Εισαγωγή στην ASP.NET 4.0

Σημειώσεις Σεμιναρίου

Επιμέλεια: Βασίλης Κόλιας

Ενότητα 6

- **ADO.NET**
- **Data Controls**

1.0.0

Πίνακας Περιεχομένων

11. ASP.NET και Βάσεις Δεδομένων

Στο προηγούμενο κεφάλαιο, κάναμε μια εισαγωγή στις βάσεις δεδομένων, είδαμε πως δημιουργούμε μια βάση δεδομένων και πως εισάγουμε, διαβάζουμε, ενημερώνουμε και διαγράφουμε εγγραφές από αυτή. Σε αυτή την ενότητα θα δούμε πώς πραγματοποιούμε τις ίδιες διαδικασίες μέσω όμως των μηχανισμών που παρέχει η ASP.NET για το σκοπό αυτό, δηλαδή μέσω του ADO.NET ή μέσω ειδικών controls, των data controls που είναι ειδικά φτιαγμένα για την αυτοματοποίηση των παραπάνω διαδικασιών.

11.1. ADO.NET

Το ADO.NET (ActiveX Data Objects) είναι ένα σύνολο από components που χρησιμοποιούμε για να αποκτήσουμε πρόσβαση σε δεδομένα. Είναι μέρος του base class library του .NET και χρησιμοποιείται κυρίως για την επικοινωνία, τη λήψη και την επεξεργασία δεδομένων από συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων, ενώ μπορεί να χρησιμοποιηθεί και για μη σχεσιακά δεδομένα.

Το ADO.NET βασίζεται στη λειτουργικότητα ενός μικρού συνόλου κλάσεων. Μπορούμε να χωρίσουμε αυτές τις κλάσεις σε δυο ομάδες: αυτές που χρησιμοποιούνται ως containers δεδομένων δηλαδή για να φιλοξενούν και να διαχειρίζονται δεδομένα (όπως την DataSet, την DataTable, την DataRow και τη DataRelation) και αυτές που χρησιμοποιούνται για να συνδεθούν σε μια συγκεκριμένη πηγή δεδομένων (όπως την Connection, την Command και την DataReader).

Οι κλάσεις που συμπεριφέρονται σαν containers δεδομένων είναι γενικού σκοπού. Ανεξάρτητα της πηγής δεδομένων που χρησιμοποιούμε, μόλις εξάγουμε τα δεδομένα, αυτά αποθηκεύονται χρησιμοποιώντας τον ίδιο container δεδομένων: την εξειδικευμένη κλάση DataSet. Μπορούμε να θεωρήσουμε ότι η DataSet παίζει τον ίδιο ρόλο που παίζει μια συλλογή ή ένας πίνακας – είναι ένα πακέτο δεδομένων. Η διαφορά είναι στο ότι η DataSet είναι προσαρμοσμένη για σχεσιακά δεδομένα, κάτι που σημαίνει ότι είναι σχεδιασμένη να αντιλαμβάνεται έννοιες όπως γραμμές και στήλες και σχέσεις πινάκων.

Η δεύτερη ομάδα κλάσεων εμφανίζεται με διάφορες μορφές. Κάθε σύνολο κλάσεων που χρησιμεύει για την αλληλεπίδραση δεδομένων καλείται ADO.NET data provider. Οι data providers είναι σχεδιασμένοι έτσι ώστε καθένας να χρησιμοποιεί το βέλτιστο τρόπο για να επικοινωνήσει με μια πηγή δεδομένων. Για παράδειγμα ο data provider του SQL Server είναι σχεδιασμένος να δουλεύει με τον SQL Server. Εσωτερικά, χρησιμοποιεί το πρωτόκολλο TDS (tabular data stream) για την επικοινωνία, εξασφαλίζοντας έτσι την καλύτερη δυνατή επίδοση. Αν χρησιμοποιούμε Oracle μπορούμε να χρησιμοποιήσουμε έναν Oracle data provider και ούτω καθεξής.

11.1.1. Το μοντέλο Data Provider

Κάθε provider χρησιμοποιεί το δικό του πρόθεμα για την ονομασία των κλάσεων. Έτσι ο SQL Server provider περιλαμβάνει τις κλάσεις με ονόματα SqlConnection και SqlCommand, ενώ ο Oracle provider περιλαμβάνει κλάσεις με ονόματα όπως OracleConnection και OracleCommand. Εσωτερικά, αυτές οι κλάσεις δουλεύουν διαφορετικά, επειδή πρέπει να συνδεθούν σε διαφορετικές βάσεις δεδομένων, χρησιμοποιώντας διαφορετικά πρωτόκολλα χαμηλού επιπέδου. Εξωτερικά ωστόσο, αυτές οι κλάσεις μοιάζουν παρόμοιες και παρέχουν το ίδιο σύνολο βασικών μεθόδων, γιατί υλοποιούν τα ίδια interfaces. Αυτό σημαίνει ότι η εφαρμογή μας αποφεύγει την πολυπλοκότητα των διαφορετικών προτύπων και μπορεί να χρησιμοποιήσει τον SQL Server provider ακριβώς όπως θα χρησιμοποιούσε τον Oracle provider. Στην πραγματικότητα είναι δυνατόν να «μεταφράσουμε» ένα μπλοκ κώδικα που αλληλεπιδρά με μια βάση SQL Server με ένα μπλοκ Oracle-specific Κώδικα αλλάζοντας απλώς τα ονόματα των κλάσεων.

Σε αυτό το σεμινάριο θα χρησιμοποιήσουμε μόνο SQL Server και κατά συνέπεια τον SQL Server data provider. Οι κλάσεις που θα χρησιμοποιήσουμε εντάσσονται σε τρεις κατηγορίες:

Namespace	Σκοπός
System.Data.SqlClient	Περιέχει τις κλάσεις που χρησιμοποιούμε για να συνδεθούμε σε μια βάση δεδομένων SQL Server και να εκτελέσουμε εντολές (όπως η κλάση SqlConnection και SqlCommand).
System.Data.SqlTypes	Περιέχει δομές για τύπους δεδομένων του SQL Server όπως ο SqlMoney και SqlDateTime. Μπορούμε να χρησιμοποιήσουμε αυτούς τους τύπους για να δουλέψουμε με τύπους δεδομένων του SQL Server χωρίς να χρειάζεται να τους μετατρέψουμε σε κάποιον από τους ενσωματωμένους τύπους δεδομένων του .NET όπως τους System.Decimal και System.DateTime. Η χρήση αυτών των τύπων δεν απαιτείται, αλλά χρησιμεύει στην αποφυγή προβλημάτων που μπορεί να προκύψουν κατά τη μετατροπή, όπως η στρογγυλοποίηση.
System.Data	Περιέχει θεμελιώδεις κλάσεις με την κύρια λειτουργικότητα του ADO.NET. Αυτό το namespace περιλαμβάνει τις κλάσεις DataSet και DataRelation, που επιτρέπουν το χειρισμό δομημένων σχεσιακών δεδομένων. Αυτές οι κλάσεις είναι εντελώς ανεξάρτητες, οποιουδήποτε ειδικού τύπου βάσης ή τρόπου επικοινωνίας με αυτή.

11.1.2. Άμεση επικοινωνία με τη βάση

Ο πιο ευθύς τρόπος για να αλληλεπιδράσουμε με μια βάση δεδομένων είναι να προβούμε σε άμεση επικοινωνία με τη βάση. Όταν χρησιμοποιούμε την άμεση επικοινωνία, πρέπει να δημιουργήσουμε ένα ερώτημα SQL και να το εκτελέσουμε. Υπάρχει η δυνατότητα να χρησιμοποιήσουμε ερωτήματα για να λάβουμε, εισάγουμε, ενημερώσουμε και να διαγράψουμε δεδομένα.

Όταν υποβάλλουμε ερωτήματα για δεδομένα χρησιμοποιώντας την άμεση επικοινωνία, δεν κρατείται αντίγραφο των δεδομένων στη μνήμη του server. Αντίθετα εργαζόμαστε απευθείας με την ίδια τη βάση για ένα σύντομο χρονικό διάστημα, ενώ η σύνδεση σε αυτή είναι ακόμα ανοιχτή και κλείνουμε τη σύνδεση το συντομότερο δυνατό. Η μέθοδος αυτή είναι αρκετά διαφορετική από τον αποσυνδεδεμένο τρόπο πρόσβασης στη βάση, όπου κρατείται αντίγραφο των δεδομένων σε ένα αντικείμενο DataSet για να μπορούμε να δουλεύουμε με αυτό ακόμα και μετά το κλείσιμο της σύνδεσης στη βάση.

Το άμεσο μοντέλο ενδείκνυται για ASP.NET σελίδες που δεν χρειάζονται ένα αντίγραφο των δεδομένων τους στη μνήμη του server για μεγάλα χρονικά διαστήματα. Μια ASP.NET σελίδα φορτώνεται όταν ο χρήστης έχει αιτηθεί για αυτήν και τερματίζει μόλις η απάντηση έχει επιστραφεί στο χρήστη. Έτσι είναι ιδανική για περιπτώσεις που η σελίδα γνωρίζουμε εκ των προτέρων ότι έχει ελάχιστο χρόνο ζωής.

Η μεθοδολογία που ακολουθούμε για να υποβάλλουμε ερωτήματα με το άμεσο μοντέλο, αποτελείται από τα εξής βήματα:

1. Δημιουργούμε αντικείμενα των κλάσεων Connection, Command και DataReader.
2. Ανοίγουμε τη σύνδεση μέσω ενός αντικείμενου Connection.
3. Χρησιμοποιούμε τη DataReader για να λάβουμε δεδομένα από τη βάση και τα εμφανίζουμε σε ένα control στη web form.
4. Κλείνουμε τη σύνδεση.
5. Στέλνουμε τη σελίδα στο χρήστη. Σε αυτό το σημείο, όλα τα αντικείμενα του ADO.NET έχουν καταστραφεί και τα δεδομένα που βλέπει ο χρήστης έχουν αποσυνδεθεί με τα δεδομένα που υπάρχουν στη βάση.

Για να προσθέσουμε ή να ενημερώσουμε δεδομένα, ακολουθούμε τα εξής βήματα:

1. Δημιουργούμε αντικείμενα των κλάσεων Connection και Command.
2. Ανοίγουμε τη σύνδεση μέσω ενός αντικειμένου της κλάσης Connection.
3. Εκτελούμε το ερώτημα μέσω ενός αντικείμενου της κλάσης Command.
4. Κλείνουμε τη σύνδεση.

Πριν χρησιμοποιήσουμε οποιαδήποτε κλάση του ADO.NET πρέπει να σιγουρευτούμε ότι έχουμε εισάγει τα κατάλληλα namespaces. Όπως αναφέραμε, θα χρησιμοποιήσουμε τον SQL Server provider και κατά συνέπεια πρέπει να εισάγουμε τα εξής namespaces:

```
C#

using System.Data;
using System.Data.SqlClient;
```

```
VB

Imports System.Data
Imports System.Data.SqlClient
```

Κώδικας 11.1 – Τα βασικά namespaces του ADO.NET

Δημιουργία μιας σύνδεσης

Πριν αρχίσουμε να επεξεργαζόμαστε τα δεδομένα, πρέπει να πραγματοποιήσουμε μια σύνδεση στην πηγή δεδομένων. Γενικά, οι συνδέσεις είναι περιορισμένες μέχρι έναν αριθμό και αν υπερβούμε αυτόν τον αριθμό, οποιαδήποτε προσπάθεια πραγματοποίησης σύνδεσης θα αποτύχει. Για το λόγο αυτό πρέπει να προσπαθήσουμε να κρατήσουμε μια σύνδεση ανοιχτή για όσο το δυνατόν λιγότερο χρόνο γίνεται. Πρέπει επίσης να εσωκλείουμε τον κώδικα που επικοινωνεί με τη βάση μέσα σε try/catch blocks έτσι ώστε να δράσουμε ανάλογα κατά την έγερση κάποιας εξαίρεσης και να είμαστε σίγουροι ότι σε κάθε περίπτωση θα κλείσει η σύνδεση ακόμα και αν δεν μπορούμε να ολοκληρώσουμε τη διαδικασία.

Όταν δημιουργούμε ένα αντικείμενο, χρειάζεται να ορίσουμε μια τιμή για την ιδιότητα ConnectionString. Ένα ConnectionString είναι στην ουσία ένα απλό string το οποίο έχει συγκεκριμένη δομή και ορίζει όλη την πληροφορία που χρειάζεται η εφαρμογή για να εντοπίσει την πηγή δεδομένων, να επιλέξει τη βάση δεδομένων και να κάνει log in σε αυτή. Ο ορισμός του ConnectionString είναι αρκετά απλός. Ακολουθεί ένα παράδειγμα που χρησιμοποιεί ένα connection string για να συνδεθεί σε μια βάση στον SQL Server.

```
C#

SqlConnection myConnection = new SqlConnection();
myConnection.ConnectionString = "Data Source=localhost;" +
   "Initial Catalog=Pubs;Integrated Security=SSPI";
```

```
VB

Dim myConnection As SqlConnection = New SqlConnection()
myConnection.ConnectionString = "Data Source=localhost;" &
   "Initial Catalog=Pubs;Integrated Security=SSPI"
```

Κώδικας 11.2 - Ορισμός ενός Connection String

Αν χρησιμοποιούμε SQL Server Express, το connection string θα περιλαμβάνει και το όνομα ενός instance όπως φαίνεται παρακάτω:

```
C#

SqlConnection myConnection = new SqlConnection();
myConnection.ConnectionString = @"Data Source=localhost\SQLEXPRESS;" +
"Initial Catalog=Pubs;Integrated Security=SSPI";
```

```
VB

Dim myConnection As SqlConnection = New SqlConnection()
myConnection.ConnectionString = @"Data Source=localhost\SQLEXPRESS;" & _
"Initial Catalog=Pubs;Integrated Security=SSPI"
```

Κώδικας 11.3 – Εναλλακτικός ορισμός ενός Connection String

Πρέπει να σημειώσουμε ότι ο μεταγλωττιστής υποθέτει ότι κάθε backslash προσδιορίζει την έναρξη μιας ακολουθίας ειδικών χαρακτήρων. Οπότε πρέπει να προσέχουμε όταν χρησιμοποιούμε connection strings στον κώδικά μας. Υπάρχουν δυο τρόποι για να αντιμετωπιστεί αυτό το πρόβλημα. Αρχικά, μπορούμε να χρησιμοποιήσουμε δυο backslash αντί για ένα, όπως π.χ. στο «localhost\\SQLEXPRESS» ή μπορούμε να τοποθετήσουμε τον χαρακτήρα @ πριν το string όπως στο «@"localhost\SQLEXPRESS"». Ωστόσο αν ορίσουμε το connection string στο configuration file της εφαρμογής (web.config), δεν χρειάζεται κανένας από τους προηγούμενους τρόπους.

Το Connection String

Το connection string είναι μια σειρά από διακριτά πεδία το οποία διαχωριζόμενα από ελληνικά ερωτηματικά (;). Κάθε κομμάτι δεδομένων αποτελεί μια ιδιότητα του connection string. Η επόμενη λίστα περιγράφει μερικές από τις πιο συχνά χρησιμοποιούμενες ιδιότητες:

- Data Source: η ιδιότητα αυτή προσδιορίζει το όνομα του server όπου υπάρχει η βάση δεδομένων. Αν ο server είναι στον ίδιο υπολογιστή που φιλοξενεί το ASP.NET site, τότε αρκεί ο προσδιοριστής localhost. Η μόνη εξαίρεση είναι όταν χρησιμοποιούμε ένα named instance του SQL Server όπως στην περίπτωση του SQL Server Express, οπότε και πρέπει να χρησιμοποιήσουμε την πηγή δεδομένων localhost\SQLEXPRESS, γιατί το instance name είναι SQLEXPRESS. Μπορεί να το συναντήσουμε και ως .\SQLEXPRESS που είναι ισοδύναμο.
- Initial Catalog: Αυτό είναι το όνομα της βάσης με την οποία η σύνδεση θα πραγματοποιηθεί. Καλείται ως initial (αρχική) γιατί μπορούμε να την αλλάξουμε χρησιμοποιώντας τη μέθοδο ChangeDatabase() της κλάσης Connection.
- Integrated security: Προσδιορίζει αν θέλουμε να συνδεθούμε σε έναν SQL Server χρησιμοποιώντας έναν Windows user account που τρέχει τον κώδικα της σελίδας, δεδομένου ότι παρέχουμε μια τιμή του SSPI (Security Support Provider Interface). Εναλλακτικά, μπορούμε να παρέχουμε ένα user ID και έναν κωδικό που ορίζεται στη βάση για να κάνουμε αυθεντικοποίηση του SQL Server, παρόλο που αυτή η μέθοδος είναι λιγότερο ασφαλής.
- Connection Timeout: Το πεδίο αυτό καθορίζει πόσο θα περιμένει ο κώδικάς μας σε δευτερόλεπτα πριν επιστρέψει σφάλμα αποτυχίας σύνδεσης στη βάση. Η προεπιλεγμένη τιμή είναι 15 δευτερόλεπτα. Μπορούμε να ορίσουμε το 0 για να μην έχει όριο, αλλά αυτό είναι κακή ιδέα αφού ο κώδικας θα περιμένει επ' άπειρον ενώ περιμένει να συνδεθεί με τον server.

Υπάρχουν και άλλες λιγότερο σημαντικές ιδιότητες του connection string για τις οποίες μπορούμε να ανατρέξουμε στο documentation του Visual Studio. Θα τις αναζητήσουμε κάτω από την κλάση Connection (όπως SqlConnection ή την OleDbConnection) γιατί υπάρχουν μερικές διαφορές μεταξύ των διαφόρων κλάσεων Connection.

Συνήθως όλος ο κώδικας της βάσης θα χρησιμοποιεί το ίδιο connection string. Για το λόγο αυτό, έχει νόημα να αποθηκεύσουμε αυτό το connection string σε μια static μεταβλητή ή ακόμα καλύτερα σε ένα configuration file. Μπορούμε επίσης να δημιουργήσουμε ένα αντικείμενο Connection και περέχουμε το connection string σε αυτό άμεσα χρησιμοποιώντας τον κατάλληλο constructor:

```
C#

SqlConnection myConnection = new SqlConnection(connectionString);
```

```
VB

Dim myConnection As SqlConnection = New SqlConnection(connectionString)
```

Κώδικας 11.4 - Ορισμός ενός αντικειμένου σύνδεσης

Δεν είναι απαραίτητο να εισάγουμε μέσα στον κώδικα το connection string. Το κομμάτι <connectionStrings> του web.config είναι το κατάλληλο μέρος για να το κάνουμε. Για παράδειγμα:

```
web.config

<configuration>
   <connectionStrings>
      <add name="NorthWindDB" connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\NORTHWND.MDF;Integrated
Security=True;Connect Timeout=30;User Instance=True"/>
   </connectionStrings>
   ...
</configuration>
```

Κώδικας 11.5 - Ορισμός ενός Connection String στο αρχείο web.config

Αν χρησιμοποιήσουμε αυτό τον τρόπο θα μπορούμε να λαμβάνουμε το connection string με το όνομά του. Αρχικά κάνουμε import το namespace System.Web.Configuration και στον κώδικά μας γράφουμε κάτι σαν:

```
C#

string connectionString =
WebConfigurationManager.ConnectionStrings["NorthWindDB"].ConnectionString;
```

```
VB

Dim connectionString AS String =
WebConfigurationManager.ConnectionStrings("NorthwindDB").ConnectionString
```

Κώδικας 11.6 – Ανάκτηση της τιμής του Connection String από το web.config

Έτσι εξασφαλίζουμε ότι όλες οι σελίδες χρησιμοποιούν το ίδιο connection string. Στην περίπτωση αυτή είναι εύκολο να αλλάξουμε το connection string μιας εφαρμογής, χωρίς να χρειάζεται να πειράξουμε τον κώδικα σε πολλαπλές σελίδες.

Windows authentication

Στα προηγούμενα παραδείγματα χρησιμοποιούνταν η ενσωματωμένη αυθεντικοποίηση των Windows (integrated Windows authentication), η οποία είναι και η προεπιλεγμένη μέθοδος αυθεντικοποίησης στις νέες εγκαταστάσεις του SQL Server. Είναι επίσης δυνατή η χρήση της μεθόδου αυθεντικοποίησης του SQL Server. Σε αυτήν την περίπτωση θα πρέπει να τοποθετήσουμε εμείς ρητά τα credentials (username και password) στο connection string. Ωστόσο, η μέθοδος αυτή είναι προεπιλεγμένα απενεργοποιημένη σε παλιότερες εκδόσεις του SQL Server γιατί θεωρείται λιγότερο ασφαλής από τις υπόλοιπες.

