

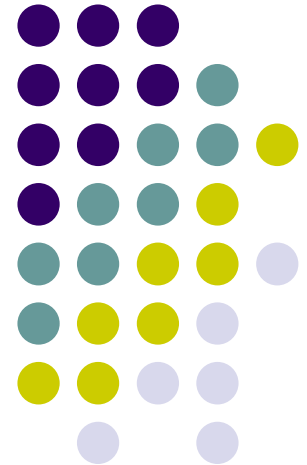
Desarrollo de Aplicaciones Web

Desarrollo Web en Entornos de Servidor



Tema 5

Técnicas de acceso a datos



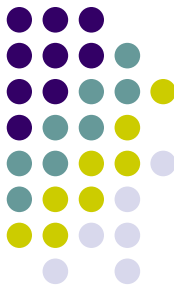
Vicente J. Aracil Miralles

vj.aracilmiralles@edu.gva.es

06/09/2022

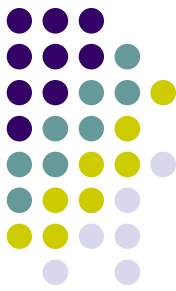
Tema 5

Técnicas de acceso a datos



Objetivos

- Comprender la necesidad de integrar datos almacenados en el servicio de información Web
- Conocer las principales características de las arquitecturas de Bases de Datos para la Web como variante de la arquitectura Cliente-Servidor
- Utilizar el modelo de acceso a datos declarativo para el desarrollo de interfaces adecuadas a los propósitos de una aplicación Web interactiva con acceso a datos
- Conocer y utilizar los controles de datos de ASP.NET
- Utilizar el modelo de acceso a datos basado en código lógico para realizar el acceso a datos a través de los objetos de ADO.NET
- Comprender la funcionalidad que ofrecen las principales clases de ADO.NET para manipular (consultar, insertar, eliminar y actualizar) datos almacenados en una base de datos
- Conocer el funcionamiento de las transacciones en las bases de datos
- Reconocer la necesidad de establecer mecanismos de seguridad para el acceso a datos



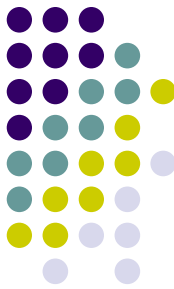
Tema 5

Técnicas de acceso a datos

Contenidos

- 5.1 Integración de datos en el servicio de información Web
- 5.2 Acceso a datos con ASP.NET
- 5.3 Modelo de acceso a datos declarativo
- 5.4 Controles de origen de a datos de ASP.NET
- 5.5 Controles enlazados a datos de ASP.NET
- 5.6 Modelo de acceso a datos mediante código
- 5.7 Introducción a ADO.NET
- 5.8 Manipulación de datos con ADO.NET
- 5.9 Transacciones
- 5.10 Seguridad de acceso a datos desde aplicaciones Web

5.1 Integración de datos en el servicio de información Web



Necesidad de integración de datos en el servicio Web

- El desarrollo de aplicaciones Web interactivas se relaciona con:
 - La necesidad de integrar en el servicio de información Web los datos almacenados en bases de datos, y en otros tipos de orígenes de datos, **de modo que las aplicaciones Web puedan representar funcionalidades complejas** que:
 - Exploten y presenten información de **naturaleza cambiante a lo largo del tiempo** para que pueda quedar adaptada a la realidad del momento
 - Deben quedar ajustadas a las **necesidades de información propias de cada usuario**
- El desarrollo de aplicaciones Web interactivas puede ayudar a:
 - Generar nuevas funcionalidades en los sistemas de gestión de las organizaciones
 - Satisfacer las necesidades y requerimientos de usuarios, que cada vez son más exigentes en relación a las funcionalidades que deben incorporar las aplicaciones Web

Las aplicaciones Web interactivas se basan en que la interacción del usuario produzca un cambio en la información contenida en el recurso recibido para quedar adaptada a la realidad del momento

5.1 Integración de datos en el servicio de información Web



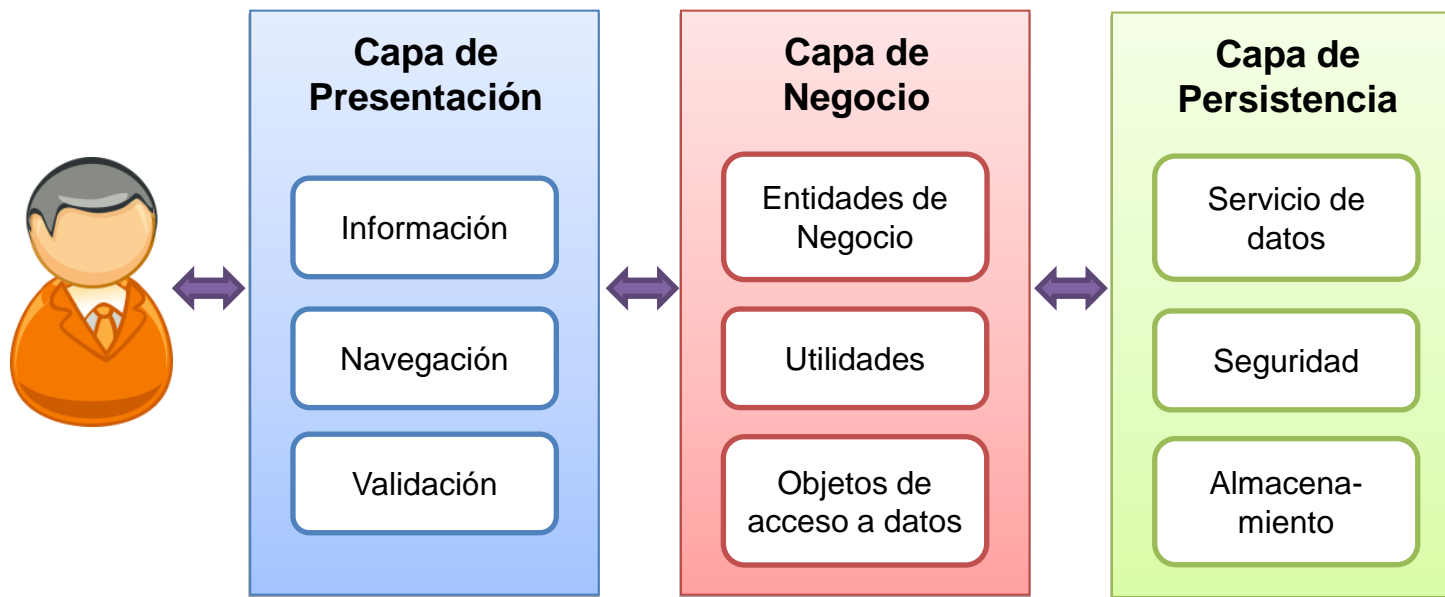
Modelo de arquitectura de software de tres capas

