Chapter Eleven:
The Web Server Environment
Chapter Objectives

• To understand the nature and characteristics of the data environment that surrounds Internet technology database applications
• To learn the purpose, features, and facilities of ODBC
• To understand the characteristics of the Microsoft .NET Framework
• To understand the nature and goals of OLE DB
• To learn the characteristics and object model of ADO.NET
Chapter Objectives

- To understand the characteristics of JDBC and the four types of JDBC drivers
- To understand the nature of JSP and know the differences between JSP and ASP.NET
- To understand HTML and PHP
- To be able to construct Web database applications pages using PHP
Introduction

• Because database applications today reside in a complicated environment, various standards have been developed for accessing database servers.

• Some of the important standards:
  – **ODBC** (Open Database Connectivity) is the early standard for relational databases.
  – **OLE DB** is Microsoft’s older object-oriented interface for relational and other databases.
  – **The .NET Framework and ADO.NET** are the current Microsoft “dot NET” data access standards providing easier access to data.
  – **JBDC** (Java Database Connectivity) is the Java “equivalent” of ODBC.
A Web server needs to publish applications that involve different data types.
The Role of the ODBC Standard

- Native Interfaces
- DBMS
- Relational Databases: Oracle Database, Microsoft SQL Server, Oracle MySQL, Microsoft Access, IBM DB2...
- Nonrelational Databases
- VSAM, ISAM, Other File Processors
- E-mail, Other Document Types
- Pictures, Audio, Other???

Browser

Web Server
The **ODBC (Open Database Connectivity)** standard provides a DBMS-independent means for processing relational database data.

- It was developed in the early 1990s by an industry committee and has been implemented by Microsoft and many other vendors.
- The goal is to allow a developer to create a single application that can access databases supported by different DBMS products without needing to be changed or recompiled.
ODBC Architecture

Application can process a database using any of the three DBMS products.
ODBC Components

• ODBC consists of a **data source**, an **application program**, a **driver manager**, and a **DBMS driver**.

• A **data source** is the database, its associated DBMS, operating system, and network platform.
  – An ODBC data source can be a relational database, a file server, or a spreadsheet.

• An **applications program** issues requests to create a connection with a data source.
ODBC Components

• A **driver manager** determines the type of DBMS for a given ODBC data source and loads that driver in memory.

• A **DBMS driver** processes ODBC requests and submits specific SQL statements to a given type of data source.
  – A **single-tier driver** processes both ODBC calls and SQL statements.
  – A **multiple-tier driver** processes ODBC calls, but passes the SQL requests to the database server.
ODBC Driver Types:
ODBC Single-Tier Driver
ODBC Driver Types: ODBC Multiple-Tier Driver
Conformance Levels

• Levels of conformance balance the scope of the ODBC standard.

• There are two types of conformance levels:
  – **ODBC conformance levels** concern the features and functions that are made available through the driver’s application program interface (API).
    • A **driver API** is a set of functions that the application can call to receive services.
  – **SQL conformance levels** specify which SQL statements, expressions, and data types a driver can process.
## ODBC Conformance Levels

<table>
<thead>
<tr>
<th>Core API</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Connect to data sources</td>
</tr>
<tr>
<td>• Prepare and execute SQL statements</td>
</tr>
<tr>
<td>• Retrieve data from a result set</td>
</tr>
<tr>
<td>• Commit or roll back transactions</td>
</tr>
<tr>
<td>• Retrieve error information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 1 API</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Core API</td>
</tr>
<tr>
<td>• Connect to data sources with driver-specific information</td>
</tr>
<tr>
<td>• Send and receive partial results</td>
</tr>
<tr>
<td>• Retrieve catalog information</td>
</tr>
<tr>
<td>• Retrieve information about driver options, capabilities, and functions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 2 API</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Core and Level 1 API</td>
</tr>
<tr>
<td>• Browse possible connections and data sources</td>
</tr>
<tr>
<td>• Retrieve native form of SQL</td>
</tr>
<tr>
<td>• Call a translation library</td>
</tr>
<tr>
<td>• Process a scrollable cursor</td>
</tr>
</tbody>
</table>
# SQL Conformance Levels