Συνοψίζοντας για τις μεθόδους αυθεντικοποίησης:

- Με την αυθεντικοποίηση του SQL Server ο SQL Server διατηρεί τις δικές του πληροφορίες στη βάση. Χρησιμοποιεί αυτήν την πληροφορία για να καθορίσει αν επιτρέπεται να χρησιμοποιήσουμε συγκεκριμένα σημεία της βάσης.
- Με την ενσωματωμένη αυθεντικοποίηση των Windows ο SQL Server αυτόματα χρησιμοποιεί την πληροφορία του λογαριασμού του λειτουργικού συστήματος για τη διαδικασία της σύνδεσης. Μέσα στη βάση αποθηκεύει πληροφορία για το ποιά δικαιώματα μπορεί να έχει κάθε χρήστης. Για να δουλέψει αυτή η μέθοδος αυθεντικοποίησης ο χρήστης πρέπει να έχει το επίπεδο πρόσβασης για τη χρήση της βάσης. Αυτό δεν είναι πρόβλημα κατά την ανάπτυξη των εφαρμογών αφού το Visual Studio χρησιμοποιεί τον user account μας. Κατά την εγκατάσταση όμως της εφαρμογής μπορεί να έχουμε πρόβλημα αφού ο κώδικας της ASP.NET τρέχει από έναν λογαριασμό περιορισμένων δικαιωμάτων, οπότε θα πρέπει να του δώσουμε περισσότερα δικαιώματα.

Κάθε database server αποθηκεύει μια κύρια λίστα από όλες τις βάσεις δεδομένων που έχουμε εγκαταστήσει σε αυτόν. Αυτή η λίστα περιλαμβάνει το όνομα κάθε βάσης δεδο-

μένων και την τοποθεσία των αρχείων που περιέχουν τα δεδομένα. Όταν δημιουργούμε μια βάση, για παράδειγμα τρέχοντας ένα script ή χρησιμοποιώντας ένα εργαλείο διαχείρισης, η πληροφορία για τη βάση προστίθεται στην κύρια λίστα. Όταν συνδεόμαστε στη βάση, προσδιορίζουμε το όνομα της βάσης χρησιμοποιώντας την τιμή του αρχικού καταλόγου στο connection string.

Ο SQL Server Express έχει ένα χαρακτηριστικό που επιτρέπει να προσπεράσουμε την κύρια λίστα και να συνδεθούμε κατευθείαν σε κάθε αρχείο βάσης δεδομένων ακόμα και αν δεν είναι στην κύρια λίστα των βάσεων. Αυτό το χαρακτηριστικό καλείται user instances και δεν είναι διαθέσιμο στην πλήρη έκδοση του SQL Server. Είναι χρήσιμο π.χ. κατά τον κύκλο ανάπτυξης ενός site όπου συχνά, θέλουμε να έχουμε ένα μόνο αρχείο βάσης μέσα στο φάκελο του web site μας και στη συνέχεια να μοιραζόμαστε και να μετακινούμε αυτό το φάκελο.

Για να χρησιμοποιήσουμε αυτό το χαρακτηριστικό πρέπει να θέσουμε την τιμή User Instances σε true στο connection string και να παρέχουμε το όνομα της βάσης δεδομένων με την οποία θέλουμε να συνδεθούμε με την τιμή AttachDBFilename. Δεν χρειάζεται να παρέχουμε μια αρχική τιμή στο Initial Catalog. Ακολουθεί ένα παράδειγμα:

C#

```
myConnection.ConnectionString = @" Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\NORTHWND.MDF;Integrated
Security=True;Connect Timeout=30;User Instance=True ";
```

VB

```
myConnection.ConnectionString = @"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\NORTHWND.MDF;Integrated
Security=True;Connect Timeout=30;User Instance=True"
```

Κώδικας 11.7 - Άμεση ανάθεση τιμής Connection String

Υπάρχει άλλο ένα σημείο που πρέπει να προσέξουμε εδώ. Το όνομα του αρχείου ξεκινάει με |DataDirectory|. Αυτό δείχνει αυτόματα προς το φάκελο App_Data μέσα στο directory της web εφαρμογής. Έτσι δεν χρειάζεται να παρέχουμε ένα πλήρες path, που μπορεί να μην καταστεί τελικά μη έγκυρο αν μεταφέρουμε την εφαρμογή στον web server. Στο συγκεκριμένο παράδειγμα το ADO.NET θα κοιτάζει πάντα στο directory App_Data (λόγω χρήσης του προσδιοριστή |DataDirectory|) για ένα αρχείο με το όνομα Northwind.mdf.

Πραγματοποίηση της σύνδεσης

Μόλις έχουμε μια λειτουργική σύνδεση, είμαστε έτοιμοι να τη χρησιμοποιήσουμε. Πριν μπορέσουμε να πραγματοποιήσουμε οποιαδήποτε ενέργεια στη βάση, πρέπει να ανοίξουμε ρητά τη σύνδεση ως εξής:

C#

```
myConnection.Open();
```

VB

```
myConnection.Open()
```

Κώδικας 11.8 – Άνοιγμα σύνδεσης με τη βάση

Για να βεβαιωθούμε ότι έχουμε επιτυχώς συνδεθεί με τη βάση, μπορούμε να δοκιμάσουμε να εμφανίσουμε κάποιες βασικές πληροφορίες της σύνδεσης. Στο επόμενο παράδειγμα εμφανίζουμε τις πληροφορίες αυτές σε ένα Label control:

C#

```
string connectionString =
WebConfigurationManager.ConnectionStrings["NorthwindDB"].ConnectionString;
SqlConnection myConnection = new SqlConnection(connectionString);
try
{
   // Try to open the connection.
```

```
   myConnection.Open();
   lblInfo.Text = "<b>Server Version:</b> " + myConnection.ServerVersion;
   lblInfo.Text += "<br /><b>Connection Is:</b> " +
   myConnection.State.ToString();
}
catch (Exception err)
{
   // Handle an error by displaying the information.
   lblInfo.Text = "Error reading the database. ";
   lblInfo.Text += err.Message;
}
finally
{
   // Either way, make sure the connection is properly closed.
   // (Even if the connection wasn't opened successfully,
   // calling Close() won't cause an error.)
   myConnection.Close();
   lblInfo.Text += "<br /><b>Now Connection Is:</b> ";
   lblInfo.Text += myConnection.State.ToString();
}
```

```
VB

Dim connectionString As String = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString
        Dim myConnection As SqlConnection = New SqlConnection(connectionString)
        Try

            'Try to open the connection.
            myConnection.Open()
            lblInfo.Text = "<b>Server Version:</b> " & myConnec-
tion.ServerVersion
            lblInfo.Text &= "<br /><b>Connection Is:</b> " &
            myConnection.State.ToString()

        Catch err As Exception

            ' Handle an error by displaying the information.
            lblInfo.Text = "Error reading the database. "
            lblInfo.Text &= err.Message

        Finally

            'Either way, make sure the connection is properly closed.
            '(Even if the connection wasn't opened successfully,
            ' calling Close() won't cause an error.)
            myConnection.Close()
            lblInfo.Text &= "<br /><b>Now Connection Is:</b> "
            lblInfo.Text &= myConnection.State.ToString()
        End Try
```

Κώδικας 11.9 – Έλεγχος επιτυχούς σύνδεσης με τη βάση

Με το που καλείται η μέθοδος Open(), έχουμε μια ενεργή σύνδεση στη βάση. Μια από τις πιο βασικές αρχές εργασίας με βάσεις δεδομένων είναι να κρατάμε όσο το δυνατόν λιγότερο τη σύνδεση ανοιχτή για να εξασφαλίσουμε ότι το site είναι αποδοτικό. Το κλείσιμο μιας σύνδεσης είναι το ίδιο εύκολο με το άνοιγμά της:

```
C#

myConnection.Close();
```

```
VB

myConnection.Close()
```

Κώδικας 11.10 - Κλείσιμο σύνδεσης με τη βάση

11.1.3.Διάβασμα εγγραφών

Το αντικείμενο Connection φέρει μερικές βασικές ιδιότητες που παρέχουν πληροφορίες αποκλειστικά για τη σύνδεση με τη βάση δεδομένων. Για να λάβουμε όμως τα δεδομένα χρειάζονται μερικά συστατικά ακόμη:

- Ένα ερώτημα SQL που επιλέγει την πληροφορία που θέλουμε

- Ένα αντικείμενο Command που εκτελεί το ερώτημα SQL
- Ένα αντικείμενο DataReader ή DataSet για να προσπελάσουμε τα αποτελέσματα του ερωτήματος

Τα αντικείμενα Command αναπαριστούν ερωτήματα SQL. Για να χρησιμοποιήσουμε ένα Command, ορίζουμε το ίδιο το αντικείμενο, καθορίζουμε το ερώτημα SQL που θέλουμε να υποβάλουμε, προσδιορίζουμε μια διαθέσιμη σύνδεση και εκτελούμε την εντολή. Μια ορθή πρακτική είναι να ανοίγουμε τη σύνδεση ακριβώς πριν εκτελέσουμε την εντολή και να την κλείνουμε όταν έχει τελειώσει. Μπορούμε να χρησιμοποιήσουμε την εντολή ως εξής:

C#

```
SqlCommand myCommand = new SqlCommand();
myCommand.Connection = myConnection;
myCommand.CommandText = SELECT EmployeeID, LastName, FirstName, Title FROM Em-
ployees";
```

VB

```
Dim myCommand As SqlCommand = New SqlCommand()
myCommand.Connection = myConnection
myCommand.CommandText = "SELECT EmployeeID, LastName, FirstName, Title FROM Em-
ployees"
```

Κώδικας 11.11 – Ορισμός SQL εντολής

ή να χρησιμοποιήσουμε τον constructor για συντομία:

C#

```
SqlCommand myCommand = new SqlCommand(
"SELECT EmployeeID, LastName, FirstName, Title FROM Employees ", myConnection);
```

VB

```
Dim myCommand AS SqlCommand = new SqlCommand(
"SELECT EmployeeID, LastName, FirstName, Title FROM Employees ", myConnection)
```

Κώδικας 11.12 – Σύντομος ορισμός SQL εντολής

11.1.4.Η κλάση DataReader

Όταν έχουμε έτοιμη την εντολή, πρέπει να αποφασίσουμε πως θα τη χρησιμοποιήσουμε. Η πιο απλή προσέγγιση είναι χρησιμοποιώντας έναν DataReader, ο οποίος επιτρέπει τη γρήγορη λήψη των αποτελεσμάτων. Ο DataReader χρησιμοποιεί μια ενεργή σύνδεση σε βάση ενώ πρέπει να χρησιμοποιείται σύντομα και αμέσως μετά να κλείνει. Ο DataReader είναι επίσης πολύ απλός. Υποστηρίζει μια γρήγορη ανάγνωση προς μια κατεύθυνση των δεδομένων, η οποία γενικά είναι ό, τι χρειαζόμαστε όταν λαμβάνουμε δεδομένα από μια βάση. Επειδή είναι βελτιστοποιημένος, είναι αποδοτικότερος σε σύγκριση με το DataSet. Πρέπει να είναι πάντα η πρώτη μας επιλογή κατά την άμεση προσπέλαση δεδομένων.

Πριν τη χρήση ενός DataReader πρέπει να βεβαιωθούμε ότι έχουμε ανοίξει μια σύνδεση. Για να δημιουργήσουμε έναν DataReader χρησιμοποιούμε τη μέθοδο ExecuteReader() του αντικειμένου command, ως εξής:

C#

```
SqlDataReader myReader;
myReader = myCommand.ExecuteReader();
```

VB

```
Dim myReader As SqlDataReader
myReader = myCommand.ExecuteReader()
```

Κώδικας 11.13 – Δημιουργία αντικειμένου Reader

Αυτές οι δύο γραμμές ορίζουν μια μεταβλητή για τον DataReader και την αρχικοποιούν εκτελώντας την εντολή. Μόλις έχουμε τον reader, μπορούμε να λάβουμε μια γραμμή κάθε φορά χρησιμοποιώντας τη μέθοδο Read():

```
C#

myReader.Read();
```

```
VB

myReader.Read()
```

Κώδικας 11.14 – Ανάκτηση εγγραφής μέσω αντικειμένου Reader

Μπορούμε να προσπελάσουμε τις τιμές στην τρέχουσα γραμμή χρησιμοποιώντας τα αντίστοιχα ονόματα των πεδίων. Στο ακόλουθο παράδειγμα ο κώδικας θα εμφανίσει τα πεδία LastName και FirstName στη σελίδα για την τρέχουσα εγγραφή στο Reader:

```
C#

Response.Write(myReader["LastName"] + " " + myReader["FirstName"] + "<br />");
```

```
VB

Response.Write(myReader("LastName") & " " + myReader("FirstName") & "<br />")
```

Κώδικας 11.15 – Εκτύπωση πεδίων μιας εγγραφής στη σελίδα μέσω αντικειμένου Reader

Για να μεταφερθούμε στην επόμενη γραμμή, χρησιμοποιούμε τη μέθοδο Read() ξανά. Αν η μέθοδος αυτή επιστρέφει True, μια γραμμή πληροφορίας έχει ληφθεί επιτυχώς. Αν επιστρέφει False, τότε επιχειρήσαμε να διαβάσουμε πέρα από το τέλος των εγγραφών που επιστράφηκαν. Δεν υπάρχει τρόπος να κινηθούμε προς τα πίσω, δηλαδή σε προηγούμενες εγγραφές.

Μόλις τελειώσουμε με την ανάγνωση των αποτελεσμάτων, κλείνουμε τον DataReader αλλά και την Connection.

```
C#

myReader.Close();
myConnection.Close();
```

```
VB

myReader.Close()
myConnection.Close()
```

Κώδικας 11.16 – Κλείσιμο Reader και σύνδεσης

Ακολουθεί ένα παράδειγμα στο οποίο φαίνεται πως χρησιμοποιούμε τα όσα περιγράφτηκαν παραπάνω για να φτιάξουμε μια απλή εφαρμογή που λαμβάνει πληροφορίες για ένα συγκεκριμένο υπάλληλο από τον πίνακα Employees. Η ιδέα είναι να μπορούμε να επιλέξουμε ένα συγκεκριμένο υπάλληλο από ένα drop-down list box και άμεσα να εμφανίζονται οι πληροφορίες του στη σελίδα.

Το παρουσιαστικό της σελίδας είναι απλό.

```
ASP

<%@ Page Language="VB" AutoEventWireup="false" Code-
File="Example_DataReader_CompleteExample.aspx.vb" Inher-
its="Example DataReader CompleteExample" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
```

```
     <asp:DropDownList ID="ddlEmployees" runat="server" onselectedin-
dexchanged="ddlEmployees SelectedIndexChanged" AutoPostBack="true">
     </asp:DropDownList>
        <br />
     <asp:Label ID="lblInfo" runat="server"></asp:Label>

    </div>
    </form>
</body>
</html>
```

Το list box θα φορτώνεται κατά το event Page.Load. Η πληροφορία που εμφανίζει θα πρέπει να λαμβάνεται μόνο μια φορά και μάλιστα την πρώτη φορά που εμφανίζεται η σελίδα. Σε όλα τα υπόλοιπα postbacks η φόρτωση της λίστας θα πρέπει να αγνοείται. Η ιδιότητα IsPostBack του αντικειμένου Page δηλώνει αν αίτηση είναι ένα postback:

C#

```
protected void Page Load(Object sender, EventArgs e)
{
   if (!this.IsPostBack)
   {
            FillEmployeeList();
   }
}
```

VB

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
        If Not Page.IsPostBack Then
            FillEmployeeList()
        End If
End Sub
```

Κώδικας 11.17 – Γέμισμα του DropDownList μόνο κατά την πρώτη φόρτωση της σελίδας

Έπειτα χρησιμοποιούμε τα αντικείμενα Connection, Command και DataReader μέσα στην υλοποίηση της μεθόδου FillEmployeeList:

C#

```
private void FillEmployeeList()
    {
        ddlEmployees.Items.Clear();
        // Define the Select statement.
        // Three pieces of information are needed: the unique id
        // and the first and last name.
        SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;

        string selectSQL = "SELECT EmployeeID, LastName, FirstName FROM Employ-
ees";
        SqlCommand cmd = new SqlCommand(selectSQL, myConnection);
        SqlDataReader reader;
        // Try to open database and read information.
        try
        {
            myConnection.Open();
            reader = cmd.ExecuteReader();

            while (reader.Read())
            {
                ListItem newItem = new ListItem();
                newItem.Text = reader["LastName"] + ", " + reader["FirstName"];
                newItem.Value = reader["EmployeeID"].ToString();
                ddlEmployees.Items.Add(newItem);
            }
            reader.Close();
        }
        catch (Exception err)
        {
            lblInfo.Text = "Error reading list of names ";
            lblInfo.Text += err.Message;
```

```
        }
        finally
        {
            myConnection.Close();
        }
    }
```

```
Private Sub FillEmployeeList()
        ddlEmployees.Items.Clear()

        Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString

        Dim selectSQL As String = "SELECT EmployeeID, LastName, FirstName FROM
Employees"
        Dim cmd As SqlCommand = New SqlCommand(selectSQL, myConnection)
        Dim reader As SqlDataReader

        Try

            myConnection.Open()
            reader = cmd.ExecuteReader()
            While (reader.Read())
                Dim newItem As ListItem = New ListItem()
                newItem.Text = reader("LastName") & ", " & reader("FirstName")
                newItem.Value = reader("EmployeeID").ToString()
                ddlEmployees.Items.Add(newItem)
            End While

            reader.Close()
        Catch  err As Exception
            lblInfo.Text = "Error reading list of names "
            lblInfo.Text &= err.Message
        Finally
            myConnection.Close()
        End Try

End Sub
```

Κώδικας 11.18 – Μέθοδος για το γέμισμα του DropDownList με υπαλλήλους

Στο παραπάνω παράδειγμα ανοίγεται μια σύνδεση με τη βάση μέσα σε ένα μπλοκ διαχείρισης εξαιρέσεων (try block) έτσι ώστε η σελίδα μας να χειριστεί οποιαδήποτε σφάλματα προκύψουν παρέχοντας σχετικές πληροφορίες. Το μπλοκ finally, επειδή εκτελείται πάντα, εξασφαλίζει ότι θα κλείσει μια σύνδεση ακόμα και αν συμβεί κάποιο σφάλμα. Επομένως είναι το κατάλληλο σημείο για να εισάγουμε κώδικα και το κλείσιμο της σύνδεσης.

Ο κώδικας που διαβάζει τα δεδομένα χρησιμοποιεί μια δομή επανάληψης. Σε κάθε επανάληψη, καλείται η μέθοδος Read για να ληφθεί άλλη μια γραμμή δεδομένων. Όταν ο reader έχει διαβάσει όλα τα διαθέσιμα δεδομένα, η μέθοδος θα επιστρέψει false, η συνθήκη επανάληψης θα αποτιμηθεί σε false και η επανάληψη θα σταματήσει.

Το μοναδικό ID (η τιμή στο πεδίο EmployeeID) αποθηκεύεται στην ιδιότητα Value του DropDownList για χρήση αργότερα. Η τιμή αυτή είναι σημαντική για τη λήψη επιπρόσθετων πληροφοριών για τη συγκεκριμένη εγγραφή που επιλέχτηκε. Αν επιλέγαμε κάποια άλλη τιμή για την υποβολή ερωτημάτων, όπως το όνομα του υπαλλήλου θα έπρεπε να μεριμνήσουμε για τους συγγραφείς με το ίδιο όνομα.

Μόλις λάβουμε την επιλογή του χρήστη από το list box μπορούμε να λάβουμε πληροφορίες για τον υπάλληλο που επέλεξε υποβάλλοντας ένα ερώτημα στη βάση με το id που του αντιστοιχεί. Για να το καταστήσουμε δυνατό αυτό όμως, πρέπει να ενεργοποιήσουμε την ιδιότητα AutoPostBack του DropDownList έτσι ώστε να πυροδοτούνται τέτοιου είδους events αυτόματα.

C#