- El modelo de arquitectura del software del servicio Web se ha definido como un caso particular del modelo de arquitectura Cliente/Servidor
 - El **modelo de desarrollo de software de tres capas** se basa en que las funcionalidades de la aplicación Web quedan agrupadas en diferentes capas, quedando cada una de estas capas especializada en la gestión de un aspecto determinado del servicio
 - Es posible encontrarse con divisiones funcionales más o menos especializadas, por lo que el modelo de desarrollo de tres capas en los entornos Cliente/Servidor puede dar lugar a arquitecturas de software multicapa, según cada caso concreto
- Dentro de las arquitecturas multicapa, la **arquitectura de tres capas** es una de las más utilizadas para el desarrollo de aplicaciones Web
 - **Divide la concepción y la construcción** de una aplicación Web en tres partes:
 - Capa de presentación
 - Capa de lógica de negocio
 - Capa de datos o de persistencia

5.1 Integración de datos en el servicio de información Web



Modelo de arquitectura de software de tres capas



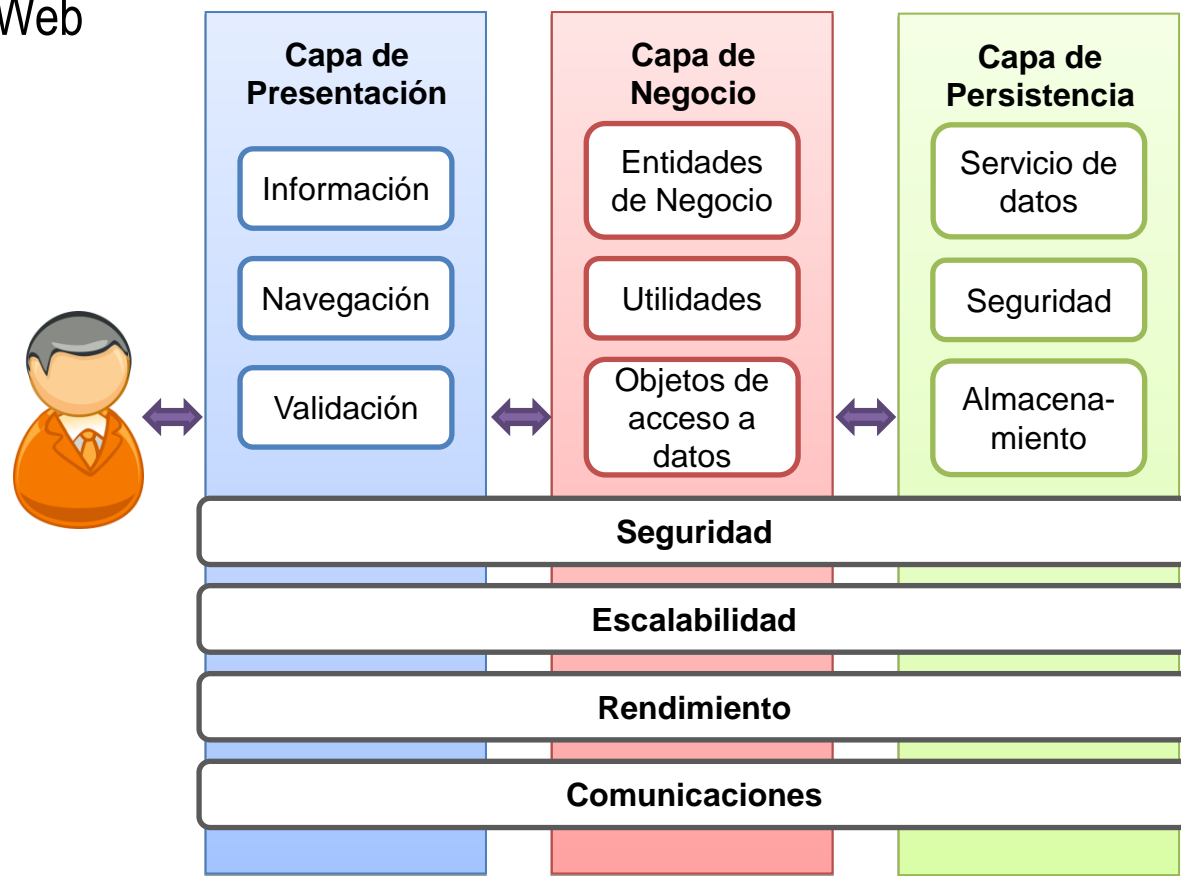
- Ayuda a concebir y construir los componentes software a partir de la división funcional de las capas lógicas de una aplicación Web
 - Cada capa es una colección de componentes software que se proporcionan servicios entre sí o a otras capas adyacentes
- Entre las decisiones de diseño que se deben adoptar (análisis y diseño) están:
 - La definición de las capas en las que se va a dividir funcionalmente la aplicación Web
 - La especificación de los componentes software que se van a agrupar en cada capa