## Minimum SQL Grammar

- CREATE TABLE, DROP TABLE
- Simple SELECT (does not include subqueries)
- INSERT, UPDATE, DELETE
- Simple expressions (A > B + C)
- CHAR, VARCHAR, LONGVARCHAR data types

## Core SQL Grammar

- Minimum SQL Grammar
- ALTER TABLE, CREATE INDEX, DROP INDEX
- CREATE VIEW, DROP VIEW
- GRANT, REVOKE
- Full SELECT (includes subqueries)
- Aggregate functions such as SUM, COUNT, MAX, MIN, AVG
- DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT, DOUBLE PRECISION data types

## Extended SQL Grammar

- Core SQL Grammar
- Outer joins
- UPDATE and DELETE using cursor positions
- Scalar functions such as SUBSTRING, ABS
- Literals for date, time, and timestamp
- Batch SQL statements
- Stored procedures
A **data source** is an ODBC data structure that identifies a database and the DBMS that processes it.

Three types of data source names:

- A **file data source** is a file that can be shared among database users having the same DBMS driver and privilege.
- A **system data source** is local to a single computer and may be used by the operating system and any user on that system.
  - System data sources are recommended for Web servers.
  - To define a system data source name, the type of driver and the database need to be specified.
- A **user data source** is available only to the user who created it.
Creating a System Data Source I

Select **System DSN** and click the **Add** button.

The **MySQL ODBC 5.1 Driver**

Select the **SQL Server Native Client 10.0**

Click the **Finish** button.
Creating a System Data Source II

Type in the name for this System DSN: VRG

Type in a description

The drop-down list arrow button—select the SQL server from the drop-down list—if the list is empty, type in the name of the server itself, not the SQL Server instance name.

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: VRG

How do you want to describe the data source?

Description: VRG Database on SQL Server R2

Which SQL Server do you want to connect to?

Server: WSBS2-002
Creating a System Data Source III

Click this check box for SQL Server authentication

Type in the user Login ID here

Type in the associated user Password here

The Next button

How should SQL Server verify the authenticity of the login ID?

2. With SQL Server authentication using a login ID and password entered by the user.

Login ID: VRG-User
Password: ************

Connect to SQL Server to obtain default settings for the additional configuration options.
Creating a System Data Source IV

Click this check box to manually select the default database.

If necessary, select the correct database from the drop-down list displayed clicking this drop-down list arrow.

The Next button.
Creating a System Data Source

The Next button
Creating a System Data Source VI
Creating a System Data Source VII

The OK button

Test Results

Microsoft SQL Server Native Client Version 10.50.1600
Running connectivity tests...
Attempting connection
Connection established
Verifying option settings
Disconnecting from server
TESTS COMPLETED SUCCESSFULLY!
Creating a System Data Source VIII

The VRG system data source

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.
32-Bit versus 64-Bit ODBC I

• The ODBC Data Sources program we have been using is for the 64-bit version of the ODBC data sources administration program, and will create 64-bit ODBC data sources.

• These will only work as long as every component in the Web application program chain is a 64-bit program. But, if any component is a 32-bit program, the 64-bit ODBC data source will not work.
32-Bit versus 64-Bit ODBC II

• In fact, when you are building the Web database applications in this chapter, if everything seems to be correctly done but the system still does not work, the most likely cause is a 32-bit program in the chain.

• To resolve this problem, you will need to use the 32-bit version of the ODBC Data Sources program located at C:\Windows\SysWOW64\odbcad32.exe.
32-Bit versus 64-Bit ODBC III

• To resolve your Web database application problem, create a 32-bit ODBC data source using ODBC Administrator, and then revise your Web page code to use that ODBC data source.

• For more information on the programs in the Windows\SysWOW64 folder, see the Wikipedia article on WoW64 at http://en.wikipedia.org/wiki/WoW64.
32-Bit versus 64-Bit ODBC IV

- The two icons for the two versions appear as shown below. Note that although the program icons are the same, the program names are different. Data Source (ODBC) is the 64-bit version, and ODBC Administrator is the 32-bit versions.