```
protected void ddlEmployees SelectedIndexChanged(object sender, EventArgs e)
    {

        string selectSQL;
        selectSQL = "SELECT * FROM Employees";
        selectSQL += " WHERE EmployeeID=" + ddlEmployees.SelectedItem.Value;
```

```
        // Define the ADO.NET objects.
        SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;
        SqlCommand cmd = new SqlCommand(selectSQL, myConnection);
        SqlDataReader reader;
        // Try to open database and read information.
        try
        {
            myConnection.Open();
            reader = cmd.ExecuteReader();
            reader.Read();
            // Build a string with the record information,
            // and display that in a label.
            System.Text.StringBuilder sb = new System.Text.StringBuilder();
            sb.Append("<b>");
            sb.Append(reader["LastName"]);
            sb.Append(", ");
            sb.Append(reader["FirstName"]);
            sb.Append("</b><br />");
            sb.Append("Title: ");
            sb.Append(reader["Title"]);
            sb.Append("<br />");
            sb.Append("Title of Courtesy: ");
            sb.Append(reader["TitleOfCourtesy"]);
            sb.Append("<br />");
            sb.Append("Date of Birth: ");
            sb.Append(reader["BirthDate"]);
            sb.Append("<br />");
            sb.Append("Hire Date: ");
            sb.Append(reader["HireDate"]);
            sb.Append("<br />");
            sb.Append("Address: ");
            sb.Append(reader["Address"]);
            sb.Append("<br />");
            sb.Append("City: ");
            sb.Append(reader["City"]);
            sb.Append("<br />");
            sb.Append("Region: ");
            sb.Append(reader["Region"]);
            sb.Append("<br />");
            sb.Append("PostalCode: ");
            sb.Append(reader["PostalCode"]);
            sb.Append("<br />");
            sb.Append("Country: ");
            sb.Append(reader["Country"]);
            sb.Append("<br />");


            lblInfo.Text = sb.ToString();
            reader.Close();
        }
        catch (Exception err)
        {
            lblInfo.Text = "Error getting Employee. ";
            lblInfo.Text += err.Message;
        }
        finally
        {
            myConnection.Close();
        }

    }
```

```
VB

Protected Sub ddlEmployees SelectedIndexChanged(ByVal sender As Object, ByVal e
As System.EventArgs) Handles ddlEmployees.SelectedIndexChanged

        Dim selectSQL As String
        selectSQL = "SELECT * FROM Employees"
        selectSQL &= " WHERE EmployeeID=" & ddlEmployees.SelectedItem.Value

        Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString
```

```
        Dim cmd As SqlCommand = New SqlCommand(selectSQL, myConnection)
        Dim reader As SqlDataReader

        Try
            myConnection.Open()
            reader = cmd.ExecuteReader()
            reader.Read()

            Dim sb As System.Text.StringBuilder = New Sys-
tem.Text.StringBuilder()
            sb.Append("<b>")
            sb.Append(reader("LastName"))
            sb.Append(", ")
            sb.Append(reader("FirstName"))
            sb.Append("</b><br />")
            sb.Append("Title: ")
            sb.Append(reader("Title"))
            sb.Append("<br />")
            sb.Append("Title of Courtesy: ")
            sb.Append(reader("TitleOfCourtesy"))
            sb.Append("<br />")
            sb.Append("Date of Birth: ")
            sb.Append(reader("BirthDate"))
            sb.Append("<br />")
            sb.Append("Hire Date: ")
            sb.Append(reader("HireDate"))
            sb.Append("<br />")
            sb.Append("Address: ")
            sb.Append(reader("Address"))
            sb.Append("<br />")
            sb.Append("City: ")
            sb.Append(reader("City"))
            sb.Append("<br />")
            sb.Append("Region: ")
            sb.Append(reader("Region"))
            sb.Append("<br />")
            sb.Append("PostalCode: ")
            sb.Append(reader("PostalCode"))
            sb.Append("<br />")
            sb.Append("Country: ")
            sb.Append(reader("Country"))
            sb.Append("<br />")
            lblInfo.Text = sb.ToString()
            reader.Close()

        Catch err As Exception

            lblInfo.Text = "Error getting Employee. "
            lblInfo.Text &= err.Message

        Finally
            myConnection.Close()
        End Try


End Sub
```

Κώδικας 11.19 – Εμφάνιση πληροφοριών των υπαλλήλων κατά την επιλογή από το DropDownList

11.1.5.Ενημέρωση εγγραφών

Τώρα που είδαμε πώς να λαμβάνουμε δεδομένα, οι διαδικασίες εισαγωγής, ενημέρωσης και διαγραφής δεδομένων δεν θα μας φανούν ιδιαίτερα δύσκολες. Και εδώ χρησιμοποιούμε το αντικείμενο Command, ωστόσο αυτή τη φορά δεν θα χρειαστούμε έναν DataReader γιατί δεν επιστρέφονται δεδομένα. Επίσης δεν θα χρησιμοποιήσουμε την εντολή SQL Select. Αντίθετα θα χρησιμοποιούμε τις εντολές Update, Insert ή Delete αντίστοιχα.

Για εκτελέσουμε μια εντολή Update, Insert ή Delete χρειαζόμαστε ένα αντικείμενο Command. Μετά μπορούμε να εκτελέσουμε τη μέθοδο ExecuteNonQuery(). Αυτή η μέθοδος επιστρέφει το πλήθος των γραμμών που επηρεάζονται, κάτι που επιτρέπει να ελέγξουμε αν το αποτέλεσμα του ερωτήματος είναι το επιθυμητό. Για παράδειγμα, αν επιχειρήσουμε να ενημερώσουμε ή να διαγράψουμε μια εγγραφή και πληροφορηθούμε ότι δεν επηρεάστη-

καν κάποιες εγγραφές, τότε πιθανότερα να έχουμε κάποιο λογικό λάθος στο ερώτημά μας (π.χ. κάποια μη έγκυρη τιμή η κάποιο άστοχο Where). Αν είχαμε κάποιο συντακτικό λάθος στο ερώτημά μας τότε θα προέκυπτε exception.

Για να ενημερώσουμε τα πεδία μιας εγγραφής η πιο ενδεδειγμένη κίνηση είναι η εμφάνιση των υπαρχόντων τιμών των πεδίων στα κατάλληλα controls έτσι ώστε να μπορούμε να τις αλλάξουμε.

Η σελίδα ASP έχει ελαφρώς αλλάξει ώστε να εμφανίζει text boxes μέσω των οποίων θα μπορούσαν να εισαχθούν νέες τιμές σε λίγα από τα πεδία ενός υπαλλήλου.

ASP

```
<%@ Page Language="C#" AutoEventWireup="true" Code-
File="Example DataReader Update.aspx.cs" Inherits="Example DataReader Update"
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <div>
        <asp:HiddenField ID="hdEmployeeID" runat="server" />
    </div>

    <div>
        <asp:DropDownList ID="ddlEmployees" runat="server" onselectedin-
dexchanged="ddlEmployees_SelectedIndexChanged" AutoPostBack="true">
        </asp:DropDownList>
    </div>

    <div>First Name: <asp:TextBox ID="tbFName"
runat="server"></asp:TextBox></div>
    <div>Last Name:  <asp:TextBox ID="tbLName"
runat="server"></asp:TextBox></div>
    <div>Title: <asp:TextBox ID="tbTitle" runat="server"></asp:TextBox></div>
    <div>
        <asp:Button ID="btUpdate" runat="server" Text="Update" /></div>
    </div>

    <br />
        <asp:Label ID="lblInfo" runat="server"></asp:Label>

    </form>
</body>
</html>
```

Ομοίως μέσω κώδικα θα πρέπει να εισάγουμε τις τρέχουσες τιμές των πεδίων στα κατάλληλα text boxes όταν επιλεχθεί μια τιμή από το DropDownList.

C#

```
protected void ddlEmployees SelectedIndexChanged(object sender, EventArgs e)
    {

        string selectSQL;
        selectSQL = "SELECT * FROM Employees";
        selectSQL += " WHERE EmployeeID=" + ddlEmployees.SelectedItem.Value;

        SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;
        SqlCommand cmd = new SqlCommand(selectSQL, myConnection);
        SqlDataReader reader;

        try
        {
            myConnection.Open();
```

```
            reader = cmd.ExecuteReader();
            reader.Read();

            hdEmployeeID.Value = reader["EmployeeID"].ToString();
            tbFName.Text = reader["FirstName"].ToString();
            tbLName.Text = reader["LastName"].ToString();
            tbTitle.Text = reader["Title"].ToString();

            reader.Close();
        }
        catch (Exception err)
        {
            lblInfo.Text = "Error getting Employee. ";
            lblInfo.Text += err.Message;
        }
        finally
        {
            myConnection.Close();
        }

}
```

VB

```
Protected Sub ddlEmployees SelectedIndexChanged(ByVal sender As Object, ByVal e
As System.EventArgs) Handles ddlEmployees.SelectedIndexChanged

        Dim selectSQL As String
        selectSQL = "SELECT * FROM Employees"
        selectSQL &= " WHERE EmployeeID=" & ddlEmployees.SelectedItem.Value

        Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString
        Dim cmd As SqlCommand = New SqlCommand(selectSQL, myConnection)
        Dim reader As SqlDataReader

        Try
            myConnection.Open()
            reader = cmd.ExecuteReader()
            reader.Read()

            tbFName.Text = reader("FirstName").ToString()
            tbLName.Text = reader("LastName").ToString()
            tbTitle.Text = reader("Title").ToString()

            reader.Close()

        Catch err As Exception

            lblInfo.Text = "Error getting Employee. "
            lblInfo.Text &= err.Message

        Finally
            myConnection.Close()
        End Try
    End Sub
```

Κώδικας 11.20 – Εισαγωγή τιμών στα κατάλληλα TextBox controls

Έχοντας φορτωμένες τις τιμές στα αντίστοιχα controls το μόνο που μένει είναι να υλοποιήσουμε την εντολή ενημέρωσης στην περίπτωση που εισάγει νέες τιμές ο χρήστης.

C#

```
public void Update_Data_From_Controls()
    {
        SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;

        try
        {
            myConnection.Open();
            SqlCommand myCommand = new SqlCommand();
            myCommand.Connection = myConnection;
```

```
            myCommand.CommandText = "UPDATE Employees SET LastName =@LastName,
FirstName=@FirstName, Title=@Title WHERE EmployeeID=@EmployeeID";
            myCommand.Parameters.AddWithValue("@LastName", tbLName.Text);
            myCommand.Parameters.AddWithValue("@FirstName", tbFName.Text);
            myCommand.Parameters.AddWithValue("@Title", tbTitle.Text);
            myCommand.Parameters.AddWithValue("@EmployeeID", ddlEmploy-
ees.SelectedValue);
            int result = myCommand.ExecuteNonQuery();

        }
        catch (Exception ex)
        {
        }
        finally
        {
            myConnection.Close();
        }
        FillEmployeeList();
}
```

```
VB

Public Sub Update Data From Controls()
        Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString

        Try

            myConnection.Open()

            Dim myCommand As SqlCommand = New SqlCommand()
            myCommand.Connection = myConnection


            myCommand.CommandText = "UPDATE Employees SET LastName =@LastName,
FirstName=@FirstName, Title=@Title WHERE EmployeeID=@EmployeeID"
            myCommand.Parameters.AddWithValue("@LastName", tbLName.Text)
            myCommand.Parameters.AddWithValue("@FirstName", tbFName.Text)
            myCommand.Parameters.AddWithValue("@Title", tbTitle.Text)
            myCommand.Parameters.AddWithValue("@EmployeeID", ddlEmploy-
ees.SelectedValue)
            Dim result As Integer = myCommand.ExecuteNonQuery()

        Catch ex As Exception

        Finally
            myConnection.Close()
        End Try
        FillEmployeeList()
End Sub
```

Κώδικας 11.21 - Μέθοδος για την ενημέρωση των πεδίων μιας εγγραφής

Στο παραπάνω παράδειγμα παρατηρούμε ότι γίνεται χρήση της *παραμετροποιημένης εντολής (parameterized command)*, η οποία συντάσσεται χρησιμοποιώντας τον χαρακτήρα «@» μπροστά από έναν προσδιοριστή π.χ. @LastName για να ορίσει μια παράμετρο που θα εισάγεται αντί τιμής σε στις παραμέτρους μιας έκφρασης WHERE.

Μια πιο απλή λύση θα ήταν πολύ απλά να δομήσουμε το string της SQL εντολής κάνοντας διαδοχικές συνενώσεις από string, χρησιμοποιώντας τις τιμές των controls που υπέβαλε ο χρήστης με τη φόρμα. Η προσέγγιση αυτή ωστόσο έχει δύο μειονεκτήματα: α) οι χρήστες μπορεί κατά λάθος να εισάγουν χαρακτήρες που επηρεάζουν την πρόταση SQL, για παράδειγμα, αν μια τιμή περιέχει μια απόστροφο το ερώτημα SQL που θα σχηματιστεί δεν θα είναι έγκυρο και β) οι χρήστες μπορεί επίτηδες να εισάγουν χαρακτήρες που επηρεάζουν το ερώτημα, όπως για παράδειγμα να εισάγουν μια μονή απόστροφο για να κλείσουν το ερώτημα που σχηματίζουμε προγραμματιστικά και να το συνεχίσουν με ένα δικό τους «κακόβουλο» ερώτημα. Το δεύτερο από τα δυο αυτά μειονεκτήματα καλείται SQL injection attack και πρέπει να λαμβάνεται πάντα υπόψη. Μια πιθανή αντιμετώπιση είναι με τη χρήστη των παραμετροποιημένων εντολών που αποτελούν ένα βασικό μέτρο ασφάλειας.

11.1.6.Εισαγωγή εγγραφών

Για να προσθέσουμε μια νέα εγγραφή, χρησιμοποιούμε την ίδια λίστα από controls μόνο που στην περίπτωση αυτή δεν τα αρχικοποιούμε με τις τιμές των πεδίων μιας συγκεκριμένης εγγραφής. Τα αφήνουμε κενά. Κατά τα άλλα στο μόνο που διαφέρει η εισαγωγή μιας νέας εγγραφής από την ενημέρωση μιας ήδη υπάρχουσας είναι η εντολή στο ερώτημα SQL. Η παρακάτω μέθοδος θα καλείται στο πάτημα του κουμπιού Insert.

C#

```
public void Insert Data From Controls()
    {
        SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;

        try
        {
            myConnection.Open();
            SqlCommand myCommand = new SqlCommand();
            myCommand.Connection = myConnection;
            myCommand.CommandText = "INSERT INTO Employees(LastName, FirstName,
Title) VALUES (@LastName, @FirstName, @Title)";
            myCommand.Parameters.AddWithValue("@LastName", tbLName.Text);
            myCommand.Parameters.AddWithValue("@FirstName", tbFName.Text);
            myCommand.Parameters.AddWithValue("@Title", tbTitle.Text);
            int result = myCommand.ExecuteNonQuery();

        }
        catch (Exception ex)
        {
        }
        finally
        {
            myConnection.Close();
        }
        FillEmployeeList();
}
```

VB

```
Public Sub Insert_Data_From_Controls()
        Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString

        Try

            myConnection.Open()

            Dim myCommand As SqlCommand = New SqlCommand()
            myCommand.Connection = myConnection


            myCommand.CommandText = "INSERT INTO Employees(LastName, FirstName,
Title) VALUES (@LastName, @FirstName, @Title)"
            myCommand.Parameters.AddWithValue("@LastName", tbLName.Text)
            myCommand.Parameters.AddWithValue("@FirstName", tbFName.Text)
            myCommand.Parameters.AddWithValue("@Title", tbTitle.Text)
            Dim result As Integer = myCommand.ExecuteNonQuery()

        Catch ex As Exception

        Finally
            myConnection.Close()
        End Try
        FillEmployeeList()
End Sub
```

Κώδικας 11.22 - Μέθοδος για την εισαγωγή μιας νέας εγγραφής

11.1.7. Διαγραφή εγγραφών

Η διαγραφή μιας εγγραφής είναι παρόμοια με την ενημέρωση μιας εγγραφής ως προς το γεγονός ότι χρειάζεται το ID της εγγραφής προς διαγραφή, ενώ ουσιαστικά διαφέρουν στο ερώτημα που υποβάλλεται στη βάση. Πρέπει να προσέχουμε ότι μια εντολή διαγραφής δεν θα πετύχει αν η εγγραφή συνδέεται με μια εγγραφή σε άλλο πίνακα μέσω ξένου κλειδιού.

```
C#

SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;

        try
        {
            myConnection.Open();
            SqlCommand myCommand = new SqlCommand();
            myCommand.Connection = myConnection;
            myCommand.CommandText = "DELETE FROM Employees WHERE (EmployeeID =
@EmployeeID)";
            myCommand.Parameters.AddWithValue("@EmployeeID", ddlEmploy-
ees.SelectedValue);
            int result = myCommand.ExecuteNonQuery();

        }
        catch (Exception ex)
        {
        }
        finally
        {
            myConnection.Close();
        }
        FillEmployeeList();
```

```
VB

Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString

        Try

            myConnection.Open()

            Dim myCommand As SqlCommand = New SqlCommand()
            myCommand.Connection = myConnection


            myCommand.CommandText = "DELETE FROM Employees WHERE (EmployeeID =
@EmployeeID)"
            myCommand.Parameters.AddWithValue("@EmployeeID", ddlEmploy-
ees.SelectedValue)
            Dim result As Integer = myCommand.ExecuteNonQuery()

        Catch ex As Exception

        Finally
            myConnection.Close()
        End Try
        FillEmployeeList()
```

Κώδικας 11.23 – Κώδικας για τη διαγραφή μιας εγγραφής

11.1.8. Αποσυνδεδεμένη πρόσβαση σε δεδομένα

Η αποσυνδεδεμένη πρόσβαση σε δεδομένα ουσιαστικά σημαίνει τη διατήρηση των δεδομένων που έχουν ανακτηθεί από μια βάση δεδομένων στη μνήμη χρησιμοποιώντας την κλάση DataSet. Συνδεόμαστε στη βάση όσο χρειάζεται για να λάβουμε τα δεδομένα και να τα εισάγουμε σε ένα DataSet και μόλις ολοκληρωθεί η λήψη τους αποσυνδεόμαστε από τη βάση.

Υπάρχουν πολλαπλοί λόγοι για τους οποίους είναι καλό να χρησιμοποιούμε ένα DataSet για να κρατάμε δεδομένα στη μνήμη:

- Χρειάζεται να επεξεργαστούμε τα δεδομένα, κάτι που μπορεί να διαρκέσει πολύ. Φορτώνοντας τα δεδομένα στη μνήμη εξασφαλίζουμε ότι η βάση θα παραμείνει ανοιχτή για όσο το δυνατόν λιγότερο γίνεται.
- Χρειάζεται να χρησιμοποιήσουμε το ASP.NET data binding για να γεμίσουμε ένα web control (όπως ένα GridView) με τα δεδομένα μας. Παρόλο που μπορούμε να χρησιμοποιήσουμε το DataReader, αυτό δεν δουλεύει σε όλα τα σενάρια.
- Πρέπει να κινηθούμε μπρος και πίσω κατά την επεξεργασία των δεδομένων. Αυτό δεν είναι δυνατό με το DataReader, το οποίο μπορεί να κινηθεί μόνο προς τα μπροστά.
- Χρειάζεται να κινηθούμε από τον ένα πίνακα στον άλλο. Χρησιμοποιώντας το DataSet, μπορούμε να αποθηκεύσουμε πολλαπλούς πίνακες και να ορίσουμε σχέσεις που μας επιτρέπουν να διατρέξουμε τις εγγραφές πιο αποδοτικά.
- Θέλουμε να αποθηκεύσουμε τα δεδομένα σε ένα αρχείο για αργότερη χρήση. Η DataSet περιλαμβάνει δυο μεθόδους τη WriteXml() και ReadXml() που επιτρέπουν να γράψουμε το περιεχόμενο σε ένα αρχείο και να το μετατρέψουμε σε ένα κανονικό αντικείμενο βάσης μετά.
- Χρειαζόμαστε ένα βολικό τρόπο να μεταφέρουμε δεδομένα από ένα component σε ένα άλλο.
- Χρειαζόμαστε να αποθηκεύσουμε μερικά δεδομένα που ζητούνται συχνά για μελλοντικές αιτήσεις. Αυτή η τεχνική καλείται caching.

Το DataSet παρακολουθεί τις αλλαγές που γίνονται στις εγγραφές του, κάτι που μας επιτρέπει να το χρησιμοποιήσουμε για να ενημερώσουμε εγγραφές. Η βασική αρχή είναι απλή. Γεμίζουμε ένα DataSet κανονικά, αλλάζουμε μια ή περισσότερες εγγραφές και ενημερώνουμε τη βάση χρησιμοποιώντας έναν DataAdapter.

Για τους λόγους αυτούς η πλειονότητα των ASP.NET εφαρμογών χρησιμοποιούν το αντικείμενο DataSet για να αποθηκεύουν δεδομένα αλλά όχι για να κάνουν αλλαγές. Αντίθετα χρησιμοποιούν άμεσες αλλαγές για να υποβάλλουν τις αλλαγές.

Η οικογένεια κλάσεων της DataSet αποτελείται από τις εξής κλάσεις:

1. Τη DataSet που αποτελείται από
    - Μια συλλογή από αντικείμενα DataTable
    - Μια συλλογή από αντικείμενα DataRelation
2. Τη DataTable που με τη σειρά της αποτελείται από
    - Μια συλλογή από αντικείμενα DataRow
    - Μια συλλογή από αντικείμενα DataColumn
    - Μια συλλογή από αντικείμενα Constraint
    - Μια συλλογή από αντικείμενα DataView

Γεμίζουμε ένα DataSet με τον ίδιο τρόπο που συνδέουμε έναν DataReader. Ωστόσο, παρόλο που ο DataReader κρατά μια σύνδεση, τα δεδομένα σε ένα DataSet είναι αποσυνδεδεμένα από τη βάση. Ακολουθεί το παράδειγμα που παρουσιάσαμε παραπάνω μόνο που αυτή τη φορά χρησιμοποιείται ένα DataSet αντί να χρησιμοποιηθεί ένας Data Reader.

C#