5.1 Integración de datos en el servicio de información Web



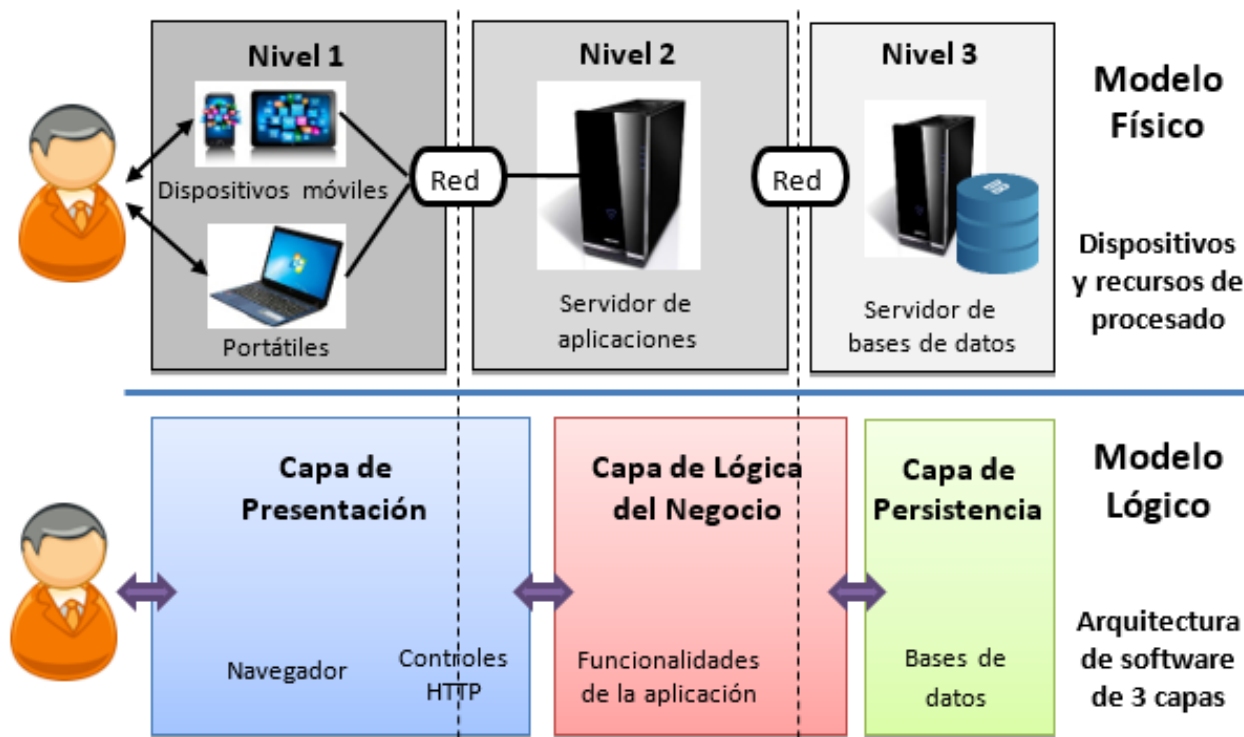
Aspectos transversales en la arquitectura de tres capas

- En las aplicaciones Web existen **aspectos transversales** cuya funcionalidad no se implementa en una sola capa, sino que deben estar distribuidos por todas las capas de la aplicación Web



5.1 Integración de datos en el servicio de información Web

Modelo lógico y físico de la arquitectura de tres capas



Es muy poco habitual que cada uno de los niveles del modelo físico (hardware) coincida perfectamente con una capa del modelo lógico (software) de la arquitectura del software

- En el caso concreto de la ilustración, la capa de presentación contiene controles de validación de servidor y la capa de lógica del negocio contiene procedimientos almacenados en la base de datos



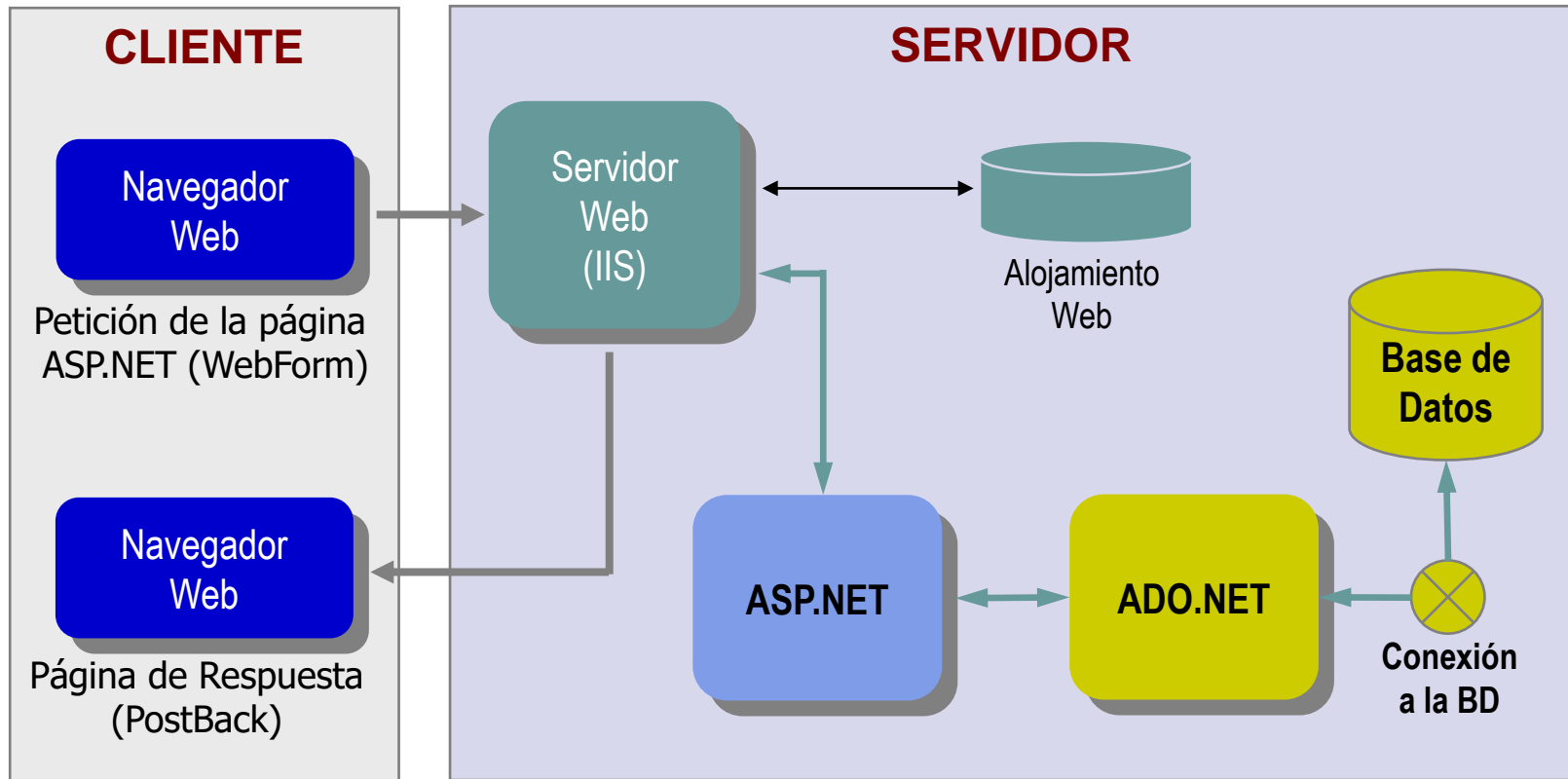
Características generales

- La tecnología ASP.NET contiene características que permiten conectarse a diversos tipos de orígenes de datos:
 - Bases de datos
 - Datos almacenados en archivos XML
 - Objetos comerciales de servicios de datos
- La tecnología nativa de acceso a bases de datos de ASP.NET se basa en el uso de la librería de clases ADO.NET (*ActiveX Data Objects .NET*), cuyas características más significativas son:
 - Facilita el acceso estándar a datos mediante conexiones a cualquier base de datos
 - Constituye un mecanismo para el acceso a bases de datos desde las aplicaciones desarrolladas a través de la infraestructura de desarrollo .NET Framework
 - Se basa en el uso de componentes en objeto reutilizables