This the 64-bit version of the ODBC Data Source Administrator program

This the 32-bit version of the ODBC Data Source Administrator program
## The Microsoft .NET Framework

<table>
<thead>
<tr>
<th>Version</th>
<th>Components</th>
</tr>
</thead>
</table>
| 4.0     | - Parallel LINQ (PLINQ)  
          - Task Parallel Library (TPL) |
| 3.5     | - Language Integrated Query (LINQ)  
          - ADO.NET Entity Framework  
          - ADO.NET Data Services  
          - ADO.NET AJAX |
| 3.0     | - Windows Presentation Foundation (WPF)  
          - Windows Communication Foundation (WCF)  
          - Windows Workflow Foundation (WWF) |
| 2.0     | - ASP.NET  
          - ADO.NET  
          - Base Class Library  
          - Common Language Runtime |
The Role of OLE DB

Native Interfaces → DBMS

DBMS → ODBC

ODBC → 

Relational Databases:
Oracle Database, Microsoft SQL Server, Oracle MySQL, Microsoft Access, IBM DB2 . . .

Nonrelational Databases

VSAM, ISAM, Other File Processors

E-mail, Other Document Types

Pictures, Audio, Other?????

Browser → Web Server

Browser → Web Server

Browser → Web Server
• **OLE DB** is an implementation of the Microsoft OLE object standard.
  – OLE DB objects are COM objects and support all required interfaces for such objects.

• OLE DB breaks the features and functions of a DBMS into COM objects, making it easier for vendors to implement portions of functionality.
  – This characteristic overcomes a major disadvantage of ODBC.
  – With ODBC, a vendor must create an ODBC driver for almost all DBMS features and functions in order to participate in ODBC at all.
Object-Oriented Concepts

• An **object-oriented programming object** is an abstraction that is defined by its properties and methods.
  – An **abstraction** is a generalization of something.
  – A **property** specifies a set of characteristics of an object.
  – A **method** refers to actions that an object can perform.
  – A **collection** is an object that contains a group of other objects.
OLE DB Goals

• Create object interfaces for DBMS functionality pieces:
  – Query, update, transaction management, etc.

• Increase flexibility:
  – Allow data consumers to use only the objects they need
  – Allow data providers to expose pieces of DBMS functionality
  – Providers can deliver functionality in multiple interfaces
  – Interfaces are standardized and extensible

• Provide object interfaces over any type of data:
  – Relational and non-relational database, ODBC or native, VSAM and other files, Email, etc.

• Do not force data to be converted or moved from where it is
OLE DB Basic Constructs

• There are data consumers and data providers:
  – **Data consumers**—users of OLE DB functionality.
  – **Data providers**—sources of OLE DB functionality.

• An **interface** is a set of objects and the properties and methods they expose in that interface:
  – Objects may expose different properties and methods in different interfaces.

• An **implementation** is how an object accomplishes its tasks:
  – Implementations are hidden from the outside world and may be changed without impacting the users of the objects.
• A rowset is equivalent to a cursor.

• OLE DB has two types of data providers:
  – Tabular data provider—exposes data via rowsets
    • Examples: DBMS, spreadsheets, ISAMs
  – Service provider—a transformer of data through OLE DB interfaces
    • It is both a consumer and a provider of transformed data.
    • Examples: query processors, XML document creator
Rowset Interfaces

- **IRowSet**
  - Methods for sequential iteration through a rowset
- **IAccessor**
  - Methods for setting and determining bindings between rowset and client program variables
- **IColumnsInfo**
  - Methods for determining information about the columns in the rowset
- **Other interfaces**
  - Scrollable cursors
  - Create, update, delete rows
  - Directly access particular rows (bookmarks)
  - Explicitly set locks
  - And so on
The Role of ADO

Relational Databases: Oracle Database, Microsoft SQL Server, Oracle MySQL, Microsoft Access, IBM DB2

Native Interfaces

DBMS

ODBC

Nonrelational Databases

VSAM, ISAM, Other File Processors

E-mail, Other Document Types

Pictures, Audio, Other???

Browser

Web Server

Browser

Browser
The Role of ADO.NET

- ADO.NET
- DBMS
- Windows Applications
- Web Applications
- XML Web Services
- DB
A .NET **data provider** is a library of classes that provides ADO.NET services.

