            surname="";
            username="";
            password="";
      }

      public void setName(String n) { name=n; }

      public String getName() { return name; }

      public void setSurname(String s) { surname=s; }

      public String getSurname() { return surname; }

      public void setUsername(String u) { username=u; }

      public String getUsername() { return username; }

      public void setPassword(String p) { password=p; }

      public String getPassword() { return password; }
}
```

<jsp: useBean …/> tag (1/2)

- Πραγματοποιεί την πρόσβαση σε ένα JavaBean

```
<jsp:useBean
        id="beanInstanceName"
        scope="page | request | session | application"
        {
            class="package.class" |
            type="package.class" |
            class="package.class" type="package.class" |
            beanName="{package.class | <%= expression %>}"
type="package.class"
        }
        {
            /> |
            > other elements </jsp:useBean>
        }
```

<jsp: useBean …/> tag (2/2)

- Παραδείγματα χρήσης

```
<jsp:useBean id="user" class="UserBean"/>


<jsp:useBean id="user" class="UserBean">
     initialization code
</jsp:useBean>


<jsp:useBean id="user" class="UserBean" scope="session">
     initialization code
</jsp:useBean>
```

JavaBeans Properties (1/2)

- Πρόσβαση μέσω ειδικών tags

```
<jsp:getProperty name="beanInstanceName" property="propertyName" />

<jsp:setProperty
        name="beanInstanceName"
        {
            property= "*"    |
            property="propertyName" [ param="parameterName" ]    |
            property="propertyName" value="{string | <%= expression %>}"
        }
/>
```

JavaBeans Properties (2/2)

- Παραδείγματα

```
<jsp:getProperty name="user" property="name"/>

<jsp:setProperty name="user" property="*"/>

<jsp:setProperty name="user" property="name"/>

<jsp:setProperty name="user" property="name" value="Ioannis"/>

<jsp:setProperty name="calculator" property="sum" value="<%=15*2%>"/>
```

Άσκηση 4η

- Να υλοποιηθεί η Άσκηση 1 με την χρήση components

Άσκηση 4η

```
<%@ page import="java.sql.*" %>

<html>
     <head>
          <title>Registering User</title>
     </head>
     <body>
     <center>
<jsp:useBean name="user" class="UserBean">
     <jsp:setProperty name="user" property="*"/>
</jsp:useBean>

     <table border="2">
          <tr bgcolor="cyan">
               <td>Name</td>
               <td>Surname</td>
               <td>Phone</td>
               <td>UserName</td>
               <td>Password</td>
          </tr>
          <tr>
               <td><jsp:getProperty name="user" property="name"/></td>
               <td><jsp:getProperty name="user" property="surname"/></td>
               <td><jsp:getProperty name="user" property="phone"/></td>
               <td><jsp:getProperty name="user" property="username"/></td>
               <td><jsp:getProperty name="user" property="password"/></td>
          </tr>
     </table>
     <p>
     <body>
</html>
```

Άσκηση 5η

- Να υλοποιηθεί η Άσκηση 2 με την χρήση components