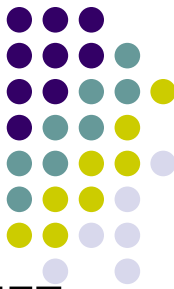
5.2 Acceso a datos con ASP.NET



Dinámica del mecanismo de acceso a datos de ASP.NET

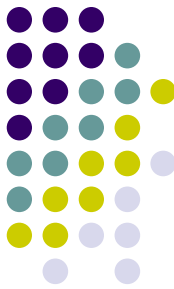


La página de respuesta HTML incorpora de forma dinámica en su contenido la información resultante del acceso a los datos y de la ejecución del procesamiento en el servidor



Dinámica del mecanismo de acceso a datos de ASP.NET

- Dinámica del procesamiento del acceso a datos en el servidor Web con ASP.NET cuando se incluye un acceso a los datos almacenados en una base de datos:
 1. En primer lugar, un usuario a través de un navegador Web **solicita una página de ASP.NET** que incluye procesamiento en el servidor Web y el acceso a un servidor de base de datos para realizar una operación de manipulación de los datos (consulta, inserción, eliminación o modificación)
 2. Establecida la conexión con el servidor Web, éste localiza la página de ASP.NET solicitada en su sistema de almacenamiento asociado y, al detectar que el recurso incluye código de procesamiento en el servidor Web, se realiza **una llamada al módulo de procesamiento de ASP.NET**
 3. Se inicia la **ejecución del procesamiento** solicitado en el módulo de procesamiento de ASP.NET
 4. Dado que el procesamiento en el servidor Web incluye un acceso a un servidor de base de datos, que suele estar especificado mediante una sentencia de SQL de manipulación de datos, **se establece la conexión con la base de datos y se ejecuta el comando** de acceso a datos
 5. Una vez finalizado el procesamiento, **el servidor Web envía el resultado del procesamiento al cliente a través de un documento HTML** válido que incorpora de forma dinámica en su contenido la información resultante del acceso a los datos y de la ejecución del procesamiento en el servidor Web
 6. Finalmente, conforme el cliente va recibiendo el contenido del documento HTML resultante del procesamiento realizado en el servidor Web (página de respuesta), este contenido va siendo interpretado por el navegador Web y **se presenta sobre la ventana de visualización del navegador**



Modelos de acceso a datos de ASP.NET

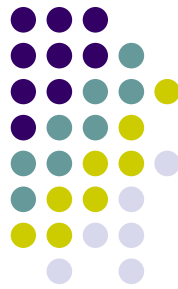
- Las aplicaciones Web de ASP.NET, pueden obtener acceso a datos de dos formas:
 - **Modelo de acceso a datos declarativo**
 - El acceso a datos no precisa de código lógico para los escenarios de acceso a datos más comunes: mostrar, seleccionar, actualizar, insertar y eliminar datos
 - Se basa en la utilización de controles de servidor especializados para establecer el origen y el enlace a los datos. Estos controles se denominan **controles de acceso a datos** y se agregan sobre la interfaz del Web Form (archivo .aspx) arrastrándolos a la vista Diseño desde el cuadro de herramientas o escribiendo el código de marcado correspondiente en la vista Código
 - **Modelo de acceso a datos mediante código**
 - El acceso a datos se realiza a través de **código lógico** de acceso a datos que se escribe en el archivo de código subyacente (archivo .aspx.cs)
 - El código lógico utiliza, según sea el caso, las:
 - Clases del espacio de nombres *System.Data*, para el acceso a datos almacenados en bases de datos
 - Clases del espacio de nombres *System.Xml*, para el acceso a datos almacenados en documentos XML
 - El conjunto de clases de forman el espacio de nombres *System.Data* recibe, comúnmente, el nombre de librería de clases ADO.NET



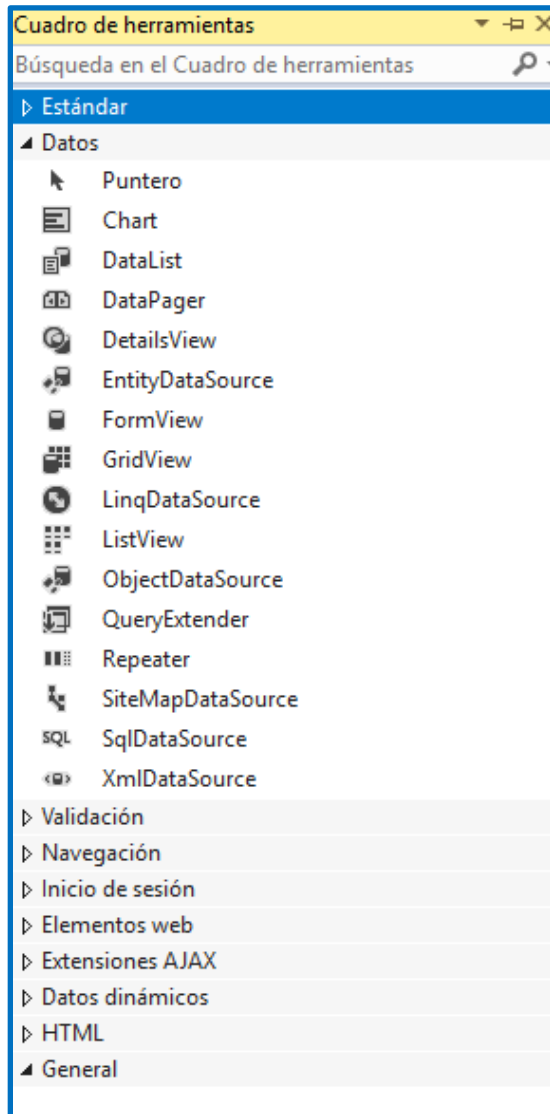
Características

- Los escenarios más comunes de acceso a datos pueden ser resueltos mediante el modelo de acceso a datos declarativo, utilizando para ello los **controles de acceso a datos** de ASP.NET
- Los controles de acceso a datos
 - Son controles de servidor especializados que facilitan el acceso y la presentación de los datos en las páginas de ASP.NET
 - Administran las tareas subyacentes para conectar con un origen de datos y, también, para presentar, interactuar y modificar los datos almacenados
- Para acceder a datos almacenados mediante los controles de acceso a datos desde las páginas Web de ASP.NET es necesario:
 - Establecer la conexión con el origen de datos desde la aplicación Web
 - Se utilizan los controles de origen de datos
 - Presentar e interactuar con los datos
 - Se utilizan los controles enlazados a datos