Microsoft’s provides three data providers:

- **OLE DB data provider** can be used to process any OLE DB-compliant data source.
- **SQL Server Client data provider** is purpose-built for use with SQL Server.
- **Oracle Database Client data provider** is purpose-built for use with Oracle Database.
Data Provider Components

- A **connection object** is similar to the ODBC’s connection object.
- A **command object** is created on an established connection.
- A **data reader** provides read-only, forward-only, fast access to database data.
- An **application** can get and put data to and from the database using the command object.
- A **DataSet** is an in-memory database that is disconnected from any regular database.
  - It distinguishes ADO.NET from the previous data access technology.
Data Provider Components

ADO.NET Data Providers:
- OLE DB
- SQL Server Client
- Oracle Database Client
- Others...
The ADO.NET DataSet

• A **DataSet** is an in-memory database that is disconnected from any regular database.

• DataSets can have:
  – Multiple tables, views, and relationships.
    • Tables may have surrogate key (auto increment columns), primary keys, and be declared as unique.
  – Referential integrity rules and actions.
  – The equivalent of triggers.

• DataSets may be constructed from several different databases and managed by different DBMS.
The ADO.NET Object Model
The ADO.NET Dataset Object Model
DataSet Advantages

• DataSet contents and its XML schema can be easily formatted as an XML document.
• Also, XML schema documents can be read to create the structure of the dataset, and XML documents can be read to fill the dataset.
• DataSets are needed to provide a standardized, nonproprietary means to process database views.
  – This is important for the processing of views with multiple multivalue paths.
DataSet Disadvantages

- Because DataSet data is disconnected from the regular database, only optimistic locking can be used when updating the regular database with the DataSet.
- In the case of conflict, either the dataset must be reprocessed or the data change must be forced onto the database, causing the lost update problem.
- Thus, datasets cannot be used for applications in which optimistic locking is problematical.
  - Instead, the ADO.NET command object should be used.
JDBC

- **JDBC** is an alternative to ODBC and ADO that provides database access to programs written in Java.
- JDBC used to be a “nonacronym”—it didn’t stand for anything!
- Now it stands for Java DataBase Connectivity.
- JDBC drivers are available for most DBMS products:
## JDBC Driver Types

<table>
<thead>
<tr>
<th>Driver Type</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>JDBC–ODBC bridge. Provides a Java API that interfaces to an ODBC driver. Enables processing of ODBC data sources from Java.</td>
</tr>
<tr>
<td>2</td>
<td>A Java API that connects to the native-library of a DBMS product. The Java program and the DBMS must reside on the same machine, or the DBMS must handle the intermachine communication, if not.</td>
</tr>
<tr>
<td>3</td>
<td>A Java API that connects to a DBMS-independent network protocol. Can be used for servlets and applets.</td>
</tr>
<tr>
<td>4</td>
<td>A Java API that connects to a DBMS-dependent network protocol. Can be used for servlets and applets.</td>
</tr>
</tbody>
</table>
Java Servlets and Applets

• Java programs are compiled into an operating system independent bytecode.
• Various operating systems use their own bytecode interpreters a.k.a. Java virtual machines.
• An applet is transmitted to a browser via HTTP and is invoked on the client workstation using the HTTP protocol.
• A servlet is a Java program that is invoked on the server to respond to HTTP requests.
• Type 3 and Type 4 drivers can be used for both applets and servlets.
• Type 2 drivers can be used only in servlets.
JDBC Components
Java Server Pages

- **Java Server Pages (JSP)** provide a means to create dynamic Webpages using HTML, XML, and Java.

- JSPs provide the capabilities of a full object-oriented language to the page developer.
  - Neither VBScript nor JavaScript can be used in a JSP.
  - JSPs are compiled into machine-independent bytecode.
JSP Compilation Process
IIS Web Server I

The C: drive
The inetpub folder
The wwwroot folder
The iisstart.htm file
IIS Web Server II

The **Default Web Site** location maps to the wwwroot folder

The **iisstart.htm** file

The **Content View** pane is selected
This Web page is generated by the iisstart.htm file.
index.html as Default Document