```
SqlConnection myConnection = new SqlConnection();
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;
        string selectSQL = "SELECT EmployeeID, LastName, FirstName, Title FROM
Employees";
        SqlCommand cmd = new SqlCommand(selectSQL, myConnection);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        DataSet ds = new DataSet();

        try
        {
            myConnection.Open();
            adapter.Fill(ds, "Employees");
```

```
        }
        catch (Exception err)
        {

        }
        finally
        {
            myConnection.Close();
        }


        foreach (DataRow row in ds.Tables["Employees"].Rows)
        {
            Response.Write(row["LastName"] + ", " + row["FirstName"] +
"<br/>");
        }
```

VB

```
Dim myConnection As SqlConnection = New SqlConnection()
        myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString
        Dim selectSQL As String = "SELECT EmployeeID, LastName, FirstName, Ti-
tle FROM Employees"
        Dim cmd As SqlCommand = New SqlCommand(selectSQL, myConnection)
        Dim adapter As SqlDataAdapter = New SqlDataAdapter(cmd)
        Dim ds As DataSet = New DataSet()

        Try
            myConnection.Open()
            adapter.Fill(ds, "Employees")

        Catch err As Exception

        Finally

            myConnection.Close()

        End Try


        For Each row As DataRow In ds.Tables("Employees").Rows
            Response.Write(row("LastName") & ", " & row("FirstName") & "<br/>")
        Next
```

Κώδικας 11.24 – Ανάκτηση και παρουσίαση δεδομένων μέσω ενός αντικειμένου DataSet

Η μέθοδος DataAdapter.Fill() παίρνει σαν είσοδο ένα DataSet και εισάγει δεδομένα από έναν πίνακα της βάσης. Στην περίπτωσή μας, ο πίνακας ονομάζεται Employees και αυτό το όνομα χρησιμοποιείται και αργότερα για να προσπελαστεί ο πίνακας στο DataSet.

Επιλογή πολλαπλών πινάκων

Ένα DataSet μπορεί να περιέχει όσους πίνακες χρειαζόμαστε και μπορούμε να προσθέσουμε σχέσεις στους πίνακες για να προσομοιάσουμε καλύτερα τα αντίστοιχα δεδομένα στη βάση. Δυστυχώς δεν γίνεται να συνδέσουμε αυτόματα τους πίνακες βασιζόμενοι στις υπάρχουσες σχέσεις τους στη βάση δεδομένων. Ωστόσο, γίνεται να προσθέσουμε σχέσεις με μερικές επιπρόσθετες γραμμές κώδικα, όπως φαίνεται στο παράδειγμα παρακάτω.

Στη βάση Northwind, ο πίνακας Employees συνδέεται με τον πίνακα Territories μέσω ενός τρίτου πίνακα του EmployeeTerritories. Στο επόμενο παράδειγμα φαίνεται μια απλή σελίδα που απαριθμεί τους υπαλλήλους και τις περιοχές στις οποίες αυτοί δραστηριοποιούνται. Το ενδιαφέρον κομμάτι σε αυτό το παράδειγμα βρίσκεται στο γεγονός ότι γίνεται χρήση του μηχανισμού της σύνδεσης πινάκων.

Αρχικά δημιουργούνται τα κλασικά αντικείμενα του ADO.NET μαζί με ένα DataSet. Όλα αυτά τα βήματα πραγματοποιούνται στην custom μέθοδο CreateList(), που καλείται από τον Page.Load event handler έτσι ώστε η έξοδος να δημιουργείται όταν παράγεται η σελίδα για πρώτη φορά. Έπειτα εξάγεται όλη η πληροφορία από τη βάση και τοποθετείται μέσα σε ένα DataSet. Αυτή η ενέργεια θα πραγματοποιούνταν με ξεχωριστά αντικείμενα

Command, αλλά για να γίνει λίγο καθαρότερος ο κώδικας, το παράδειγμα χρησιμοποιεί μόνο ένα και αλλάζει την ιδιότητα CommandText κάθε φορά:

```
C#

try
        {
            myConnection.Open();
            adapter.Fill(ds, "Employees");

            cmd.CommandText = "SELECT EmployeeID, TerritoryID FROM EmployeeTer-
ritories";
            adapter.Fill(ds, "EmployeeTerritories");

            cmd.CommandText = "SELECT TerritoryID, TerritoryDescription, Re-
gionID FROM Territories";
            adapter.Fill(ds, "Territories");

        }
        catch (Exception err)
        {
            lblList.Text = "Error reading list of names. ";
            lblList.Text += err.Message;
        }
        finally
        {
            myConnection.Close();
        }
```

```
VB

Try
            myConnection.Open()
            adapter.Fill(ds, "Employees")

            cmd.CommandText = "SELECT EmployeeID, TerritoryID FROM EmployeeTer-
ritories"
            adapter.Fill(ds, "EmployeeTerritories")

            cmd.CommandText = "SELECT TerritoryID, TerritoryDescription, Re-
gionID FROM Territories"
            adapter.Fill(ds, "Territories")
Catch err As Exception

            lblList.Text = "Error reading list of names. "
            lblList.Text &= err.Message

Finally

            myConnection.Close()

End Try
```

Ορίζοντας σχέσεις

Τώρα που τα δεδομένα είναι μέσα στο DataSet, μπορούμε να δημιουργήσουμε δύο αντικείμενα DataRelation για να διευκολύνουμε την πλοήγηση μεταξύ διασυνδεδεμένης πληροφορίας. Σε αυτήν την περίπτωση, αυτά τα αντικείμενα DataRelation αναπαριστούν τους περιορισμούς ξένου κλειδιού που ορίζονται στη βάση. Θυμίζουμε ότι ένα ξένο κλειδί είναι ένας περιορισμός που θέτουμε σε μια βάση, ο οποίος συσχετίζει έναν πίνακα με έναν άλλο.

Για να ορίσουμε ένα DataRelation, χρειάζεται να προσδιορίσουμε τα σχετιζόμενα πεδία από δυο διαφορετικούς πίνακες και πρέπει να του δώσουμε ένα μοναδικό όνομα. Η σειρά των σχετιζόμενων πεδίων είναι σημαντική. Σε αυτό το παράδειγμα κάθε υπάλληλος μπορεί να έχει περισσότερες από μια εισόδους στον πίνακα EmployeeTerritories. Κάθε περιοχή μπορεί να έχει περισσότερες εγγραφές στον πίνακα EmployeeTerritories. Μόλις έχουμε δημιουργήσει αυτά τα αντικείμενα DataRelation μπορούμε να τα εισάγουμε στο DataSet.

```
C#
```

```
DataRelation Employee_EmployeeTerritories = new DataRela-
tion("Employee EmployeeTerritories",
ds.Tables["Employees"].Columns["EmployeeID"],
ds.Tables["EmployeeTerritories"].Columns["EmployeeID"]);
ds.Relations.Add(Employee_EmployeeTerritories);

DataRelation Territories EmployeeTerritories = new DataRela-
tion("Territories EmployeeTerritories",
ds.Tables["Territories"].Columns["TerritoryID"],
ds.Tables["EmployeeTerritories"].Columns["TerritoryID"]);
ds.Relations.Add(Territories_EmployeeTerritories);
```

VB

```
Dim Employee_EmployeeTerritories As DataRelation = New DataRela-
tion("Employee EmployeeTerritories",
ds.Tables("Employees").Columns("EmployeeID"),
ds.Tables("EmployeeTerritories").Columns("EmployeeID"))
            ds.Relations.Add(Employee EmployeeTerritories)
            Dim Territories_EmployeeTerritories As DataRelation = New DataRela-
tion("Territories_EmployeeTerritories",
ds.Tables("Territories").Columns("TerritoryID"),
ds.Tables("EmployeeTerritories").Columns("TerritoryID"))
            ds.Relations.Add(Territories_EmployeeTerritories)
```

Κώδικας 11.25 – Ορισμός σχέσεων μεταξύ πινάκων στο DataSet

Ο υπόλοιπος κώδικας διατρέχει επαναληπτικά το DataSet. Ωστόσο, αντίθετα με το προηγούμενο παράδειγμα, το οποίο διέτρεχε ένα μόνο πίνακα, αυτό το παράδειγμα χρησιμοποιεί τα αντικείμενα DataRelation για να «διακλαδώσει» τους σχετιζόμενους πίνακες. Η διαδικασία είναι η εξής:

1. Επιλογή της πρώτη εγγραφής του πίνακα Employees.
2. Χρησιμοποιώντας τη σχέση Employee_EmployeeTerritories, βρίσκουμε όλες τις εγγραφές παιδιά που αντιστοιχούν σε αυτόν το συγγραφέα. Για να το κάνουμε αυτό καλούμε τη μέθοδο DataRow.GetChildRows() και την περνάμε στο αντικείμενο DataRelationship που μοντελοποιεί τη σχέση ανάμεσα στα Employees και τον πίνακα EmployeeTerritories.
3. Για κάθε αποτέλεσμα του EmployeeTerritories, ελέγχεται η αντίστοιχη εγγραφή στον Territories για να ληφθεί ο πλήρης τίτλος, καλώντας τη μέθοδο DataRow.GetParentRows() και περνώντας το αντικείμενο DataRelationship που συνδέει τον EmployeeTerritories και τον Territories.
4. Μετάβαση στην επόμενη εγγραφή και επανάληψη της διαδικασίας.

Ο κώδικας είναι αρκετά ξεκάθαρος.

```
C#

foreach (DataRow rowEmployee in ds.Tables["Employees"].Rows)
            {
                lblList.Text += "<br /><b>" + rowEmployee["LastName"];
                lblList.Text += " " + rowEmployee["FirstName"] + "</b><br />";
                foreach (DataRow rowEmployeeTerritories in rowEmploy-
ee.GetChildRows(Employee_EmployeeTerritories))
                {
                    DataRow rowTerritories = rowEmployeeTerrito-
ries.GetParentRows(Territories EmployeeTerritories)[0];
                    lblList.Text += "  ";
                    lblList.Text += rowTerritories["TerritoryDescription"] +
"<br />";
                }
}
```

VB

```
For Each rowEmployee As DataRow In ds.Tables("Employees").Rows

                lblList.Text &= "<br /><b>" & rowEmployee("LastName")
                lblList.Text &= " " & rowEmployee("FirstName") & "</b><br />"
                For Each rowEmployeeTerritories As DataRow In rowEmploy-
ee.GetChildRows(Employee EmployeeTerritories)
```

```
                    Dim rowTerritories As DataRow = rowEmployeeTerrito-
ries.GetParentRows(Territories EmployeeTerritories)(0)
                    lblList.Text &= "  "
                    lblList.Text &= rowTerritories("TerritoryDescription") &
"<br />"
                Next

Next
```

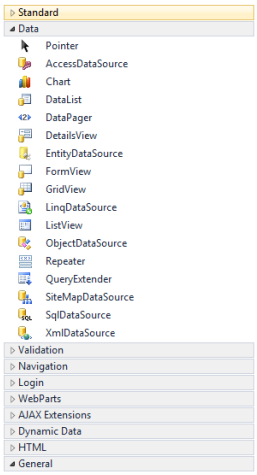
Κώδικας 11.26 – Ανάκτηση δεδομένων μέσω DataSet από δύο συνδεδεμένους πίνακες

11.2. Data Controls

Για την αποδοτική μεταχείριση δεδομένων, η ASP.NET παρέχει δυο είδη από data controls:

- Τα data-bound controls
- Τα data source controls

Το πρώτο είδος περιέχει controls που χρησιμοποιούμε για να εμφανίσουμε και να επεξεργαστούμε δεδομένα, όπως το GridView, το Repeater και το ListView σε μια σελίδα. Το δεύτερο είδος περιέχει controls που χρησιμοποιούνται για τη λήψη δεδομένων από μια πηγή δεδομένων, όπως μια βάση δεδομένων ή ένα αρχείο XML και τότε να προωθήσουν τα δεδομένα αυτά στα data-bound controls. Μπορούμε να δούμε τα διαθέσιμα data controls αν επεκτείνουμε τον κόμβο data από το toolbox. Τα διαθέσιμα αυτά controls φαίνονται στην παρακάτω εικόνα.



Εικόνα 11.1 – Τα διαθέσιμα data bound και data source controls

Data-bound Controls

Τα data-bound controls χρησιμοποιούνται για την εμφάνιση και επεξεργασία δεδομένων σε μια ιστοσελίδα και είναι εφτά. Τα διακρίνουμε περαιτέρω σε α) List controls τα οποία εμφανίζουν πολλαπλές εγγραφές ταυτόχρονα, β) Single Item Controls που εμφανίζουν μια εγγραφή κάθε φορά και γ) paging controlσ που βοηθάει στην παρουσίαση των δεδομένων σε σελίδες. Στην παρούσα έκδοση του .NET υπάρχει μόνο ένα τέτοιο control.

List Controls

Επειδή η ASP.NET παρέχει διάφορα controls για την εμφάνιση λιστών εγγραφών, μπορεί να αναρωτηθούμε πότε είναι καταλληλότερο να προτιμήσουμε ένα control έναντι των άλλων.

Το Gridview είναι ένα ευέλικτο control που παρέχει αυτόματη σελιδοποίηση (όπου οι εγγραφές εμφανίζονται σε πολλαπλές σελίδες), ταξινόμηση, επεξεργασία, διαγραφή και επιλογή. Εμφανίζει τα δεδομένα σαν ένα λογιστικό φύλλο τύπου excel με γραμμές και στήλες, όπου κάθε γραμμή περιέχει μια πλήρη εγγραφή.

Παρόλο που υπάρχουν αρκετές επιλογές για τη μορφοποίηση αυτού του control, δεν γίνεται να αλλάξουμε ριζικά τον τρόπο με τον οποίο αυτά παρουσιάζονται. Επιπρόσθετα, το GridView δεν επιτρέπει την άμεση εισαγωγή εγγραφών στην πηγή των δεδομένων.

Το DataList control επιτρέπει την παρουσίαση εγγραφών δεδομένων όχι μόνο σε γραμμές όπως το GridView, αλλά και σαν στήλες, επιτρέποντας μια «πινακοειδή» παρουσίαση των δεδομένων. Το control αυτό, επιτρέπει τον ορισμό του παρουσιαστικού των δεδομένων μέσα από templates. Το control αυτό όμως, δεν υποστηρίζει paging και sorting από μόνο του και δεν επιτρέπει την εισαγωγή νέων εγγραφών ή την ενημέρωση και διαγραφή υπαρχόντων εγγραφών.

Το Repeater control δίνει τη μεγαλύτερη ευελιξία σχετικά με την HTML που παράγεται, επειδή το ίδιο το control δεν παράγει καθόλου HTML στην έξοδο στη σελίδα. Για το λόγο αυτό, συχνά χρησιμοποιείται για την εμφάνιση αριθμημένων λιστών ή λιστών που δεν επιθυμούμε να έχουν περιττή HTML. Ορίζουμε ολόκληρο το markup μέσα από τα υπεράριθμα που συνοδεύουν το control. Η ευελιξία αυτή όμως έχει με ένα τίμημα: το control δεν έχει ενσωματωμένες δυνατότητες σελιδοποίησης, ταξινόμησης ή επεξεργασίας των δεδομένων.

Το ListView εισήχθη στην ASP.NET 3.5 και είναι ο καλύτερος συνδυασμός του GridView, της DataList και του Repeater. Έλαβε μερικές αλλαγές στην ASP.NET 4 που το έκαναν ακόμα ευκολότερο στη χρήση. Το control επιτρέπει την επεξεργασία, τη διαγραφή και τη σελιδοποίηση των δεδομένων, όπως το GridView. Υποστηρίζει τα multi-column και multi-row layouts όπως το DataList και επιτρέπει τον πλήρη έλεγχο του markup που παράγεται από το control, όπως ακριβώς επιτρέπει και το Repeater.

Single Item Controls

Τα DetailsView και FormView controls μοιάζουν ως προς το γεγονός ότι εμφανίζουν μια μόνο εγγραφή κάθε φορά. Το DetailsView χρησιμοποιεί ένα ενσωματωμένο «πινακοειδή» τρόπο εμφάνισης δεδομένων, ενώ το FormView control επιτρέπει τον ορισμό του παρουσιαστικού από εμάς. Και τα δύο controls μας δίνουν τη δυνατότητα ορισμού templates για διαφορετικές περιπτώσεις όπως τη read-only εμφάνιση δεδομένων και την εισαγωγή και ενημέρωση δεδομένων.

Paging Controls

Άλλο ένα χρήσιμο control είναι το DataPager, που επιτρέπει τη σελιδοποίηση των διαφόρων controls. Μέχρι τώρα όμως μπορεί να χρησιμοποιηθεί μόνο για την επέκταση του ListView control, αλλά αυτό μπορεί να αλλάξει σε μελλοντικές εκδόσεις του .NET.

Data Source Controls

Η κατηγορία δεδομένων του Toolbox περιέχει εφτά διαφορετικά controls τύπου Data Source που μπορούμε να χρησιμοποιήσουμε για να δέσουμε δεδομένα στα data-bound controls.

Το XMLDataSource και το SiteMapDataSource χρησιμοποιούνται για να δέσουν ιεραρχικά δεδομένα σε μορφή XML σε αυτά τα controls. Είδαμε το SiteMapDataSource όταν δημιουργήσαμε ένα sitemap στο κεφάλαιο 3.

Το control AccessDataSource χρησιμοποιείται για την εμφάνιση δεδομένων από μια βάση δεδομένων Microsoft Access. Η εργασία με αυτό το control είναι αρκετά απλή και μέχρις ενός σημείου όμοια με το control SqlDataSource το οποίο χρησιμοποιείται για σύνδεση σε βάσεις κυρίως SQL Server και το οποίο θα μελετήσουμε αναλυτικά στη συνέχεια.

Το control ObjectDataSource επιτρέπει τη σύνδεση των data-bound controls σε διαφορετικά αντικείμενα στην εφαρμογή μας. Αντί να δένουμε τα controls που εργάζονται με τα δεδομένα απευθείας με μια βάση δεδομένων, τα δένουμε με δεδομένα αντικειμένων από ένα ξεχωριστό επίπεδο.

Τα υπόλοιπα τρία controls είναι τα το EntityDataSource και το LinqDataSource. Αυτά θα μελετηθούν αναλυτικότερα σε επόμενα κεφάλαια.

11.3. SqlDataSource

Το control SqlDataSource μας δίνει πρόσβαση σε δεδομένα που βρίσκονται σε μια σχεσιακή βάση δεδομένων. Αυτή μπορεί να είναι μια βάση δεδομένων σε SQL Server, Oracle αλλά και οποιαδήποτε OLE DB και ODBC βάση δεδομένων. Μπορούμε να χρησιμοποιήσουμε το SqlDataSource με οποιοδήποτε data-bound control όπως το GridView, FormView και

DetailsView για την εμφάνιση και την επεξεργασία δεδομένων σε μια σελίδα χρησιμοποιώντας ελάχιστο ή ακόμα και καθόλου κώδικα.

Χρησιμοποιώντας το SqlDataSource μπορούμε να αποκτήσουμε πρόσβαση και να επεξεργαστούμε τα δεδομένα μιας βάσης δεδομένων από μια ASP.NET σελίδα χωρίς να χρησιμοποιούμε άμεσα τις κλάσεις του ADO.NET. Σε αυτή την περίπτωση παρέχουμε απλά ένα connection string για να συνδεθούμε στη βάση και ορίζουμε τα SQL ερωτήματα ή τις stored procedures που χρησιμοποιούν τα δεδομένα μας. Κατά την εκτέλεση, το ίδιο το control ανοίγει αυτόματα τη σύνδεση, εκτελεί μια εντολή SQL ή μια stored procedure και αφού επιστρέψει τα επιλεγμένα δεδομένα (αν υπάρχουν), κλείνει τη σύνδεση.

11.3.1. Σύνδεση του SqlDataSource με μια πηγή δεδομένων

Όταν ρυθμίζουμε ένα SqlDataSource control, ορίζουμε την ιδιότητα ProviderName στον τύπο της βάσης δεδομένων (ο προεπιλεγμένος είναι ο System.Data.SqlClient) και την ιδιότητα ConnectionString σε ένα connection string που περιλαμβάνει τις πληροφορίες που χρειάζονται για να συνδεθούμε στη βάση. Τα περιεχόμενα του connection string διαφέρουν ανάλογα με τον τύπο της βάσης δεδομένων με τον οποίο επικοινωνεί το control. Για παράδειγμα το control SqlDataSource απαιτεί ένα όνομα server, ένα όνομα βάσης και πληροφορίες για το πώς θα αυθεντικοποιήσουμε το χρήστη που συνδέεται στον SQL Server (εδώ να τονίσουμε ότι ο χρήστης είναι του SQL Server όχι της ASP.NET εφαρμογής). Για να βρούμε τα έγκυρα connection strings πρέπει να ανατρέξουμε στο documentation των κλάσεων SqlConnection, OracleConnection, OleDbConnection και OdbcConnection και να εντοπίσουμε την ιδιότητα ConnectionString.

11.3.2.Εντολές με το SqlDataSource Control

Μπορούμε να ορίσουμε μέχρι τέσσερις εντολές (δηλαδή ερωτήματα SQL) για το control SqlDataSource: μια SelectCommand, μια UpdateCommand, μια DeleteCommand και μια InsertCommand. Κάθε εντολή είναι μια ξεχωριστή ιδιότητα του control. Για κάθε εντολή, ορίζουμε ένα ερώτημα SQL το οποίο θα εκτελεί το control. Μπορούμε να ορίσουμε και το όνομα μιας stored procedure αν αυτό υποστηρίζεται από τη βάση δεδομένων στην οποία συνδέεται το control.

Μπορούμε να ορίσουμε παραμετροποιημένες εντολές όταν θα πρέπει να συμπεριλάβουμε τιμές που ορίζονται κατά την εκτέλεση. Το επόμενο παράδειγμα δείχνει μια τυπική παραμετροποιημένη εντολή:

```
Select CustomerID, CompanyName From Customers Where City = @city
```

Κώδικας 11.27 – Εντολή SQL με παράμετρο

Εντός του SqlDataSource, μπορούμε να δημιουργήσουμε αντικείμενα παραμέτρων που ορίζουν από πού θα πάρει τις τιμές η εντολή κατά την εκτέλεσή της. Θα μπορούσε να είναι ένα άλλο control της σελίδας, ένα query string, μια παράμετρος του Session και ούτω καθεξής. Εναλλακτικά μπορούμε να ορίσουμε τις τιμές προγραμματιστικά.

Το control εκτελεί τις εντολές όταν καλείται η αντίστοιχη μέθοδος Select, Update, Delete ή Insert. Η μέθοδος Select καλείται αυτόματα όταν καλούμε τη μέθοδο DataBind της σελίδας ή ενός control που είναι δεμένο με το control. Μπορούμε επίσης να καλέσουμε οποιαδήποτε από τις τέσσερις μεθόδους ρητά όταν θέλουμε το control να εκτελεί μια εντολή. Μερικά controls όπως το GridView, μπορεί να καλούν τις μεθόδους αυτόματα, χωρίς να απαιτούν να καλούμε τις μεθόδους ρητά.

Διάβασμα δεδομένων με το SqlDataSource

Μπορούμε να χρησιμοποιήσουμε το SqlDataSource για να λάβουμε δεδομένα από μια βάση με ελάχιστο ή και καθόλου κώδικα. Το control SqlDataSource μπορεί να δουλέψει με κάθε βάση δεδομένων που για την οποία υπάρχει ένας ADO.NET provider διαθέσιμος στις ρυθμίσεις. Όπως αναφέραμε αυτές περιλαμβάνουν τον SQL Server, την Oracle, και τις βάσεις με τις οποίες θα επικοινωνούμε μέσω της ODBC ή του OLEDB. Η βάση δεδομένων είναι που θα εκτελεί τα ερωτήματα (ή τις stored procedures) που ορίζονται στο

SqlDataSource. Αυτό που πρέπει να θυμόμαστε ωστόσο είναι ότι το SqlDataSource control συμπεριφέρεται με τον ίδιο τρόπο για όλες τις βάσεις δεδομένων. Επομένως το μαθαίνουμε μια φορά και το χρησιμοποιούμε σε όλες τις βάσεις.

Για να λάβουμε τα δεδομένα από μια βάση χρησιμοποιώντας το SqlDataSource control, πρέπει να θέσουμε τουλάχιστον τις εξής properties:

- ProviderName: το οποίο λαμβάνει το όνομα του ADO.NET provider που αναπαριστά τη βάση με την οποία δουλεύουμε. Εμείς που δουλεύουμε με τον SQL Server θέτουμε την ιδιότητα ProviderName σε "System.Data.SqlClient". Αν θέλαμε να δουλέψουμε με μια Oracle βάση δεδομένων θα θέταμε "System.Data.OracleClient" και ούτω καθεξής.
- ConnectionString: το οποίο λαμβάνει το κατάλληλο connection string για τη βάση δεδομένων μας.
- SelectCommand: το οποίο λαμβάνει ένα ερώτημα SQL ή μια stored procedure που επιστρέφει δεδομένα από μια βάση. Η σύνταξη του ερωτήματος αυτού εξαρτάται από το σχήμα της βάσης δεδομένων που χρησιμοποιούμε.

Ορίζοντας το όνομα ενός Provider

Ορίζουμε την ιδιότητα ProviderName στο όνομα του ADO.NET provider που σχετίζεται με τον τύπο της βάσης όπου είναι αποθηκευμένα τα δεδομένα. Η λίστα με τους διαθέσιμους providers βρίσκεται στις ρυθμίσεις του site, είτε στο αρχείο Machine.config, είτε στο Web.config. Προεπιλεγμένα το SqlDataSource control χρησιμοποιεί τον ADO.NET provider System.Data.SqlClient που αντιστοιχεί στον SQL Server. Συνεπώς, αν συνδεόμαστε με μια βάση SQL Server δεν χρειάζεται να ορίσουμε τον provider ρητά.

Ορίζοντας ένα Connection String

Ορίζουμε την τιμή της ιδιότητας ConnectionString σε ένα string που χρησιμοποιείται για μια συγκεκριμένη βάση δεδομένων. Ωστόσο το να θέτουμε την τιμή της ιδιότητας αυτής μέσα στον κώδικα δεν ευνοεί τη συντήρηση, ιδιαίτερα για τα μεγάλα sites. Για να συντηρείται πιο εύκολα η εφαρμογή μας και για να είναι πιο ασφαλής, είναι προτιμότερο να αποθηκεύουμε τα connection strings στο element connectionStrings στο αρχείο ρυθμίσεων της εφαρμογής μας. Μπορούμε τότε να αναφερθούμε στο string αυτό με μια έκφραση όπως η παρακάτω:

```
ASP

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>"
  SelectCommand="SELECT * FROM [Categories]">
</asp:SqlDataSource>
```

Κώδικας 11.28 – Αναφορά σε ένα ConnectionString το οποίο έχει αποθηκευθεί στο web.config

Για περισσότερη ασφάλεια, μπορούμε να κρυπτογραφήσουμε το κομμάτι <connectionStrings>.

Ορίζοντας την εντολή Select

Μπορούμε να ορίσουμε ένα SQL ερώτημα που θα εκτελεί το SQLDataSource θέτοντας την ιδιότητα SelectCommand. Στο επόμενο παράδειγμα φαίνεται ένα ερώτημα SQL που λαμβάνει ένα σύνολο αποτελεσμάτων που αποτελείται από τα επίθετα των υπαλλήλων στον πίνακα Employees:

```
SELECT LastName FROM Employees
```

Στον παρακάτω κώδικα φαίνεται πως μπορούμε να ορίσουμε τις ιδιότητες ConnectionString και την SelectCommand ενός SqlDataSource control για την εμφάνιση των εγγραφών που επιστρέφονται από το παραπάνω ερώτημα χρησιμοποιώντας ένα ListBox control.

```
ASP
```

```
<%@ Page language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">
      <asp:SqlDataSource
          id="SqlDataSource1"
          runat="server"
          DataSourceMode="DataReader"
          ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
          SelectCommand="SELECT LastName FROM Employees">
      </asp:SqlDataSource>

      <asp:ListBox
          id="ListBox1"
          runat="server"
          DataTextField="LastName"
          DataSourceID="SqlDataSource1">
      </asp:ListBox>

    </form>
  </body>
</html>
```

Κώδικας 11.29 – Σύνδεση ενός SqlDataSource με ένα ListBox

Κατά την εκτέλεση, το control υποβάλει το κείμενο της ιδιότητας SelectCommand στη βάση, η οποία στη συνέχεια του επιστρέφει το αποτέλεσμα του ερωτήματος. Κάθε data-bound control που είναι δεμένο με το συγκεκριμένο SqlDataSource control, εμφανίζει το σύνολο των αποτελεσμάτων στην σελίδα ASP.NET.

Πέρασμα παραμέτρων σε προτάσεις SQL

Οι χρήστες συχνά αλληλεπιδρούν με δεδομένα που βασίζονται σε παραμέτρους που μπορούν να αποτιμηθούν μόνο κατά την εκτέλεση της εφαρμογής. Για παράδειγμα, μια σελίδα μπορεί να εμφανίζει τα δεδομένα μιας συγκεκριμένης ημερομηνίας. Αν ο χρήστης επιλέξει μια διαφορετική ημερομηνία, τα δεδομένα στη σελίδα μπορεί να αλλάξουν. Είτε αλλάζει η ημερομηνία από το χρήστη, είτε αλλάζει προγραμματιστικά από την εφαρμογή, το ερώτημα που υποβάλλεται στη βάση μπορεί να γίνει πιο ευέλικτο και πιο εύκολο στη συντήρηση αν είναι ένα παραμετροποιημένο ερώτημα SQL, στο οποίο τα στοιχεία της πρότασης SQL δένονται με μεταβλητές που αποτιμώνται κατά την εκτέλεση.

Το control υποστηρίζει παραμετροποιημένα ερωτήματα συνδέοντας παραμέτρους που προσθέτουμε στη συλλογή SelectParameters με placeholders στο ερώτημα SelectCommand. Οι τιμές των παραμέτρων μπορούν να διαβαστούν από άλλο control στη σελίδα, από το state του session, από το προφίλ του χρήστη ή από άλλα στοιχεία.

Το συντακτικό που χρησιμοποιείται για τους placeholders ποικίλλει ανάλογα με τον τύπο της βάσης δεδομένων. Αν δουλεύουμε με SQL Server, η παράμετρος ξεκινά με τον χαρακτήρα @ και το όνομά του αντιστοιχεί με το όνομα του αντικειμένου της παραμέτρου στη συλλογή SelectParameters. Αν δουλεύουμε με μια ODBC ή μια βάση δεδομένων OLEDB, οι παράμετροι δεν είναι ονομασμένοι και ορίζονται με τον χαρακτήρα '?'.

Στο επόμενο παράδειγμα φαίνεται πώς ένα ερώτημα SQL λαμβάνει τις παραγγελίες ενός χρήστη βασισμένο στο ID του συνδεδεμένου υπαλλήλου, από τη βάση Northwind.

```
SELECT * FROM Orders WHERE EmployeeID = @empid
```

Στο παραπάνω παράδειγμα η παράμετρος @empid αποτιμάται κατά την εκτέλεση. Στον παρακάτω κώδικα φαίνεται ένα ερώτημα με παραμέτρους που παίρνει την τιμή από ένα άλλο control της σελίδας:

```
ASP
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">

      <p><asp:dropdownlist
          id="DropDownList1"
          runat="server"
          autopostback="True">
          <asp:listitem selected="True">Sales Representative</asp:listitem>
          <asp:listitem>Sales Manager</asp:listitem>
          <asp:listitem>Vice President, Sales</asp:listitem>
      </asp:dropdownlist></p>

      <asp:sqldatasource
          id="SqlDataSource1"
          runat="server"
          connectionstring="<%$ ConnectionStrings:MyNorthwind%>"
          selectcommand="SELECT LastName FROM Employees WHERE Title = @Title">
          <selectparameters>
              <asp:controlparameter name="Title" controlid="DropDownList1"
propertyname="SelectedValue"/>
          </selectparameters>
      </asp:sqldatasource>

      <p><asp:listbox
          id="ListBox1"
          runat="server"
          datasourceid="SqlDataSource1"
          datatextfield="LastName">
      </asp:listbox></p>

    </form>
  </body>
</html>
```

Κώδικας 11.30 - Ερώτημα με παράμετρο που παίρνει την τιμή από ένα άλλο control της σελίδας

Ορίζοντας πώς επιστρέφονται τα δεδομένα

Η ιδιότητα DataSourceMode του SqlDataSource control καθορίζει πως θα διατηρούνται τα δεδομένα από το control. Προεπιλεγμένα η ιδιότητα αυτή τίθεται στην κλάση DataSet κάτι που σημαίνει ότι τα αποτελέσματα που επιστρέφει η βάση αποθηκεύονται στη μνήμη του server από το SqlDataSource control. Αν θέλουμε να μην αποθηκεύονται στη μνήμη του server χρησιμοποιούμε την κλάση DataReader. Η πρώτη περίπτωση ωστόσο είναι αυτή που δίνει τη δυνατότητα στα διάφορα data-bound controls να έχουν τις ιδιότητες της σελιδοποίησης και της ταξινόμησης.

Στο παρακάτω παράδειγμα φαίνεται πως χρησιμοποιούμε την ιδιότητα DataSourceMode του control SqlDataSource για ένα σενάριο που δεν απαιτεί sorting, paging ή filtering.

```
ASP

<%@ Page language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">
      <asp:SqlDataSource
          id="SqlDataSource1"
          runat="server"
          DataSourceMode="DataReader"
```

```
            ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
            SelectCommand="SELECT LastName FROM Employees">
        </asp:SqlDataSource>

        <asp:ListBox
            id="ListBox1"
            runat="server"
            DataTextField="LastName"
            DataSourceID="SqlDataSource1">
        </asp:ListBox>

    </form>
  </body>
</html>
```

Κώδικας 11.31 – Ορίζοντας την ιδιότητα DataSourceMode

Εμφάνιση των δεδομένων

Για να εμφανίσουμε δεδομένα σε μια σελίδας ASP.NET χρησιμοποιούμε ένα data-bound control όπως το GridView, το DetailsView ή το FormView ή controls όπως το ListBox ή το DropDownList. Το data-bound control συμπεριφέρεται σαν ένας καταναλωτής των δεδομένων που λαμβάνει το SqlDataSource από τη βάση. Θέτουμε την ιδιότητα DataSourceID του data-bound control στο ID του SqlDataSource control. Όταν παράγεται η σελίδα, το SqlDataSource control λαμβάνει τα δεδομένα και τα διαθέτει στο data-bound control, το οποίο με τη σειρά του τα εμφανίζει. Στο επόμενο παράδειγμα φαίνεται πως εμφανίζονται τα αποτελέσματα ενός ερωτήματος σε ένα GridView.

```
ASP

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">

      <asp:SqlDataSource
          id="SqlDataSource1"
          runat="server"
          DataSourceMode="DataReader"
          ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
          SelectCommand="SELECT FirstName, LastName, Title FROM Employees">
      </asp:SqlDataSource>

      <asp:GridView
          id="GridView1"
          runat="server"
          DataSourceID="SqlDataSource1">
      </asp:GridView>

    </form>
  </body>
</html>
```

Κώδικας 11.32 – Εμφάνιση αποτελεσμάτων σε ένα GridView

11.4. GridView

Το GridView είναι ίσως το πιο αυτοματοποιημένο από τα data bound controls της ASP.NET. Παρουσιάζει τα δεδομένα του υπό την μορφή πίνακα/πλέγματος με τα δεδομένα να τοποθετούνται σε γραμμές και στήλες. Παρέχει δυνατότητα για καθορισμό header και footer, αλλαγής της εμφάνισης των στηλών μέσω των προτύπων (templates), ταξινόμηση των τιμών με βάση κάποια στήλη, σελιδοποίηση των αποτελεσμάτων καθώς και χαρακτηριστικά για αυτοματοποιημένες διαδικασίες επεξεργασίας (editing) των δεδομένων του.

Τα χαρακτηριστικά αυτά του GridView γλυτώνουν τον προγραμματιστή από ένα σημαντικό ποσοστό εισαγωγής κώδικα από την άλλη όμως δεν παρέχουν την ευελιξία και την

δυνατότητα για πλήρη παραμετροποίηση που παρέχει π.χ. ένα Repeater control το οποίο θα γνωρίσουμε στη συνέχεια.

Ένα GridView μπορεί να εισαχθεί με απλό drag 'n drop στη σελίδα από τη γραμμή εργαλείων ή να οριστεί σε μία ASP σελίδα απλά ως εξής:

```
ASP

<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
```

Κώδικας 11.33 – Ορισμός ενός GridView

Μερικές από τις σημαντικότερες ιδιότητες του GridView είναι οι εξής:

- DataKeyNames – Θέτει μία ή περισσότερες στήλες ως θεωρητικά primary keys στο GridView.
- AutoGenerateColumns – Ορίζει αν τα πεδία του GridView θα πρέπει να παράγονται αυτόματα.
- SelectedIndex – Ορίζει τη γραμμή στο GridView που θα είναι αρχικά επιλεγμένη.

11.4.1.Σύνδεση GridView με SqlDataSource

Η σημαντικότερη ίσως ιδιότητα ενός GridView είναι αυτη που το συνδέει (bind) με ένα SqlDataSource (ή κάποιο άλλο αντικείμενο DataSource) ώστε να ανακτήσει δεδομένα από μια βάση. Αυτή είναι η DataSourceID. Για να επιτευχθεί η σύνδεση μεταξύ των δύο τύπων αντικειμένων η τιμή αυτής της ιδιότητας θα πρέπει να τίθεται ίση με το id του επιθυμητού SqlDataSource.

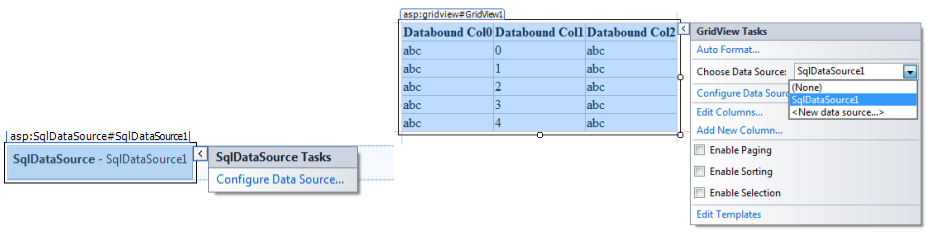
```
ASP

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:NorthWind %>"
        SelectCommand="SELECT [EmployeeID], [LastName], [FirstName], [Title]
FROM [Employees]"></asp:SqlDataSource>