5.3 Modelo de acceso a datos declarativo

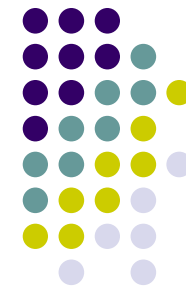


Controles de acceso a datos



- Tipos de controles de acceso a datos de ASP.NET:
 - **Controles de origen de datos**
 - Administran las tareas de conexión al origen de datos y de lectura o escritura de información
 - Actúan como intermediarios entre un origen de datos en particular (base de datos, archivo XML u objeto comercial) y los controles que constituyen la interfaz de usuario para poder presentar e interactuar con los datos
 - No representan ningún aspecto relativo a la interfaz de usuario
 - **Controles enlazados a datos**
 - Representan los controles que constituyen la interfaz de usuario para presentar e interactuar con los datos y poder realizar las operaciones de manipulación (consultar, insertar, eliminar y modificar) de datos
 - Se enlazan a un control de origen de datos, a través de su propiedad ***DataSource***

5.4 Controles de origen de datos de ASP.NET



Tipos de controles de origen de datos

Control de origen de datos	Descripción
SqlDataSource	Permite trabajar con proveedores de datos de ADO.NET, que proporcionan acceso a diversos tipos de bases de datos: Microsoft SQL Server, OLE DB, ODBC u Oracle
XmlDataSource	Permite enlazar con datos almacenados en archivos XML. También es útil para el uso de controles de servidor ASP.NET jerárquicos tales como el control TreeView o Menu
LinqDataSource	Permite utilizar el lenguaje LINQ (Language-Integrated Query) en una página Web ASP.NET para recuperar y modificar datos en un objeto de datos. Admite la generación automática de comandos de selección, actualización, inserción y eliminación de datos. También admite operaciones de ordenación, filtrado y paginación
ObjectDataSource	Permite utilizar objetos comerciales de servicio de datos. Facilita el desarrollo de aplicaciones Web que se basan en objetos para administrar los datos
EntityDataSource	Permite enlazar a datos basados en Entity Data Model (EDM) de Entity Framework (EF). Admite la generación automática de comandos de actualización, inserción, eliminación y selección, así como la ordenación, el filtrado y la paginación de datos.
SiteMapDataSource	Se utiliza juntamente con los controles Menu y TreeView para establecer las opciones de navegación de una aplicación Web

5.5 Controles enlazados a datos de ASP.NET



Tipos de controles enlazados a datos

Control enlazado a datos	Descripción
GridView	Muestra los datos en una tabla y permite editar, actualizar, eliminar, ordenar y paginar datos sin necesidad de escribir código lógico
DetailsView	Muestra un registro cada vez en una disposición de tabla de campos y permite editar, eliminar e insertar un registro. Se pueden incluir opciones de paginación.
FormView	Es similar al control DetailsView, pero permite definir una disposición de formato libre para cada registro. El control FormView es como un control DataList para un registro único.
ListView	Permite definir el diseño de los datos enlazados mediante plantillas. Admite operaciones de ordenación automática, edición, inserción y eliminación. Reemplaza a Repeater y DataList.
Repeater	Representa los datos en una lista. Cada elemento se representa utilizando una plantilla de elemento definida por el usuario.
DataList	Representa los datos en una tabla. Cada elemento se representa utilizando una plantilla de elemento definida por el usuario.
DataPager	Se puede utilizar para mostrar controles de paginación para otros controles de enlazado a datos que implementan la interfaz.
Chart	Representa los datos en un gráfico de tipo: estándar, líneas, barras, circulares, etc.
Controles de lista	Son los controles ListBox, DropDownList, CheckBoxList, RadioButtonList y BulletedList que pueden ser enlazados a un origen de datos

5.5 Controles enlazados a datos de ASP.NET



Ejemplo. Mantenimiento de empleados

EMPLEADOS

Nombre del empleado:

Buscar

	Código Empleado	NIF	Nombre	Departamento
<input type="button" value="Seleccionar"/>	A-00120	48124441-X	Jiménez Varela, Pilar	Desarrollo
<input type="button" value="Seleccionar"/>	A-18088	21999001-K	Salort Martinez, Juan Ignacio	Innovación tecnológica
<input type="button" value="Seleccionar"/>	A-20423	21123123-T	García Vallejo, Juan	Investigación básica
<input type="button" value="Seleccionar"/>	A-21449	34001989-V	Barrachina Carot, Carmen	Investigación básica
<input type="button" value="Seleccionar"/>	B-11033	38555667-T	Monllor Fernandez, Francisco	Formación
Siguiente Último				

Código Empleado	B-11033
NIF	<input type="text" value="38555667-T"/>
Nombre	<input type="text" value="Monllor Fernandez, Francisco"/>
Dirección	<input type="text" value="Pza. Murano Blas, 2 - 5º"/>
Ciudad	<input type="text" value="03008 Alicante (Alicante)"/>
Teléfono	<input type="text" value="666111222"/>
Correo	<input type="text" value="fmonllor@acme.com"/>
Fecha Nacimiento	<input type="text" value="02/05/1955"/>
Fecha Incorporación	<input type="text" value="02/09/1960"/>
Sexo	<input checked="" type="radio"/> Hombre <input type="radio"/> Mujer
Código Departamento	<input type="text" value="Formación"/>
<input type="button" value="Actualizar"/>	<input type="button" value="Cancelar"/>