The Features View
Default Document settings page

The index.html file name is already listed

The Features View pane is selected
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <title>View Ridge Gallery Demonstration Pages Home Page</title>
</head>
<body>
<h1 style="text-align: center; color: blue">
  Database Processing (12th Edition)
</h1>
<h2 style="text-align: center; font-weight: bold">
  David M. Kroenke
</h2>
<h2 style="text-align: center; font-weight: bold">
  David J. Auer
</h2>
<hr />
<h2 style="text-align: center; color: blue">
  Welcome to the View Ridge Gallery Home Page
</h2>
<hr />
<p>Chapter 11 Demonstration Pages From Figures in the Text:</p>
<p>Example 1: &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
  &lt;a href="ReadArtist.php"
       &gt;
    Display the ARTIST Table (LastName, FirstName, Nationality)
  &lt;/a&gt;
</p>
<hr />
</body>
</html>
This IP number indicates that the Web server is on a separate computer—if the Web server is on your computer itself, use the URL http://localhost/DBP/VRG
index.html in Eclipse

The DBP-e12-VRG project—Eclipse organizes work into projects.

The index.html HTML code—note how color coding has been added to indentation to keep the code organized and readable.
<?php

// Get connection
$DSN = "VRG";
$user = "VRG-User";
$password = "VRG-User+password";
$conn = odbc_connect($DSN, $user, $password);

// Test connection
if (!$conn)
{
    exit("ODBC Connection Failed: ", $conn);
}

// Create SQL statement
$sql = "SELECT LastName, FirstName, Nationality FROM ARTIST";

// Execute SQL statement
$recordset = odbc_exec($conn, $sql);

// Test existence of recordset
if (!$recordset)
{
    exit("SQL Statement Error: ", $sql);
}

<!-- Page Headers -->
<h1>
The View Ridge Gallery Artist Table
</h1>
<h2>
ARTIST
</h2>
<?php

// Table headers
echo "<table class='output' border='1'>
<tr>
  <th>LastName</th>
  <th>FirstName</th>
  <th>Nationality</th>
</tr>";

// Table data
while($RecordSetRow = odbc_fetch_array($RecordSet))
{
  echo "<tr>
    <td>" . $RecordSetRow['LastName'] . "</td>
    <td>" . $RecordSetRow['FirstName'] . "</td>
    <td>" . $RecordSetRow['Nationality'] . "</td>
  </tr>";
}

// Close connection
odbc_close($Conn);

<br />
<hr />

<p class="footer">
  <a href="../VRG/index.html">
    Return to View Ridge Gallery Home Page
  </a>
</p>
<hr />
</body>
</html>
The ReadArtist.php code—PHP code is enclosed in the `<?php` and `?>` symbols, which are displayed in red in Eclipse.
ReadArtist.php
Initial HTML Code