<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataKeyNames="EmployeeID" DataSourceID="SqlDataSource1">
            <Columns>
                <asp:BoundField DataField="EmployeeID" HeaderText="EmployeeID"
                    InsertVisible="False" ReadOnly="True"
SortExpression="EmployeeID" />
                <asp:BoundField DataField="LastName" HeaderText="LastName"
                    SortExpression="LastName" />
                <asp:BoundField DataField="FirstName" HeaderText="FirstName"
                    SortExpression="FirstName" />
                <asp:BoundField DataField="Title" HeaderText="Title"
SortExpression="Title" />
            </Columns>
</asp:GridView>
```

Κώδικας 11.34 – Σύνδεση SqlDataSource με ένα GridView

Η διαδικασία της σύνδεσης ενός GridView με ένα DataSource μπορεί να πραγματοποιηθεί και οπτικά μέσα από το design mode του Visual Studio, και θα δημιουργήσει ακριβώς τα ίδια αποτελέσματα.



Εικόνα 11.2 – Σύνδεση SqlDataSource με ένα GridView

Έχοντας αρχικά δημιουργήσει ένα SqlDataSource και ένα GridView στη σελίδα κάνουμε κλικ στο βελάκι του GridView για να εμφανίσουμε το παράθυρο GridView Tasks. Από εκεί κάνουμε κλικ στην DropDownList με την ονομασία Choose Data Source και επιλέγουμε το SqlDataSource που μας ενδιαφέρει.

11.4.2.Σύνδεση GridView με SqlDataReader

Πέρα από τον απλό τρόπο σύνδεσης ενός GridView με ένα SqlDataSource, ένας εναλλακτικός τρόπος εισαγωγής δεδομένων σε ένα GridView είναι μέσω ενός αντικειμένου SqlDataReader. Ο τρόπος αυτός υλοποιείται μόνο προγραμματιστικά.

C#

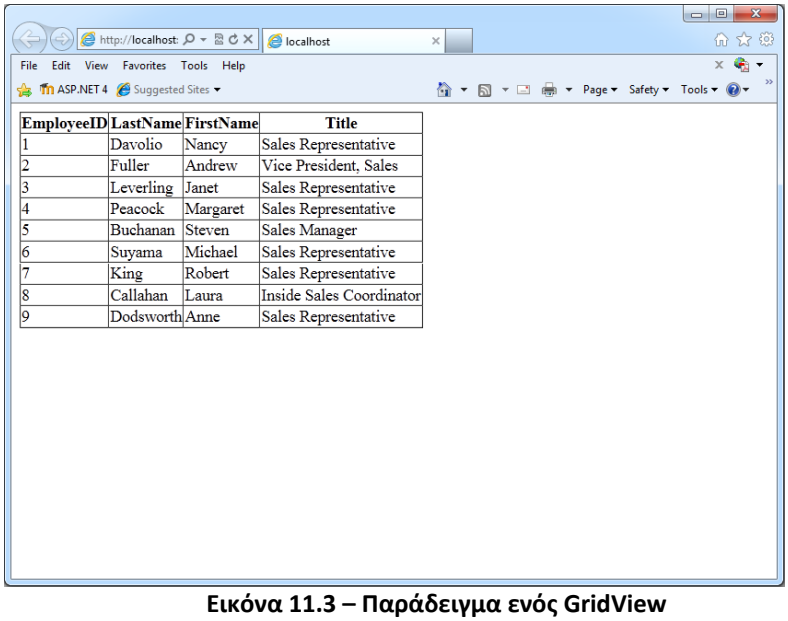
```
GridView1.DataSource = reader;
GridView1.DataBind();
```

VB

```
GridView1.DataSource = reader
GridView1.DataBind()
```

Κώδικας 11.35 – Σύνδεση ενός GridView με ένα Reader

Σε αντίθεση με άλλα data controls όπου πρέπει αρχικά να έχουμε ορίσει πρότυπα εμφάνισης των στοιχείων (γραμμές ή στήλες ανάλογα με το control), στην περίπτωση του GridView δημιουργείται αυτόματα μια βασική δομή πίνακα χωρίς να γράψουμε καθόλου κώδικα. Το GridView κατά την μετάφρασή του σε HTML κώδικα παράγει ένα HTML table, με κάθε εγγραφή να τοποθετείται σε ένα διαφορετικό tr και κάθε στήλη να τοποθετείται σε διαφορετικό td. Αυτό συμβαίνει γιατί το GridView εξορισμού εξετάζει εκ των προτέρων την δομή των δεδομένων του reader και δημιουργεί ένα πίνακα με στήλες τα πεδία που περιέχονται στα δεδομένα. Αυτή η συμπεριφορά ελέγχεται μέσω της ιδιότητας AutoGenerateColumns η οποία φέρει εξ ορισμού την τιμή true. Η μορφή του πίνακα που προκύπτει μοιάζει με τον πίνακα αποτελεσμάτων που θα λαμβάναμε τυπικά μέσα από το περιβάλλον του SQL Server.



EmployeeID	LastName	FirstName	Title
1	Davolio	Nancy	Sales Representative
2	Fuller	Andrew	Vice President, Sales
3	Leverling	Janet	Sales Representative
4	Peacock	Margaret	Sales Representative
5	Buchanan	Steven	Sales Manager
6	Suyama	Michael	Sales Representative
7	King	Robert	Sales Representative
8	Callahan	Laura	Inside Sales Coordinator
9	Dodsworth	Anne	Sales Representative

Εικόνα 11.3 – Παράδειγμα ενός GridView

11.4.3.Τύποι πεδίων

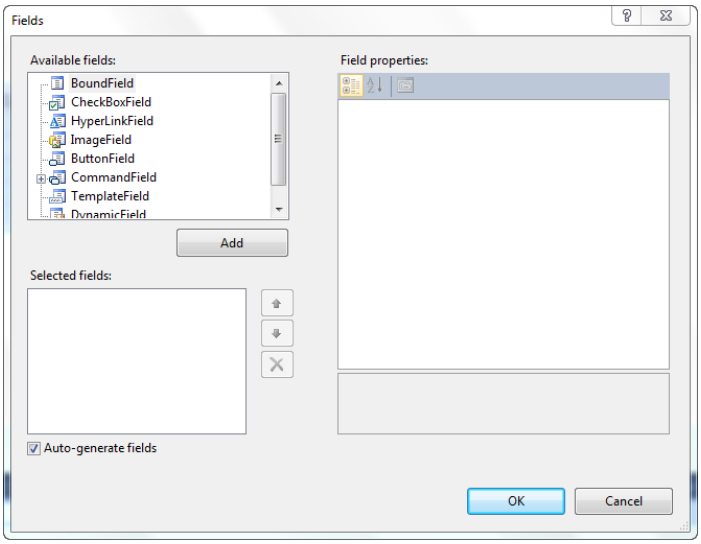
Η διαδικασία του binding εισήγαγε αυτόματα το tag Columns καθώς και ένα σύνολο από BoundField tags εντός του. Το GridView υποστηρίζει στήλες διαφορετικών τύπων. Η στήλες τύπου BoundFields επιλέγονται εξ ορισμού για πεδία που περιέχουν κείμενο ή αριθμούς. Πιο αναλυτικά οι τύποι των στηλών που υποστηρίζει ένα GridView είναι:

- BoundField - Για την απλή παρουσίαση κειμένου ή εισαγωγή κειμένου.
- ButtonField - Για την προβολή ενός κουμπιού.
- CheckBoxField - Για την προβολή ενός CheckBox. Αυτός ο τύπος χρησιμοποιείται εξ ορισμού για πεδία της βάσης που παίρνουν τιμές true/false.
- CommandField - Παρέχει μια επιλογή για κουμπιά εντολών, Insert, Update, Delete.
- HyperLinkField - Προβάλει ένα υπερσύνδεσμο.
- ImageField - Προβάλει μια εικόνα.
- TemplateField - Προβάλει ένα σύνθετο σύνολο από ενδεχομένως πολλά controls και κώδικα HTML με βάση προδιαγραφές που ορίζουμε εμείς ως ένα πρότυπο (template).

Το BoundField είναι ο πιο διαδεδομένος τύπος στήλης στα GridView. Οι βασικές ιδιότητες ενός BoundField είναι:

- DataField - Ορίζει το πεδίο από τα δεδομένα που επιστρέφει το ερώτημα που θα εμφανίζει η στήλη.
- ReadOnly - Ορίζει αν η τιμή του συγκεκριμένου πεδίου θα μπορεί να αλλάξει αν η συγκεκριμένη γραμμή του GridView έρθει σε edit mode.
- InsertVisible - Ορίζει αν θα υπάρχει η δυνατότητα εισαγωγής δεδομένων για το συγκεκριμένο πεδίο κατά το insert mode.
- NullDisplayText - Ορίζει το κείμενο που θα εμφανίζεται για εγγραφές με τιμές null.
- HeaderText - Ορίζει το κείμενο της κεφαλίδας για το συγκεκριμένο πεδίο.
- FooterText - Ορίζει το κείμενο του υποσέλιδου για το συγκεκριμένο πεδίο.

Πέραν της χειροκίνητης εισαγωγής πεδίων και ιδιοτήτων, υπάρχει η δυνατότητα να εργαστούμε με τις στήλες ενός GridView οπτικά. Κάνοντας κλικ στο βελάκι του Gridview εμφανίζεται το παράθυρο με την ονομασία GridView Tasks. Επιλέγοντας το Edit Columns… εμφανίζεται ένα παράθυρο το οποίο μας επιτρέπει να τροποποιήσουμε ιδιότητες στηλών, να καταργήσουμε στήλες να αλλάξουμε τη σειρά εμφάνισής τους καθώς επίσης και να προσθέσουμε νέες.

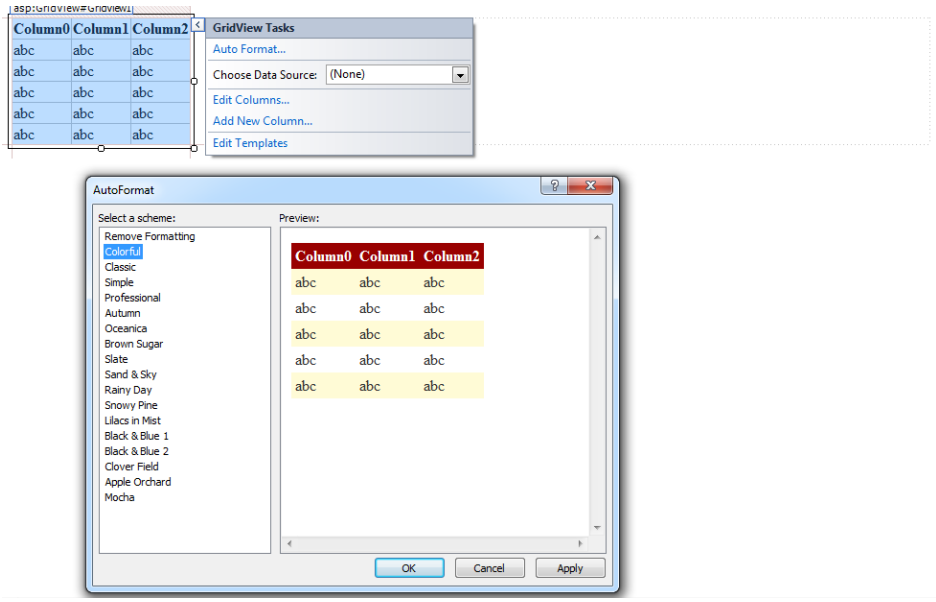


Εικόνα 11.4 – Προσθήκη ή τροποποίηση στηλών σε ένα GridView

11.4.4.Μορφοποίηση

Το αποτέλεσμα που προκύπτει από τις διαδικασίες που περιγράφτηκαν παραπάνω ελάχιστες φορές θα ανταποκρίνεται στις αισθητικές απαιτήσεις μιας εφαρμογής. Φυσικά το GridView δίνει τη δυνατότητα για εκτεταμένη αλλαγή της εμφάνισής του.

Μπορούμε οπτικά να τροποποιήσουμε την εμφάνιση ενός GridView στο design mode της σελίδας. Πιο συγκεκριμένα, κάνοντας κλικ στο βελάκι που αναδύει το menu με τα GridView Tasks και στη συνέχεια κάνοντας κλικ στην επιλογή Auto Format… μας δίδεται η δυνατότητα να επιλέξουμε ένα από τα πολλά προκαθορισμένα σχέδια όπως φαίνεται στο σχήμα χχχ. Ακόμη και έτσι όμως, σπάνια κάποια προκαθορισμένα σχήματα σχέδια μπορούν να ταιριάξουν απόλυτα στην εκάστοτε εφαρμογή.



Εικόνα 11.5 – Μορφοποίηση ενός GridView με προεπιλεγμένα σχέδια

Για πιο λεπτομερή καθορισμό του παρουσιαστικού ενός GridView μπορούμε να βασιστούμε σε μια σειρά από styles. Πιο συγκεκριμένα υπάρχουν:

- HeaderStyle - Ορίζει το παρουσιαστικό της κεφαλίδας του GridView.
- RowStyle - Ορίζει το παρουσιαστικό των γραμμών του GridView.
- AlternatingRowStyle - Ορίζει το παρουσιαστικό κάθε δεύτερης γραμμής του GridView.
- SelectedRowStyle - Ορίζει το παρουσιαστικό της γραμμής που είναι επιλεγμένη.
- EditRowStyle - Ορίζει το παρουσιαστικό της γραμμής που βρίσκεται σε edit mode.
- EmptyDataRowStyle - Ορίζει το παρουσιαστικό της γραμμής όταν αυτή είναι κενή δηλαδή κατά την περίπτωση που η εντολή SELECT δεν επιστρέφει δεδομένα.
- FooterStyle - Ορίζει το παρουσιαστικό του υποσέλιδου αν αυτό φυσικά έχει οριστεί να είναι ορατό.
- PagerStyle - Ορίζει το παρουσιαστικό της γραμμής που περιέχει τα κουμπιά για τη μετάβαση σε άλλες σελίδες αποτελεσμάτων του GridView.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η περίπτωση του TemplateField το οποίο έμμεσα μπορεί να καθορίσει σε μεγάλο βαθμό το παρουσιαστικό ολόκληρου του GridView. Ένα TemplateField επιτρέπει τον ορισμό ενός προτύπου (template) μιας στήλης. Στο πρότυπο αυτό μπορούμε να ορίσουμε ASP.NET controls και στοιχεία HTML δημιουργώντας έτσι σύν-

θετα κελιά που μπορεί να περιέχουν ακόμα και τιμές από πολλαπλά πεδία. Ένα TemplateField βασίζεται σε ειδικά tags για τον καθορισμό ενός προτύπου της στήλης όταν αυτή βρίσκεται σε συγκεκριμένες καταστάσεις π.χ. βρίσκεται σε edit mode.

Πιο συγκεκριμένα διακρίνουμε τα εξής tags:

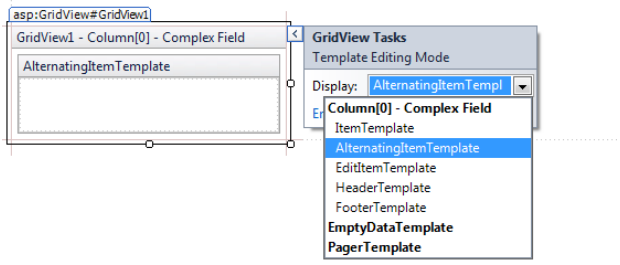
- HeaderTemplate - Ορίζει την εμφάνιση και το περιεχόμενο της κεφαλίδας μιας στήλης.
- FooterTemplate - Ορίζει την εμφάνιση και το περιεχόμενο του υποσέλιδου μιας στήλης.
- ItemTemplate - Ορίζει την εμφάνιση και το περιεχόμενο κάθε κελιού μιας στήλης ή αν χρησιμοποιείται το AlternatingItemTemplate κάθε κελιού περιττού αριθμού.
- AlternatingItemTemplate - Ορίζει την εμφάνιση και το περιεχόμενο κάθε δεύτερου κελιού μιας στήλης, με άλλα λόγια αναφέρεται στα κελιά άρτιου αριθμού.
- EditItemTemplate - Ορίζει την εμφάνιση και το περιεχόμενο των κελιών μιας στήλης όταν αυτά βρίσκονται σε edit mode. Μέσω αυτού μπορεί π.χ. να εισαχθεί validation στην εισαγωγή των δεδομένων σε κελιά ένα χαρακτηριστικό που εξ ορισμού δεν υπάρχει.
- InsertItemTemplate - Ορίζει την εμφάνιση και το περιεχόμενο των κελιών μιας στήλης όταν σε αυτά εισάγουμε μια νέα εγγραφή.
- PagerTemplate - Ορίζει την εμφάνιση των controls που αναφέρονται στο paging. Δεν αναφέρεται σε μια συγκεκριμένη στήλη.
- EmptyDataTemplate - Ορίζει την εμφάνιση των κελιών όταν δεν υπάρχουν δεδομένα. Δεν αναφέρεται σε μια συγκεκριμένη στήλη.

ASP

```
<asp:TemplateField HeaderText="Complex">
   <ItemTemplate>
        <table border="1">
           <tr>
              <td>
                 <%# Eval("LastName") %>
               </td>
                <td>
                   <%# Eval("FirstName") %>
                </td>
            </tr>
             <tr>
                <td colspan="2">
                    <%# Eval("Title") %>
                 </td>
              </tr>
         </table>
    </ItemTemplate>
</asp:TemplateField>
```

Κώδικας 11.36 - Ορισμός ενός TemplateField σε ένα GridView

Φυσικά υπάρχει η δυνατότητα να ορίσουμε τα περιεχόμενα μιας στήλης template οπτικά στο design mode μιας σελίδας χωρίς την εισαγωγή κώδικα. Για να κάνουμε κάτι τέτοιο θα πρέπει οπωσδήποτε το GridView να έχει τουλάχιστον μια στήλη που θα περιέχει ένα TemplateField. Αρχικά θα πρέπει να κάνουμε κλικ στο βελάκι του GridView ώστε να αναδυθεί το παράθυρο GridView Tasks. Σε αυτό κάνουμε την επιλογή Edit Templates. Κάνοντας κλικ για ακόμα μια φορά στο βελάκι μπορούμε να θέσουμε το παρουσιαστικό για μια εκ των προαναφερθέντων τύπων Template μέσω της DropDownList με το όνομα Display. Στη συνέχεια μπορούμε να εισάγουμε τα controls της επιλογής μας από το toolbox δομώντας κατ' αυτό τον τρόπο το πρότυπο της αρεσκείας μας. Τέλος μπορούμε ανά πάσα στιγμή να τερματίσουμε την συγγραφή της template πατώντας End Template Editing.



Εικόνα 11.6 – Δόμηση μιας Template ενός GridView

11.4.5.Ταξινόμηση

Μέσω της διαδικασίας του sorting είναι δυνατόν να ταξινομηθούν οι γραμμές ενός GridView κατά συγκεκριμένες στήλες.

Για να ενεργοποιηθεί το συγκεκριμένο χαρακτηριστικό αρκεί η ιδιότητα AllowSorting του GridView να τεθεί σε true.

ASP

```
<asp:BoundField DataField="LastName" HeaderText="LastName" SortExpres-
sion="LastName" />
```

Κώδικας 11.37 – Ορισμός ενός BoundField σε ένα GridView

Αφού ενεργοποιηθεί το sorting για ένα GridView οι κεφαλίδες για τα αντίστοιχα πεδία θα μετατραπούν σε υπερσυνδέσμους οι οποίοι κατά το κλικ θα ταξινομούν τις εγγραφές πρώτα κατά αύξουσα τιμή για τη συγκεκριμένη στήλη. Κατά το δεύτερο κλικ η ταξινόμηση γίνεται κατά φθίνουσα σειρά.

Η διαδικασία του sorting θα μπορούσε να επιτευχθεί προγραμματιστικά καλώντας τη μέθοδο Sort του GridView. Π.χ. αν θέλουμε με το πάτημα ενός κουμπιού να ταξινομήσουμε ένα GridView κατά το πεδίο LastName, κατά αύξουσα σειρά τότε εντός του event handler onClick για το συγκεκριμένο κουμπί θα γράφαμε:

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
        GridView1.Sort("LastName", SortDirection.Ascending);
}
```

VB

```
Protected Sub Button1 Click(sender As Object, e As System.EventArgs) Handles
Button1.Click

     GridView1.Sort("LastName", SortDirection.Ascending)

End Sub
```

Κώδικας 11.38 – Ταξινόμηση ενός GridView κατά μια στήλη

11.4.6.Σελιδοποίηση

Μέσω του μηχανισμού της σελιδοποίησης (paging) τα αποτελέσματα που θα παρουσιάζονταν κανονικά σε ένα GridView σπάνε σε σελίδες προκαθορισμένου πλήθους εγγραφών. Ανά πάσα στιγμή μια από τις σελίδες παρουσιάζεται στο χρήστη ενώ παρέχονται μηχανισμοί ώστε ο χρήστης να περιηγηθεί και στις υπόλοιπες σελίδες των αποτελεσμάτων. Ο μηχανισμός του paging είναι χρήσιμος σε περιπτώσεις όπου τα αποτελέσματα του ερωτήματος αναμένεται να είναι πολλά σε πλήθος.