Aspectos generales

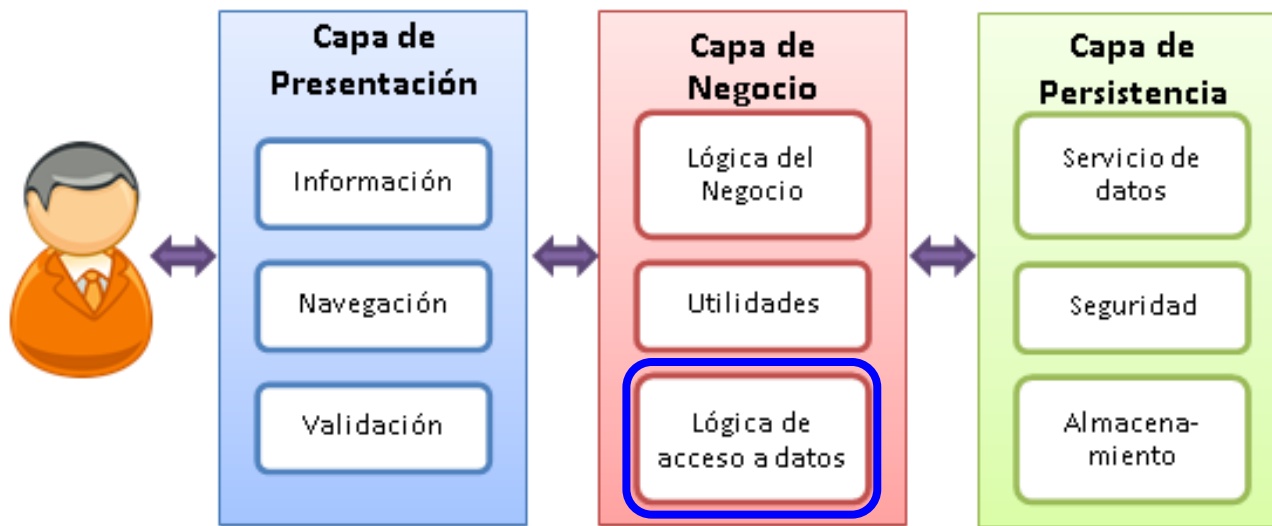
- El modelo de acceso a datos mediante código de ASP.NET se realiza **escribiendo código lógico en los archivos de código subyacente**, para especificar el acceso a datos almacenados desde una aplicación Web
 - El código lógico que expresa el acceso a los datos almacenados utiliza las clases incluidas en la librería de clases **ADO.NET** (ActiveX Data Object .NET)
 - Estas clases especializadas en el acceso a datos almacenados en bases de datos están definidas en el espacio de nombres **System.Data**
 - Cuando se emplea un modelo de acceso a datos mediante código en una aplicación Web construida según un modelo de arquitectura de tres capas, **la capa de presentación no trabaja directamente con los datos**
 - Se invocan las clases y los métodos de ADO.NET desde el código lógico que forma la capa de lógica del negocio para crear y establecer una conexión a la base de datos y para expresar los comandos SELECT, INSERT, UPDATE y DELETE necesarios para implementar la manipulación de los datos
 - De este modo, el **código lógico que establece el acceso a los datos en la capa de lógica de negocio es independiente del resto de las capas**. Existen diversos enfoques para llevar a cabo esta separación funcional del código entre las tres capas

5.6 Modelo de acceso a datos mediante código

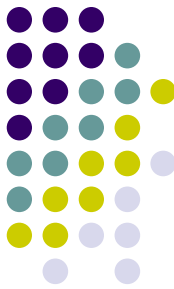


Enfoques para la separación funcional del código

- Un primer enfoque, que garantiza una separación del código entre las tres capas, consiste en:
 - Implementar la lógica de acceso a datos en bibliotecas de clase independientes y mantener la lógica del negocio en los archivos de código subyacente
- Un segundo enfoque, normalmente el recomendado, consiste en:
 - Implementar la lógica del acceso a los datos en los archivos de código subyacente, junto con el código de la lógica del negocio



5.7 Introducción a ADO.NET



La librería de clases ADO.NET

- Es un **conjunto de clases que exponen servicios de acceso a datos** que se emplean cuando se utiliza un modelo de acceso a datos mediante código.
- Se utiliza para:
 - Conectarse a orígenes de datos
 - Ejecutar comandos para manipular los datos (SELECT, INSERT, DELETE y UPDATE) desde las aplicaciones
 - Recuperar los resultados de las consultas
- La librería de clases ADO.NET proporciona acceso a diversos tipos de orígenes de datos:
 - Bases de datos relacionales: SQL Server, Oracle, ODBC, etc.
 - Archivos en formato XML
 - Objetos de servicios de información



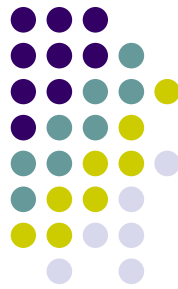
ADO.NET constituye una parte integral de .NET Framework