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>InsertNewArtist</title>
  <style type="text/css">
    h1 {text-align: center; color: blue}
    h2 {font-family: Arial, sans-serif; text-align: left; color: blue}
    p.footer {text-align: center}
    table.output {font-family: Arial, sans-serif}
  </style>
</head>
<body>
</body>
```
<?php

// Get connection
$DSN = "VRG";
$User = "VRG-User";
$Password = "VRG-User+password";

$conn = odbc_connect($DSN, $User, $Password);

// Test connection
if (!$conn)
{
    exit ("ODBC Connection Failed: ". $conn);
}
// Create SQL statement
$SQL = "SELECT LastName, FirstName, Nationality FROM ARTIST";

// Execute SQL statement
$RecordSet = odbc_exec($Conn, $SQL);

// Test existence of recordset
if (!$RecordSet)
{
    exit ("SQL Statement Error: " . $SQL);
}
?>
<!-- Page Headers -->
<h1>
The View Ridge Gallery Artist Table
</h1>
<hr />
<h2>
ARTIST
</h2>

<?php

// Table headers
echo "<table class='output' border='1' 
   <tr>
     <th>LastName</th>
     <th>FirstName</th>
     <th>Nationality</th>
   </tr>

//Table data
while($RecordSetRow = odbc_fetch_array($RecordSet))
{
    echo "<tr>
    echo "<td>" . $RecordSetRow['LastName'] . "</td>
    echo "<td>" . $RecordSetRow['FirstName'] . "</td>
    echo "<td>" . $RecordSetRow['Nationality'] . "</td>

    echo "</tr>
}

echo "</table>";
ReadArtist.php

Disconnecting from the Database

// Close connection
odbc_close($Conn);

?>
<br />
<hr />
<p class="footer">
    <a href="../VRG/index.html">
        Return to View Ridge Gallery Home Page
    </a>
</p>
</hr />
</body>
</html>
This IP number indicates that the Web server is on a separate computer—if the Web server is on your computer itself, use the URL http://localhost/DBP/VRG

Click to return to the View Ridge Gallery Home Page
<?php

// Get connection
$DSN = "MGR2";
$User = "VRG-User";
$Password = "VRG-User+password";

$PDOconnection = new PDO("odbc:".$DSN, $User, $Password);

// Test connection
if (!$PDOconnection)
    exit ("ODBC Connection Failed: ", $PDOconnection);

// Create SQL statement
$sql = "SELECT LastName, FirstName, Nationality FROM ARTIST";

// Execute SQL statement
$RecordSet = $PDOconnection->query($sql);

// Test existence of recordset
if (!$RecordSet)
    exit ("SQL Statement Error: ", $sql);

?>

<!-- Page Headers -->

<h1>
    The View Ridge Gallery Artist Table
</h1>

<h2>
    ARTIST
</h2>
<?php

// Table headers
echo "<table class='output' border='1' "
<tr>
    <th>LastName</th>
    <th>FirstName</th>
    <th>Nationality</th>
</tr>
";

// Table data
while($GRecordSetRow = $GRecordSet->fetch())
{
    echo "<tr>
    echo "<td>" . $GRecordSetRow['LastName'] . "</td>
    echo "<td>" . $GRecordSetRow['FirstName'] . "</td>
    echo "<td>" . $GRecordSetRow['Nationality'] . "</td>
    echo "</tr>";
}

echo "</table>";

// Close connection
$PDOconnection = null;

<br />
<hr />
<p class="footer">
    <a href="/VRG/index.html"
        Return to View Ridge Gallery Home Page
</a>
</p>
<br />
</body>
</html>
### The View Ridge Gallery Artist Table

<table>
<thead>
<tr>
<th>Artist</th>
<th>FirstName</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miro</td>
<td>Joan</td>
<td>Spanish</td>
</tr>
<tr>
<td>Kandinsky</td>
<td>Wassily</td>
<td>Russian</td>
</tr>
<tr>
<td>Klee</td>
<td>Paul</td>
<td>German</td>
</tr>
<tr>
<td>Matisse</td>
<td>Henri</td>
<td>French</td>
</tr>
<tr>
<td>Chagall</td>
<td>Marc</td>
<td>French</td>
</tr>
<tr>
<td>Sargent</td>
<td>John Singer</td>
<td>United States</td>
</tr>
<tr>
<td>Tobey</td>
<td>Mark</td>
<td>United States</td>
</tr>
<tr>
<td>Horiuchi</td>
<td>Paul</td>
<td>United States</td>
</tr>
<tr>
<td>Graves</td>
<td>Morris</td>
<td>United States</td>
</tr>
<tr>
<td>Anderson</td>
<td>Guy</td>
<td>United States</td>
</tr>
</tbody>
</table>

[Return to View Ridge Gallery Home Page]
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>NewCustomerAndInterestsForm</title>
<style type="text/css">
  h1 {text-align: center; color: blue; }
  h2 {font-family: Arial, sans-serif; text-align: left; color: blue; }
  p, footer {text-align: center; }
  table, output {font-family: Arial, sans-serif; }
</style>
</head>
<body>
<form action="InsertNewCustomerAndInterestsPDO.php" method="POST">
  <!-- Page Headers -->
  <h1> The View Ridge Gallery New Customer Form 

  <br />
  <p> Enter customer data: </p>
  </body>
  <table>
    <tr>
      <td>&nbsp; Last Name: &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
<tr>
<td>Email:<br />
</td>
<td>
<input type="text" name="Email" size="100" />
</td>
</tr>
</table>
<p>Select artist nationality:<br />
</p>
<select name="Nationality">
<option value="Canadian">Canadian</option>
<option value="English">English</option>
<option value="French">French</option>
<option value="German">German</option>
<option value="Mexican">Mexican</option>
<option value="Russian">Russian</option>
<option value="Spanish">Spanish</option>
<option value="United States">United States</option>
</select>
<br />
<p>
<input type="submit" value="Add New Customer" />
<input type="reset" value="Reset Values" />
</p>
</form>
<br />
<br />
<p class="footer">
<a href="/VRG/index.html">
Return to View Ridge Gallery Home Page
</a>
</p>
<br />
</body>
</html>
The New Customer and Interests Form

The customer data is entered in the **Last Name**, **First Name**, **Area Code**, **Phone**, and **Email** text boxes.

The artist nationality is selected from the drop-down list.

The **Add New Customer** button is used to submit the data.

The **Reset Values** button is used to clear the data in the form.
```php
// Get connection
$DSN = "DBD:\$DSN";
$User = "DB User\Password";
$Password = "DB User\Password";