Οι βασικές ιδιότητες ενός GridView που αφορούν το paging είναι:

- AllowPaging - Ενεργοποιεί ή απενεργοποιεί το paging για το GridView. Εξ ορισμού η τιμή του είναι false.

- PageSize - Θέτει τον αριθμό των αντικειμένων που θα εμφανίζονται σε μια σελίδα. Η τιμή εξ ορισμού είναι 10.
- PageIndex - Θέτει ή επιστρέφει το δείκτη της σελίδας που προβάλλεται αυτή τη στιγμή στο GridView.
- PagerSettings – Παρέχει μια ποικιλία από επιλογές σχετικές με την εμφάνιση των controls που αφορούν τη σελιδοποίηση.
- PagerStyle - Παρέχει έξτρα επιλογές για τον καθορισμό του στυλ των controls που αφορούν το paging. Οι επιλογές μεταξύ άλλων περιλαμβάνουν τη γραμματοσειρά, το χρώμα και τη στοίχιση του κειμένου για αυτά τα controls.

Για να ενεργοποιηθεί η αυτόματη διαδικασία της σελιδοποίησης το μόνο που έχουμε να κάνουμε είναι να θέσουμε την τιμή της ιδιότητας AllowPaging σε true. Αν δεν θέσουμε την ιδιότητα PageSize τότε ισχύει η εξ ορισμού τιμή που είναι 10 εγγραφές ανά σελίδα. Να σημειώσουμε ότι το αυτόματο paging δεν είναι δυνατόν όταν το GridView έχει γίνει bind με ένα αντικείμενο Reader.

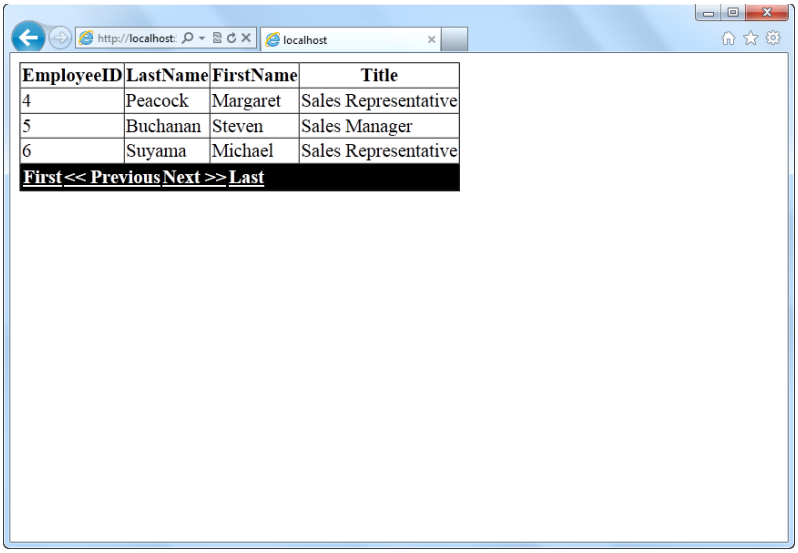
Σε αυτό το σημείο καλό θα είναι να διευκρινίσουμε ότι το paging δεν μειώνει τον αριθμό των στοιχείων που επιστρέφει το query επομένως ο φόρτος για τη βάση είναι ο ίδιος και κάθε φορά γίνονται bind στο GridView όλες οι εγγραφές του ερωτήματος.

Τα controls που αφορούν το paging μπορούν να μορφοποιηθούν μέσω των ιδιοτήτων PagerStyle καθώς και PagerSettings. Η PagerStyle χρησιμοποιείται για να καθορίσει χαρακτηριστικά όπως το χρώμα, το μέγεθος και τη γραμματοσειρά των controls ενώ το PagerSettings καθορίζει μεταξύ άλλων τον τύπο των controls (οι πιθανές τιμές είναι Numeric, NumericFirstLast, NextPrevious, NextPreviousFirstLast) που θα εμφανίζονται και πιθανώς το κείμενό τους.

```
ASP

<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1" AllowPaging="true" PageSize="3">
            <PagerStyle BackColor="Black" ForeColor="White" Font-Bold="true" />
            <PagerSettings Mode="NextPreviousFirstLast" NextPageText="Next >>" PreviousPageText="<< Previous" LastPageText="Last" FirstPageText="First" />
</asp:GridView>
```

Κώδικας 11.39 – Ορισμός στυλ και ρυθμίσεων του paging control



Εικόνα 11.7 – Παράδειγμα paging control

11.4.7.CommandField

Το GridView παρέχει τη δυνατότητα εκτέλεσης πολύπλοκων διαδικασιών όπως εισαγωγή, ενημέρωση, διαγραφή δεδομένων με τον εξαιρετικά απλό μηχανισμό του CommandField. Πεδία τύπου CommandField διαθέτουν ιδιότητες όπως ShowSelectButton, ShowInsertButton, ShowUpdateButton, ShowCancelButton, ShowDeleteButton η οποίες όταν τίθενται σε true εμφανίζουν το αντίστοιχο κουμπί στη στήλη. Τέτοια κουμπιά όταν πατηθούν

θέτουν τη συγκεκριμένη γραμμή σε ανάλογη κατάσταση άρα εμφανίζεται η αντίστοιχη Template για τη γραμμή (εφόσον αυτή υπάρχει) και εκτελείται η αντίστοιχη εντολή από το SqlDataSource (εφόσον το GridView έχει συνδεθεί με ένα SqlDataSource).

Στο παρακάτω παράδειγμα με το κλικ του κουμπιού Select η γραμμή λαμβάνει κίτρινο background με βάση το στυλ που έχει οριστεί για τις επιλεγμένες γραμμές. Χωρίς καν να έχουμε ορίσει μια template για την περίπτωση που η γραμμή είναι σε edit mode, όταν πατάμε Edit τότε αυτομάτως εμφανίζονται textboxes για τα πεδία της συγκεκριμένης γραμμής, επιτρέποντάς μας να τροποποιήσουμε τις τιμές των πεδίων. Επιπρόσθετα εμφανίζονται τα κουμπιά Update και Cancel.

```
ASP

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\NORTHWND.MDF;Integrated
Security=True;Connect Timeout=30;User Instance=True"
            ProviderName="System.Data.SqlClient"
            SelectCommand="SELECT [EmployeeID], [LastName], [FirstName], [Ti-
tle] FROM [Employees]"
            UpdateCommand="UPDATE Employees SET LastName=@LastName, First-
Name=@FirstName, Title=@Title WHERE EmployeeID=@EmployeeID"
            DeleteCommand="DELETE FROM Persons WHERE EmployeeID=@EmployeeID ">
        </asp:SqlDataSource>

        <asp:GridView ID="GridView1" runat="server" Data-
SourceID="SqlDataSource1" AutoGenerateColumns="true" DataKeyNames="EmployeeID"
>
            <Columns>
                <asp:CommandField HeaderText="Commands" ShowDeleteButton="True"
ShowEditButton="True" ShowHeader="True" ShowSelectButton="True" />
            </Columns>
            <SelectedRowStyle BackColor="Yellow" />
        </asp:GridView>
```

Κώδικας 11.40 – Ορισμός ενός command field για επεξεργασία εγγραφών

11.4.8.Επεξεργασία χωρίς CommandField

Συνήθως όταν επιθυμούμε ένα GridView να έχει δυνατότητες εισαγωγής, τροποποίησης και διαγραφής στοιχείων απλά προσθέτουμε μια στήλη τύπου CommandButton. Ωστόσο από τη στιγμή που το GridView υποστηρίζει templates η προσέγγιση του CommandButton ενδεχομένως να μην είναι πάντα η βέλτιστη. Σε ορισμένες περιπτώσεις θα πρέπει ο προγραμματιστής να ορίζει δικά του controls τα οποία θα έχουν ανάλογη συμπεριφορά.

Η διαδικασία αυτή σε ένα GridView είναι αρκετά απλή. Αρκεί να υπάρχει ένα Button με το κατάλληλο CommandName π.χ. Edit, Update, Cancel, Delete για τις αντίστοιχες διαδικασίες.

```
ASP

<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1"
            AllowPaging="True" PageSize="3">
            <PagerStyle BackColor="Black" ForeColor="White" Font-Bold="true" />
            <Columns>
                <asp:TemplateField HeaderText="Command">
                    <ItemTemplate>
                        <asp:Button ID="Button1" runat="server" Text="Edit"
CommandName="Edit" />
                    </ItemTemplate>
                    <EditItemTemplate>
                        <asp:Button ID="Button1" runat="server" Text="Save"
Width="100" CommandName="Update" />
                        <br />
                         <asp:Button ID="Button2" runat="server" Text="Don't
Save" Width="100" CommandName="Cancel" />
                    </EditItemTemplate>
                </asp:TemplateField>
            </Columns>
            <PagerSettings Mode="NextPreviousFirstLast" NextPageText="Next >>"
PreviousPageText="<< Previous" LastPageText="Last" FirstPageText="First" />
</asp:GridView>
```

Κώδικας 11.41 – Επεξεργασία εγγραφών μέσω απλών κουμπιών

11.5. Repeater

Το Repeater control επιτρέπει την παρουσίαση δεδομένων από μια πηγή με μορφή στοιχείων επαναλαμβανόμενων δομής. Παρέχει τη δυνατότητα ο προγραμματιστής να ορίσει ένα πρότυπο εμφάνισης των αντικειμένων της λίστας. Αν και είναι το πιο απλό από τα data bound controls είναι το πιο ισχυρό καθώς επιτρέπει εκτεταμένη παραμετροποίηση και κατά συνέπεια με την ανάλογη εργασία μπορεί να παράγει πολύπλοκα αποτελέσματα.

Η πιο απλή μορφή ενός Repeater είναι η εξής:

```
ASP

<asp:Repeater ID="Repeater1" runat="server">
   <ItemTemplate>
   </ItemTemplate>
</asp:Repeater>
```

Κώδικας 11.42 – Ορισμός ενός Repeater

11.5.1.Έλεγχος Εμφάνισης

Όπως και το GridView έτσι και το Repeater control ορίζει ένα σύνολο από στοιχεία τα οποία χρησιμοποιούνται για να καθορίσουν το παρουσιαστικό των αντικειμένων που αυτό προβάλει. Τα στοιχεία αυτά καθορίζουν στην ουσία πρότυπα (templates) εμφάνισης εγγραφών.

Το στοιχείο ItemTemplate είναι το μόνο υποχρεωτικό στοιχείο ενός Repeater. Εντός αυτού ορίζεται συνήθως ένα κομμάτι HTML κώδικα (μπορεί να περιλαμβάνονται και ASP controls) το οποίο καθορίζει την εμφάνιση κάθε στοιχείου του Repeater. Ο κώδικας εντός του ItemTemplate θα επαναλαμβάνεται για κάθε στοιχείο που πρόκειται να παρουσιαστεί.

Ομοίως ένα στοιχείο AlternatingItemTemplate αν έχει οριστεί εμφανίζει τα περιεχόμενά του για κάθε δεύτερη εγγραφή αντί του ItemTemplate. Είναι χρήσιμο όταν θέλουμε να εισάγουμε μια διαφοροποίηση στην εμφάνιση των αποτελεσμάτων, στοιχείο παρά στοιχείο, για αισθητικούς κυρίως λόγους.

Το στοιχείο HeaderTemplate σε αντίθεση με τα στοιχεία ItemTemplate δεν επαναλαμβάνεται. Αν οριστεί από το χρήστη εμφανίζεται μόνο μια φορά στην κορυφή των αποτελεσμάτων ως κεφαλίδα του control. Είναι χρήσιμο για να εμφανίζουμε τον τίτλο των αποτελεσμάτων.

Το στοιχείο FooterTemplate είναι παρόμοιο με ένα HeaderTemplate όμως εμφανίζεται στο κάτω μέρος των αποτελεσμάτων του control.

Το στοιχείο SeperatorTemplate επιτρέπει τον ορισμό κώδικα που θα προβάλλεται μεταξύ των ItemTemplate. Επαναλαμβάνεται για κάθε εγγραφή στο set των αποτελεσμάτων ενός ερωτήματος εκτός της πρώτης και της τελευταίας.

Έτσι ένα απλό παράδειγμα ενός Repeater είναι το εξής:

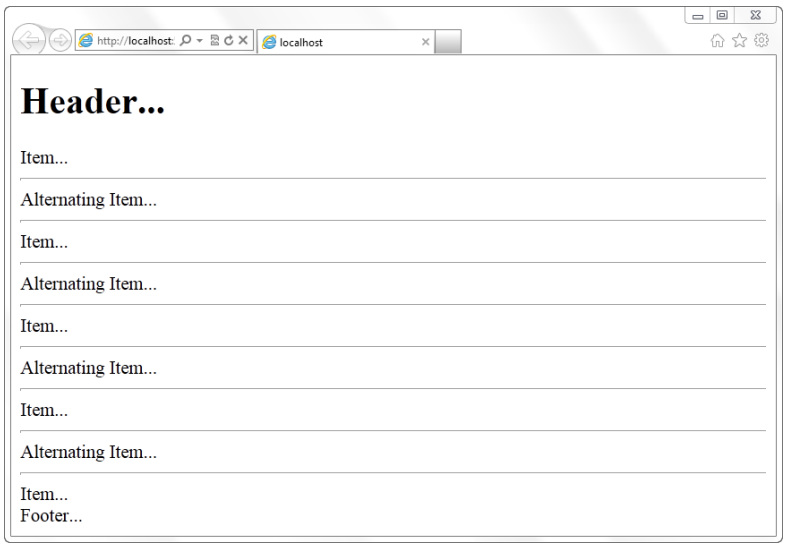
```
ASP

<asp:Repeater ID="Repeater1" runat="server">
            <HeaderTemplate>
                <div><h1>Header...</h1></div>
            </HeaderTemplate>
            <ItemTemplate>
                <div>Item...</div>
            </ItemTemplate>
            <AlternatingItemTemplate>
                <div>Alternating Item...</div>
            </AlternatingItemTemplate>
            <SeparatorTemplate>
                <hr />
            </SeparatorTemplate>
            <FooterTemplate>
                <div>Footer...</div>
            </FooterTemplate>
</asp:Repeater>
```

Κώδικας 11.43 – Repeater με διάφορα στοιχεία template

11.5.2.Παρουσίαση δεδομένων

Στο παραπάνω παράδειγμα για κάθε εγγραφή που υπάρχει στο σετ των αποτελεσμάτων θα επαναληφθεί ο κώδικας που βρίσκεται εντός ItemTemplate και για κάθε δεύτερο στοιχείο ο κώδικας που βρίσκεται εντός του AlternatingItemTemplate. Αν π.χ. το σύνολο των δεδομένων περιλάμβανε 9 εγγραφές θα λαμβάναμε ως αποτέλεσμα αυτό που φαίνεται στην εικόνα χχχ :



Εικόνα 11.8 – Παράδειγμα ενός Repeater

Θα πρέπει να διευκρινίσουμε ότι για να είμαστε σε θέση να δούμε δεδομένα σε αυτό το control θα πρέπει πάντα να προβαίνουμε σε δύο βασικές ενέργειες.

- Να καθορίσουμε την πηγή απ' όπου θα αντλεί δεδομένα αυτό το control. Συνήθως σε αυτές τις περιπτώσεις τα δεδομένα προκύπτουν από ένα απλό SQL select ερώτημα.
- Να εκτελέσουμε τη διαδικασία του Databind που στην ουσία δένει συγκεκριμένο Data source με το control.

Αυτή η διαδικασία είναι παρόμοια με την αντίστοιχη ενός GridView που περιγράφτηκε παραπάνω. Έτσι ένας Repeater (όπως και τα υπόλοιπα data control) μπορεί να λάβει ως DataSource αντικείμενα διαφορετικού τύπου όπως π.χ. ένα SqlDataReader ή ένα SqlDataSource.

Από τη στιγμή που ένας Repeater δεν έχει τους αυτόματους μηχανισμούς εμφάνισης δεδομένων που διαθέτει ένα GridView αποκτά ιδιαίτερο ενδιαφέρον να δούμε πως γίνεται η προβολή περιεχομένου που προέρχεται από μια βάση σε ένα Repeater.

Ένας απλός τρόπος είναι μέσω της μεθόδου Eval η οποία παίρνει σαν παράμετρο το όνομα του πεδίου και επιστρέφει την τιμή του πεδίου αυτού στην τρέχουσα εγγραφή. Στη συγκεκριμένη περίπτωση είναι ευκολότερο να καλούμε τη συγκεκριμένη μέθοδο από τον κώδικα της ASP και όχι το code behind για να έχουμε καλύτερο έλεγχο της ακριβούς θέσης που πρόκειται να εμφανιστεί κάθε πεδίο. Η ακολουθίες <%# και %> είναι απαραίτητο να υπάρχουν αφού επιχειρούμε να τρέξουμε μια μέθοδο από κώδικα ASP και όχι από το code behind.

ASP

```
<asp:Repeater ID="Repeater1" runat="server">
            <HeaderTemplate>
                <div><h1>Employees</h1></div>
            </HeaderTemplate>
            <ItemTemplate>
                <div><span><%# Eval("LastName")%>, </span><span><%#
Eval("FirstName")%>: </span><span>   <%# Eval("Title")%></span></div>
```

```
            </ItemTemplate>
            <AlternatingItemTemplate>
                <div><span><%# Eval("LastName")%>, </span><span><%#
Eval("FirstName")%>: </span><span>   <%# Eval("Title")%></span></div>
            </AlternatingItemTemplate>
            <SeparatorTemplate>
                <hr />
            </SeparatorTemplate>
</asp:Repeater>
```

Κώδικας 11.44 - Repeater με κλήση της μεθόδου Eval

11.6. DataList

Η DataList είναι ένα control για την παρουσίαση δεδομένων, με χαρακτηριστικά που συνδυάζουν αυτά ενός Repeater και ενός GridView. Η βασικότερη διαφορά μεταξύ μιας DataList και ενός Repeater έγκειται στο ότι το DataList παρέχει εξ ορισμού μια έτοιμη template για την παρουσίαση των δεδομένων. Πιο συγκεκριμένα ένα DataList μεταφράζεται σε ένα πίνακα το οποίο έχει ένα μόνο κελί. Με άλλα λόγια ένα DataList που προβάλει 2 εγγραφές θα παράγει εξ ορισμού τον ακόλουθο HTML κώδικα:

```
<table>
<tr>
<td>
        τιμή πεδίου 1
        τιμή πεδίου 2
        τιμή πεδίου 3
        ...
</td>
</tr>

<tr>
<td>
        τιμή πεδίου 1
        τιμή πεδίου 2
        τιμή πεδίου 3
        ...
</td>
</tr>
</table>
```

Κώδικας 11.45 – HTML κώδικας που παράγεται από μια DataList

Μια Datalist με παρόμοια μορφή με αυτή του GridView που περιγράψαμε παραπάνω είναι και η εξής:

```
ASP

<asp:DataList ID="DataList1" runat="server">
            <HeaderTemplate>
                    <div class="header_row">
                    <asp:Label id="Label2" runat="server"  Width="100" >Employ-
eeID</asp:Label>
                    <asp:Label id="lblLName" runat="server"  Width="100" >Last
Name</asp:Label>
                    <asp:Label id="lblFName" runat="server" Width="100" >First
Name</asp:Label>
                    <asp:Label id="lblTitle" runat="server"
Width="100">Title</asp:Label>
                    </div>
            </HeaderTemplate>
            <ItemTemplate>
                  <div class="normal_row">
                    <asp:Label id="Label1" runat="server"  Width="100"><%#
Eval("EmployeeID")%></asp:Label>
                    <asp:Label id="lblLName" runat="server"  Width="100"><%#
Eval("LastName")%></asp:Label>
                    <asp:Label id="lblFName" runat="server" Width="100"><%#
Eval("FirstName")%></asp:Label>
                    <asp:Label id="lblTitle" runat="server"  Width="100"><%#
Eval("Title")%></asp:Label>
                   </div>
            </ItemTemplate>
            <AlternatingItemTemplate>
```

```
                <div class="alternative_row">
                  <asp:Label id="Label1" runat="server"  Width="100"><%#
Eval("EmployeeID")%></asp:Label>
                  <asp:Label id="lblLName" runat="server" Width="100"><%#
Eval("LastName")%></asp:Label>
                  <asp:Label id="lblFName" runat="server"  Width="100"><%#
Eval("FirstName")%></asp:Label>
                  <asp:Label id="lblTitle" runat="server" Width="100"><%#
Eval("Title")%></asp:Label>
                 </div>
            </AlternatingItemTemplate>
</asp:DataList>
```

Κώδικας 11.46 – Παράδειγμα μιας DataList

Στο παραπάνω παράδειγμα ίσως παρατηρήσετε ότι για κάθε μια από τις Template έχουμε εισάγει Label controls. Αυτό έγινε για να τονίσουμε ένα γεγονός που φαινομενικά είναι παράδοξο. Αν παρατηρήσετε προσεκτικά σε κάθε template το ID του πρώτου Label (εμφανίζει το ID του υπαλλήλου) είναι το ίδιο. Το ίδιο ισχύει και για τα υπόλοιπα labels. Το γεγονός ότι επιτρέπονται στοιχεία με το ίδιο ID μεταξύ διαφορετικών στοιχείων template είναι διότι κατά τη μετάφρασή τους στον αντίστοιχο HTML κώδικα τα στοιχεία αυτά λαμβάνουν διαφορετικό ID. Πιο συγκεκριμένα, το ID που λάβουν τα controls είναι πάντα της μορφής: ΌνομαDatalist_ΌνομαΣτοιχείου_ΑριθμόςΕγγραφής.