Componentes de ADO.NET

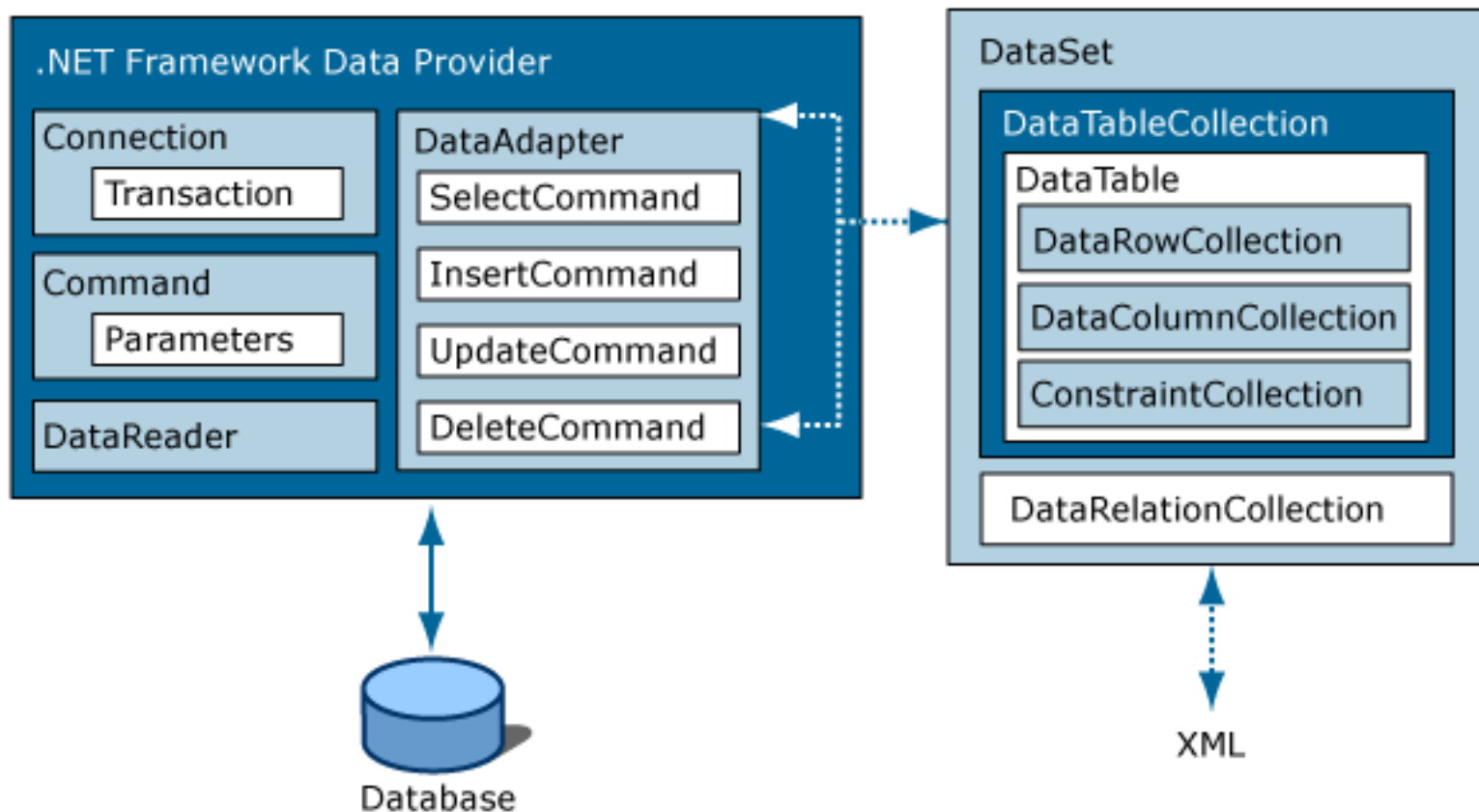
- Los componentes de ADO.NET para el acceso a datos son:
 - Los **proveedores de datos de .NET Framework**
 - Son componentes diseñados especialmente para realizar la manipulación de datos
 - El objeto Connection permite establecer la conexión con un origen de datos
 - El objeto Command permite ejecutar comandos sobre el origen de datos para recuperar datos, modificar datos, ejecutar procedimientos almacenados, etc.
 - El objeto DataReader permite almacenar temporalmente en memoria el flujo de datos resultante de una consulta realizada al origen de datos
 - El objeto DataAdapter utiliza un objeto DataSet para almacenar temporalmente en memoria los datos recuperados mediante una consulta y, además, permite sincronizar con el origen de datos las modificaciones aplicadas a los datos incluidos en el DataSet.
 - El **DataSet**
 - Está diseñado para realiza un acceso a datos de bajo nivel
 - Facilita la manipulación de datos de lectura y escritura y, el acceso a secuencias de datos de avance hacia adelante y retroceso hacia a atrás

5.7 Introducción a ADO.NET

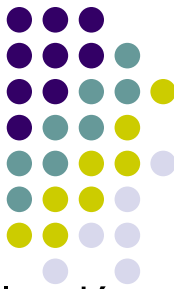


Arquitectura de ADO.NET

- Relación entre los componentes de ADO.NET



5.7 Introducción a ADO.NET



Elegir un DataReader o un DataAdapter para recuperar datos

- Ambos objetos, DataReader y DataAdapter, son una representación de una colección de datos en formato tabular residente en la memoria cache de la aplicación
 - Se emplean para almacenar temporalmente los datos resultantes de una consulta
 - En general, se recomienda emplear el objeto **DataReader** para procesar datos recuperados

DataReader	DataAdapter / DataSet
Acceso de sólo lectura	Acceso de lectura y escritura a datos
Admite solo la recuperación de datos	Admite la recuperación de datos y, además, puede producir reflejo de las modificaciones aplicadas a los datos incluidos en el objeto Dataset que utiliza
Modo conectado de datos	Modo desconectado de datos
Vinculado a un único control	Vinculado a múltiples controles
Recorrido y búsquedas en secuencias de datos sólo hacia delante	Recorrido y búsqueda en secuencias de datos: <ul style="list-style-type: none">● Hacia adelante y hacia atrás● Al primero y al último
Utiliza menos recursos del servidor	Utiliza más recursos del servidor
Acceso rápido a los datos	Acceso más lento



Proveedores de datos .NET Framework de ADO.NET. Generalidades

- Los **proveedores de datos .NET Framework** sirven para:
 - Conectarse a una base de datos
 - Ejecutar comandos de manipulación de datos para recuperar o modificar los datos almacenados en la base de datos
- Cuando se recuperan datos mediante una consulta, los datos recuperados se pueden colocar temporalmente en:
 - Un objeto **DataReader** para poder:
 - Presentar los datos sobre la interfaz Web en el momento que se precise
 - Un objeto **DataAdapter** para poder:
 - Presentar los datos sobre la interfaz en el momento que se precise
 - Y, además, modificar (insertar, eliminar y actualizar) los datos recuperados directamente sobre el DataSet que utiliza
- Los proveedores de datos .NET Framework facilitan una conexión de alto rendimiento y elevada funcionalidad entre el origen de datos y el código lógico de la aplicación Web



Proveedores de datos .NET Framework

Proveedor de datos .NET Framework	Descripción	Espacio de nombres que utiliza
.NET Framework para SQL Server	Proporciona acceso a datos para Microsoft SQL Server versión 7.0 o posterior	System.Data.SqlClient
.NET Framework para Oracle	Proporciona acceso a datos para orígenes de datos de Oracle. El proveedor de datos .NET Framework para Oracle es compatible con la versión 8.1.7 y posteriores del software de cliente de Oracle	System.Data.OracleClient
.NET Framework para ODBC	Proporciona acceso a datos para orígenes de datos que se exponen mediante ODBC	System.Data.Odbc
.NET Framework para OLE DB	Proporciona acceso a datos para orígenes de datos que se exponen mediante OLE DB	System.Data.OleDb
Proveedor EntityClient	Proporciona acceso a datos para las aplicaciones de <i>Entity Data Model</i> (EDM)	System.Data.EntityClient

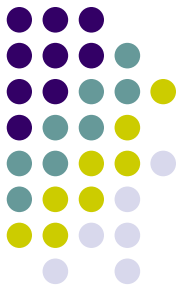
- En función del diseño y del origen de datos, la elección del proveedor de datos .NET Framework puede mejorar el rendimiento, la funcionalidad y la integridad de los datos en la aplicación Web
- Normalmente, se recomienda utilizar bases de datos Microsoft SQL Server como orígenes de datos de las aplicaciones Web basadas en ASP.NET

5.7 Introducción a ADO.NET



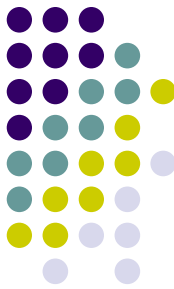
Objetos principales de los Proveedores de datos .NET Framework