$conn = new PDO("odbc:\$DSN", $User, $Password);

// Test connection
if (!$conn)
    exit("DB Connection Failed: ". $conn);

// Create short variable names
$LastName = $_POST["LastName"];
$FirstName = $_POST["FirstName"];
$AreaCode = $_POST["AreaCode"];
$PhoneNumber = $_POST["PhoneNumber"];
$Email = $_POST["Email"];
$Nationality = $_POST["Nationality"];

// Create SQL statement to call the stored Procedure
$sql = "EXEC InsertCustomerAndInterests @" . $conn . ";

// Execute SQL statement
$result = $conn->exec($sql);

// Test existence of $result
if (!$result)
    exit("SQL Statement Error: ". $sql);

// Create SQL statement to retrieve additions to
// CUSTOMER_ARTIST_INT table
$sql = "SELECT * FROM CustomerInterestsView ";
$sql .= WHERE CustomerLastName = "$LastName";
$sql .= AND CustomerFirstName = "$FirstName";
```
// Execute SQL statement
$RecordSet = $PDOconnection->query($SQL);

// Test existence of $RecordSet
if (!is($RecordSet)) {
    exit("SQL Statement Error: ". $SQL);
}

echo "<h1>
The View Ridge Gallery CUSTOMER Table
</h1>
<br />
";

echo "<h2>
New Customer and Artist Interests Added:
</h2>
<table>
  <tr>
    <td>Last Name</td>
    <td>First Name</td>
    <td>Area Code</td>
    <td>Phone Number</td>
    <td>Email</td>
    <td>Artist Nationality</td>
    <td>Nationality</td>
  </tr>

// Table headers
echo "<table class='output' border='1'>
<tr>
  <th>CustomerLastName</th>
  <th>CustomerFirstName</th>
  <th>ArtistName</th>
</tr>";
// Table data
while($RecordSetRow = $RecordSet->fetch())
{
    echo "<tr>";
    echo "<td>" . $RecordSetRow['CustomerLastName'] . "</td>
    echo "<td>" . $RecordSetRow['CustomerFirstName'] . "</td>
    echo "<td>" . $RecordSetRow['ArtistName'] . "</td>
    echo "</tr>
}

// Close connection
$PDOconnection = null;

But
<br />
<hr />
<p class="Footer">
<a href="/VRG/index.html">Return to View Ridge Gallery Home Page</a>
</p>
<br />
</hr>
</body>
</html>
The New Customer and Artist Interests Added message is displayed along with the customer and artist interest data.

## The View Ridge Gallery CUSTOMER Table

**New Customer and Artist Interests Added:**

- **Last Name:** Baxendale
- **First Name:** Richard
- **Area Code:** 206
- **Phone Number:** 876-7733
- **Email Address:** Richard.Baxendale@elsewhere.com
- **Artist Nationality:** United States

<table>
<thead>
<tr>
<th>CustomerLastName</th>
<th>CustomerFirstName</th>
<th>ArtistName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baxendale</td>
<td>Richard</td>
<td>Sargent</td>
</tr>
<tr>
<td>Baxendale</td>
<td>Richard</td>
<td>Tobey</td>
</tr>
<tr>
<td>Baxendale</td>
<td>Richard</td>
<td>Horiiuchi</td>
</tr>
<tr>
<td>Baxendale</td>
<td>Richard</td>
<td>Graves</td>
</tr>
<tr>
<td>Baxendale</td>
<td>Richard</td>
<td>Anderson</td>
</tr>
</tbody>
</table>

Return to View Ridge Gallery Home Page.
End of Presentation:
Chapter Eleven