Σε ένα DataList για να μπορέσει ένα συγκεκριμένο στοιχείο της λίστας μας να μεταβεί από μια κατάσταση σε μια άλλη (π.χ. σε edit mode ή σε selected mode) θα πρέπει να προφανώς να συμπεριλάβουμε και ένα κουμπί (που θα πραγματοποιεί αυτή τη μετάβαση), στο template του κάθε στοιχείου. Τίθεται όμως το εξής ζήτημα: Πως είναι δυνατόν να γράψουμε κώδικα που θα εκτελείται στο πάτημα ενός κουμπιού μιας συγκεκριμένης εγγραφής στη λίστα από τη στιγμή που δεν μπορούμε εκ των προτέρων να γνωρίζουμε το όνομα του κουμπιού;

Όταν ένα κουμπί εντός της λίστας πατηθεί παράγεται ένα Click Event καθώς και ένα Command event για το κουμπί αυτό όπως είναι αναμενόμενο. Επιπρόσθετα όμως παράγεται και το ItemCommand event για τη λίστα. Μέσα από τις παραμέτρους αυτού του event μπορούμε να πάρουμε διάφορες πληροφορίες για το συγκεκριμένο συμβάν μεταξύ των οποίων το όνομα του control το οποίο είναι υπεύθυνο για την πυροδότηση αυτού του event.

Ας δούμε όμως όσα περιγράφηκαν παραπάνω στην πράξη. Αρχικά θα πρέπει να τροποποιήσουμε τη DataList του προηγούμενου παραδείγματος ώστε κάθε ItemTemplate, AlternatingItemTemplate να περιλαμβάνει ένα κουμπί. Όταν πατιέται ένα τέτοιο κουμπί θα πρέπει το συγκεκριμένο στοιχείο της λίστας να μεταβαίνει σε edit mode.

Όταν ένα στοιχείο μιας DataList μεταβεί σε EditMode τότε εμφανίζεται ο κώδικας που ορίζεται εντός του στοιχείου EditItemTemplate αντί του ItemTemplate, AlternatingItemTemplate. Έτσι θα πρέπει να ορίσουμε το αντίστοιχο πρότυπο. Προφανώς αφού επιδιώκουμε να κάνουμε μετατροπή δεδομένων το πρότυπο αυτό θα πρέπει να περιλαμβάνει TextBox αντί για Label σε όσα πεδία επιθυμούμε να κάνουμε μετατροπές. Επίσης καλό θα είναι να εμφανίζονται δυο κουμπιά Save και Cancel. Το πρώτο θα χρησιμοποιείται για να αποθηκεύει τα δεδομένα εκτελώντας την κατάλληλη εντολή update και έπειτα να φέρνει το συγκεκριμένο αντικείμενο της λίστας πίσω σε normal mode. Το δεύτερο κουμπί για να φέρνει το συγκεκριμένο αντικείμενο της λίστας σε normal mode χωρίς να κάνει εκτελέσει οποιαδήποτε εντολή στη βάση.

Εντός του event handler για το ItemCommand αρχικά θα πρέπει να ελέγχουμε το είδος του command που πυροδότησε αυτό το συμβάν. Το command που μας ενδιαφέρει είναι το update. Για την περίπτωση που όντως έχει προκύψει ένα update θα πρέπει αρχικά να θέσουμε το συγκεκριμένο αντικείμενο της λίστας σε κατάσταση Edit. Πρώτα θα πρέπει να ανακτήσουμε το δείκτη του στοιχείου της λίστας (αριθμό της γραμμής) που παρήγαγε το event. Αυτό το παίρνουμε από το ίδιο το event. Στη συνέχεια θέτουμε την ιδιότητα EditItemIndex ίση με το δείκτη αυτό. Τέλος θα πρέπει να επανακτήσουμε τα δεδομένα του ερωτήματος από τη βάση και να τα συνδέουμε με το DataList να θέλουμε τα πεδία των TextBoxes να έχουν αρχική τιμή.

C#

```
protected void DataList1_ItemCommand1(object source, DataListCommandEventArgs
e)
     {
         if (e.CommandName == "update")
         {
             DataList1.EditItemIndex = e.Item.ItemIndex;
             Load Data into List();
         }
         else if (e.CommandName == "save")
         {
             SqlConnection myConnection = new SqlConnection();
             myConnection.ConnectionString = WebConfigurationManag-
er.ConnectionStrings["NorthwindDB"].ConnectionString;

             try
             {
                 myConnection.Open();
                 SqlCommand myCommand = new SqlCommand();
                 myCommand.Connection = myConnection;
                 myCommand.CommandText = "UPDATE Employees SET LastName
=@LastName, FirstName=@FirstName, Title=@Title WHERE EmployeeID=@EmployeeID";
                 myCommand.Parameters.AddWithValue("@LastName", ((Text-
Box)(e.Item.FindControl("tbLName"))).Text);
                 myCommand.Parameters.AddWithValue("@FirstName", ((Text-
Box)(e.Item.FindControl("tbFName"))).Text);
                 myCommand.Parameters.AddWithValue("@Title", ((Text-
Box)(e.Item.FindControl("tbTitle"))).Text);
                 myCommand.Parameters.AddWithValue("@EmployeeID", ((La-
bel)(e.Item.FindControl("lblID"))).Text);
                 int result = myCommand.ExecuteNonQuery();

             }
             catch (Exception ex)
             {
             }
             finally
             {
                 myConnection.Close();
             }
             DataList1.EditItemIndex = -1;

         }
         else if (e.CommandName == "cancel")
         {
             DataList1.EditItemIndex = -1;
         }

         Load Data into List();
     }
```

VB

```
Protected Sub DataList1_ItemCommand(ByVal source As Object, ByVal e As Sys-
tem.Web.UI.WebControls.DataListCommandEventArgs) Handles DataList1.ItemCommand
        If e.CommandName = "update" Then
            DataList1.EditItemIndex = e.Item.ItemIndex
            Dim connectionString As String = ConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString
            Dim strCommand As String = "Select EmployeeID, LastName, FirstName,
Title From Employees"
            Dim connection As SqlConnection = New SqlConnec-
tion(connectionString)
            Dim command As SqlCommand = New SqlCommand(strCommand, connection)
            Dim reader As SqlDataReader

            Try
                connection.Open()
                reader = command.ExecuteReader()
                DataList1.DataSource = reader
                DataList1.DataBind()
                reader.Close()
            Catch ex As Exception
                connection.Close()
            End Try
            Load Data into List()
        ElseIf e.CommandName = "save" Then
```

```
            Dim connectionString As String = ConfigurationManag-
er.ConnectionStrings("NorthwindDB").ConnectionString
            Dim strCommand As String = "UPDATE Employees SET LastName
=@LastName, FirstName=@FirstName, Title=@Title WHERE EmployeeID=@EmployeeID"
            Dim connection As SqlConnection = New SqlConnec-
tion(connectionString)
            Dim command As SqlCommand = New SqlCommand(strCommand, connection)
            command.Parameters.AddWithValue("@LastName", Di-
rectCast(e.Item.FindControl("tbLName"), TextBox).Text)
            command.Parameters.AddWithValue("@FirstName", Di-
rectCast(e.Item.FindControl("tbFName"), TextBox).Text)
            command.Parameters.AddWithValue("@Title", Di-
rectCast(e.Item.FindControl("tbTitle"), TextBox).Text)
            command.Parameters.AddWithValue("@EmployeeID", Di-
rectCast(e.Item.FindControl("lblID"), Label).Text)
            Try
                connection.Open()
                command.ExecuteNonQuery()
            Catch ex As Exception

            Finally
                connection.Close()
            End Try
            DataList1.EditItemIndex = -1
        ElseIf e.CommandName = "cancel" Then
            DataList1.EditItemIndex = -1
        End If
        Load_Data_into_List()
    End Sub
```

Κώδικας 11.47 – Ενημέρωση πεδίων εγγραφής σε μια Datalist

11.7. DetailsView

Data controls όπως το GridView και ο Repeater προορίζονται για περιπτώσεις που επιθυμούμε να εμφανίσουμε στο χρήστη πολλές εγγραφές υπό τη μορφή πινάκων ή λιστών. Υπάρχουν ωστόσο περιπτώσεις που επιθυμούμε να εμφανίσουμε λεπτομερή στοιχεία για μια μόνο εγγραφή. Για τέτοιες περιπτώσεις προορίζονται τα controls, FormView και DetailsView. Το DetailsView δημιουργεί ένα HTML πίνακα και τοποθετεί κάθε πεδίο μιας μόνο εγγραφής σε διαφορετική γραμμή του πίνακα.

```
ASP

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px"
DataSourceID="SqlDataSource1">
</asp:DetailsView>
```

Κώδικας 11.48 – Ορισμός ενός DetailsView

11.7.1.Πεδία

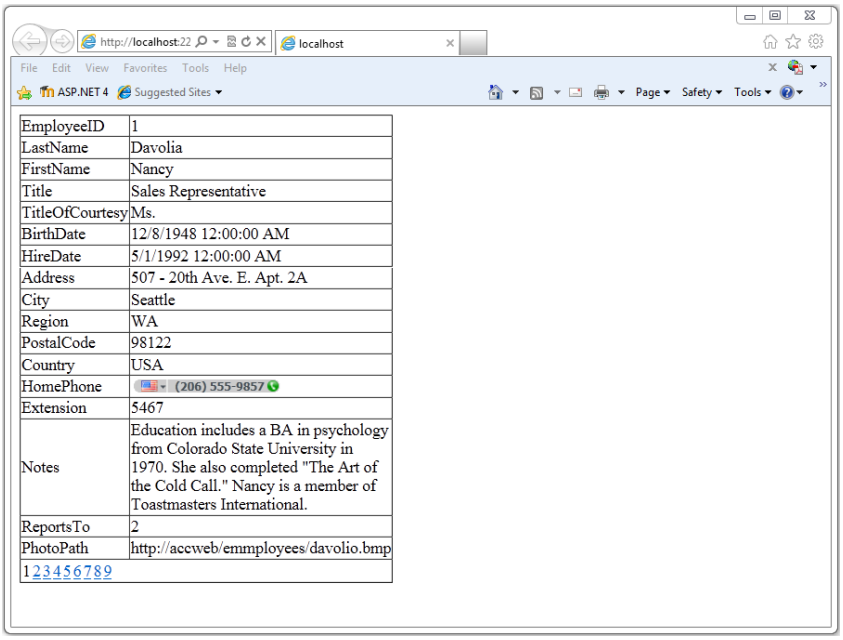
Το DetailsView υποστηρίζει ένα πλήθος διαφορετικών τύπων πεδίων: BoundField, ButtonField, CheckBoxField, CommandField, DynamicField, HyperLinkField, ImageField και TemplateField. Θα διαπιστώσετε ότι οι υποστηριζόμενοι τύποι πεδίων είναι ίδιοι με αυτούς που μπορούν να οριστούν σε ένα GridView και κατά συνέπεια η εργασία μας με αυτούς δεν έχει σημαντικές διαφορές. Να σημειωθεί ότι ορίζουμε πεδία εντός των tags Fields σε αντίθεση με ότι έχουμε συνηθίσει από το GridView που ορίζαμε πεδία εντός των tags Columns.

11.7.2.Παρουσίαση εγγραφών

Όπως είπαμε ένα DetailsView προορίζεται για την παρουσίαση δεδομένων μιας μόνο εγγραφής. Αν ένα DetailsView γίνει bind με ένα data source το οποίο ανακτά πολλές εγγραφές τότε εξ ορισμού το DetailsView θα εμφανίσει την πρώτη μόνο εγγραφή από αυτά τα δεδομένα.

Είναι δυνατόν να παρακάμψουμε αυτή τη συμπεριφορά μέσω του μηχανισμού paging. Θέτοντας την ιδιότητα AllowPaging σε true εμφανίζεται μια λίστα στο κάτω μέρος του DetailsView μέσω τις οποία μπορούμε να περιηγηθούμε και στις υπόλοιπες εγγραφές των αποτελεσμάτων. Μέσω των ιδιοτήτων PagerSettings, PagerStyle, PagerTemplate μπορούμε

να ελέγξουμε συγκεκριμένα χαρακτηριστικά του control που εμφανίζει τις σελίδες με τρόπο ανάλογο με αυτόν που έχουμε εξετάσει για το GridView.



Εικόνα 11.9 DetailsView με paging

11.7.3.Επεξεργασία εγγραφών

Το DetailsView υποστηρίζει τις διαδικασίες Insert, Update, Delete ωστόσο αυτές δεν βασίζονται σε μηχανισμό ανάλογο με το CommandButton που είδαμε στο GridView. Στην περίπτωση του DetailsView οι διαδικασίες αυτές επιτελούνται απλούστερα θέτοντας σε true την τιμή των ιδιοτήτων AutoGenerateInsertButton, AutoGenerateUpdateButton, AutoGenerateDelete. Η πράξη αυτή αυτομάτως προσθέτει τους υπερσυνδέσμους Insert, Update και Delete στο κάτω μέρος του DetailsView.

Όταν πατηθεί το Delete τότε η εγγραφή διαγράφεται άμεσα, όταν όμως πατηθεί το Insert και το Update τότε το DetailsView μεταβαίνει σε insert ή edit mode αντίστοιχα.

11.7.4.Προβολή λεπτομερειών με DetailsView

Ένα αρκετά διαδεδομένο σενάριο χρήσης ενός DetailsView είναι για να παρουσιάσει λεπτομέρειες μια εγγραφής που περιέχεται σε ένα GridView, DataList ή ένα Repeater. Σε τέτοιου είδους σενάρια τυπικά λίγα πεδία μιας λίστας εγγραφών εμφανίζονται ένα data bound control τύπου λίστας και περιέχουν παράλληλα μια στήλη που παρέχει τη δυνατότητα ανακατεύθυνσης σε μια άλλη σελίδα. Η σελίδα αυτή περιέχει ένα DetailsView με όλα τα πεδία της επιλεγμένης εγγραφή.

Ενώ στην περίπτωση του GridView μπορεί να χρησιμοποιηθεί ένα Select χωρίς παραμέτρους για να ανακτήσει αυτά τα δεδομένα, στην περίπτωση του DetailsView η εντολή Select θα πρέπει να έχει παραμέτρους και μάλιστα το ID της επιλεγμένης εγγραφής του GridView. Στο παράδειγμα που παρουσιάζουμε στη συνέχεια η πληροφορία αυτή (το επιλεγμένο ID δηλαδή) περνά από τη μία σελίδα στην άλλη μέσω ενός πεδίου στο query string. Θυμηθείτε ότι το query string γράφεται ως μια σειρά παραμέτρων-τιμών στη διεύθυνση μετά το χαρακτήρα (?).

Σε μια στήλη HyperlinkField ενός GridView μπορούμε να ορίσουμε αυτή να περνά μια παράμετρο στο query string μέσω το πεδίου DataNavigateUrlFormatString. Η ιδιότητα αυτή ορίζει τη μορφή που θα έχει το URL στο οποία θα ανακατευθύνει η στήλη. Η θέση στην οποία μπαίνουν οι παράμετροι ορίζεται από την ακολουθία {#}, όπου # είναι ο αριθμός της παραμέτρου από αυτές που ορίζονται στο DataNavigateUrlFields. Στην ιδιότητα DataNavigateUrlFields ορίζονται μια ή περισσότερες παράμετροι. Έτσι μια διεύθυνση της μορφής "~/Example_DetailsView.aspx?ID={0}" ορίζει ότι η ανακατεύθυνση θα γίνεται στη σελίδα Ex-

ample_DetailsView.aspx και το query string θα περιλαμβάνει ένα και μοναδικό πεδίο, το ID. Η τιμή του πεδίου ID θα λαμβάνεται από την πρώτη παράμετρο που ορίζεται στη λίστα DataNavigateUrlFields. Στην προκειμένη περίπτωση μιας και η συγκεκριμένη λίστα περιέχει μια μόνο τιμή η παράμετρος θα παίρνει τιμές από το πεδίο EmployeeID.

Εφόσον το επιλεγμένο ID περνά από τη μια σελίδα στην άλλη σαν παράμετρος στο query string, το DataSource το οποίο γίνεται bind στο DetailsView θα πρέπει να έχει αναγκαστικά μια παράμετρο QueryStringParameter. Το στοιχείο QueryStringParameter έχει την ιδιότητα QueryStringField η οποία ορίζει το όνομα του πεδίου στο query string από το οποίο θα παίρνει τιμές η παράμετρος. Έτσι στο παραάδειγμά μας ο κώδικας <asp:QueryStringParameter Name="EmployeeID" QueryStringField="ID" /> δηλώνει ότι στο SelectCommand η παράμετρος EmployeeID θα παίρνει τιμές από το πεδίο του query string με το όνομα ID.

```
ASP

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString= "<%$
ConnectionStrings:NorthwindDB %>"
 ProviderName="System.Data.SqlClient"
 SelectCommand="SELECT Employees.* FROM Employees WHERE EmployeeID=@EmployeeID"
>

   <SelectParameters>
      <asp:QueryStringParameter Name="EmployeeID" QueryStringField="ID" />
   </SelectParameters>
</asp:SqlDataSource>

<asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px"
DataSourceID="SqlDataSource1">

</asp:DetailsView>
```

```
ASP

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataKeyNames="EmployeeID" DataSourceID="SqlDataSource1">
   <Columns>
      <asp:HyperLinkField DataNavigateUrlFields="EmployeeID"
       DataNavigateUrlFormatString="~/Example_DetailsView.aspx?ID={0}"
       DataTextField="EmployeeID" HeaderText="EmployeeID" />
      <asp:BoundField DataField="LastName" HeaderText="LastName"
       SortExpression="LastName" />
      <asp:BoundField DataField="FirstName" HeaderText="FirstName"
       SortExpression="FirstName" />
      <asp:BoundField DataField="Title" HeaderText="Title" SortExpres-
sion="Title" />
      <asp:CommandField ButtonType="Button" HeaderText="Action" ShowEditBut-
ton="True" ShowHeader="True" />
  </Columns>
</asp:GridView>
```

Κώδικας 11.49 – Παρουσίαση λεπτομερειών μιας εγγραφής σε DetailsView