Objeto	Descripción
Connection	Establece una conexión a un origen de datos determinado. La clase base para todos los objetos Connection es DbConnection
Command	Ejecuta un comando en un origen de datos para recuperar, modificar, insertar y eliminar datos, así como, para ejecutar procedimientos almacenados y enviar o recuperar datos mediante la utilización de parámetros. Puede ejecutarse en el ámbito de un objeto Transaction desde Connection. La clase base para todos los objetos Command es DbCommand
DataReader	Lee una secuencia de datos de solo avance y solo lectura desde un origen de datos. La clase base para todos los objetos DataReader es DbDataReader
DataAdapter	Llena un DataSet y realiza las actualizaciones necesarias en el origen de datos. Proporciona el puente entre un objeto DataSet y un origen de datos. Utiliza objetos Command para ejecutar comandos SQL en el origen de datos, tanto para cargar el objeto DataSet con datos, como para sincronizar el origen de datos con los cambios aplicados a los datos incluidos en el DataSet. La clase base para todos los objetos DataAdapter es DbDataAdapter



Objetos auxiliares de los Proveedores de datos .NET Framework

Objeto	Descripción
Transaction	Incluye comandos en las transacciones que se realizan en el origen de datos. La clase base para todos los objetos Transaction es DbTransaction . ADO.NET es también compatible con las transacciones que usan clases en el espacio de nombres System.Transactions
ConnectionString Builder	Proporciona un modo sencillo de crear y administrar el contenido de las cadenas de conexión utilizadas por los objetos Connection. La clase base para todos los objetos ConnectionStringBuilder es DbConnectionStringBuilder
Parameter	Define los parámetros de entrada, salida y los valores devueltos para los comandos y procedimientos almacenados. La clase base para los objetos Parameter es DbParameter
Exception	Se devuelve cuando se detecta un error en el origen de datos. En el caso de que el error se detecte en el cliente, los proveedores de datos .NET Framework generan una excepción de .NET Framework. La clase base para todos los objetos Exception es DbException
Error	Expone la información de advertencia o error devuelto por un origen de datos
ClientPermission	Proporciona los atributos de seguridad de acceso del código de los proveedores de datos .NET Framework. La clase base de los objetos ClientPermission es DBDataPermission .

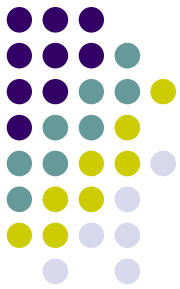


Clases base y clases derivadas de los Proveedores de datos .NET Framework

- Además de las clases base genéricas, **existen clases derivadas específicas para cada proveedor de datos .NET Framework** concreto
 - Por ejemplo, la clase base para todos los objetos Connection de los diferentes proveedores de datos .NET Framework es DbConnection. Existen clases derivadas:

Proveedor de datos de .NET Framework para ...	Clase derivada de la clase base DbConnection
SQL Server	SqlConnection
Oracle	OracleConnection
ODBC	OdbcConnection
OLE DB	OleDbConnection

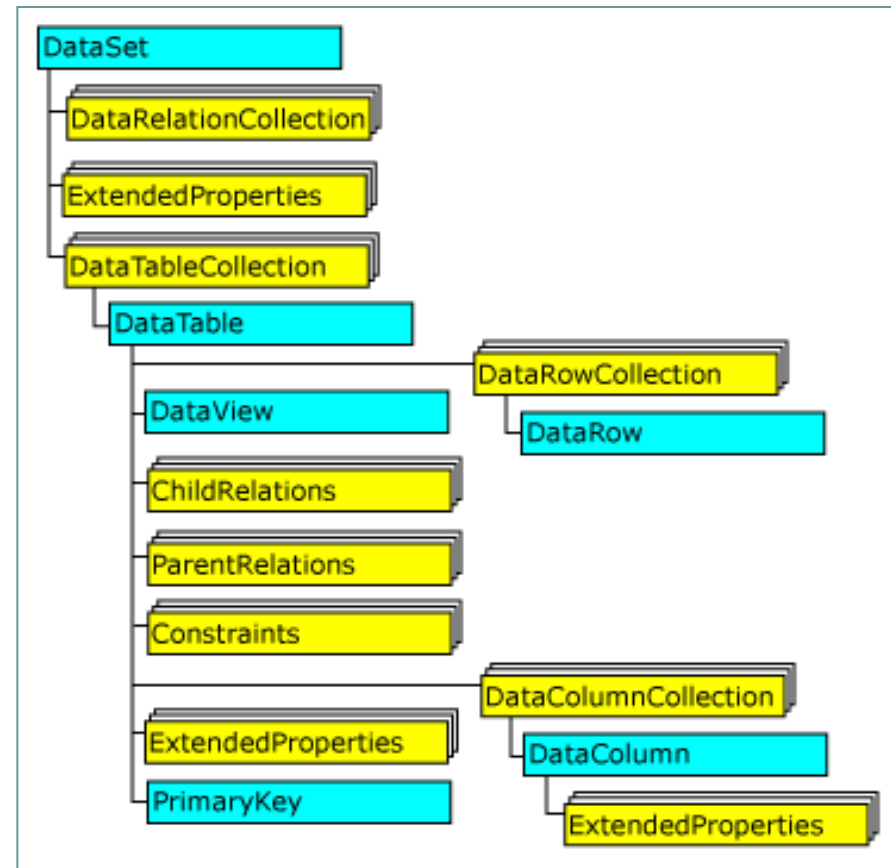
- La clase derivada SqlConnection puede tener declarados métodos específicos para manejar conexiones con SQL Server, que DbConnection puede no contener
- Esto mismo ocurre con todas de las clases base de los objetos de los proveedores de datos .NET Framework: **DbCommand**, **DbDataReader**, etc.
- **Se recomienda utilizar las clases derivadas** porque proporcionan mayor funcionalidad



Objeto DataSet y Modelo de objetos DataSet

- Un **objeto DataSet** es una representación relacional de datos de bajo nivel residente en memoria interna
 - Proporciona un modelo de programación relacional coherente independientemente del origen de datos
 - Esta constituido por una colección de datos que representa un conjunto completo de datos que incluye tablas (*DataTableCollection* y *DataTable*), restricciones (*Constraints*), relaciones entre las tablas (*DataRelationCollection*), etc.
 - Se puede utilizar para mantener y recargar su contenido desde distintos orígenes de datos o para administrar datos locales de la aplicación
 - Permite manejar escenarios de datos distribuidos y desconectados con ADO.NET

Modelo de objetos DataSet

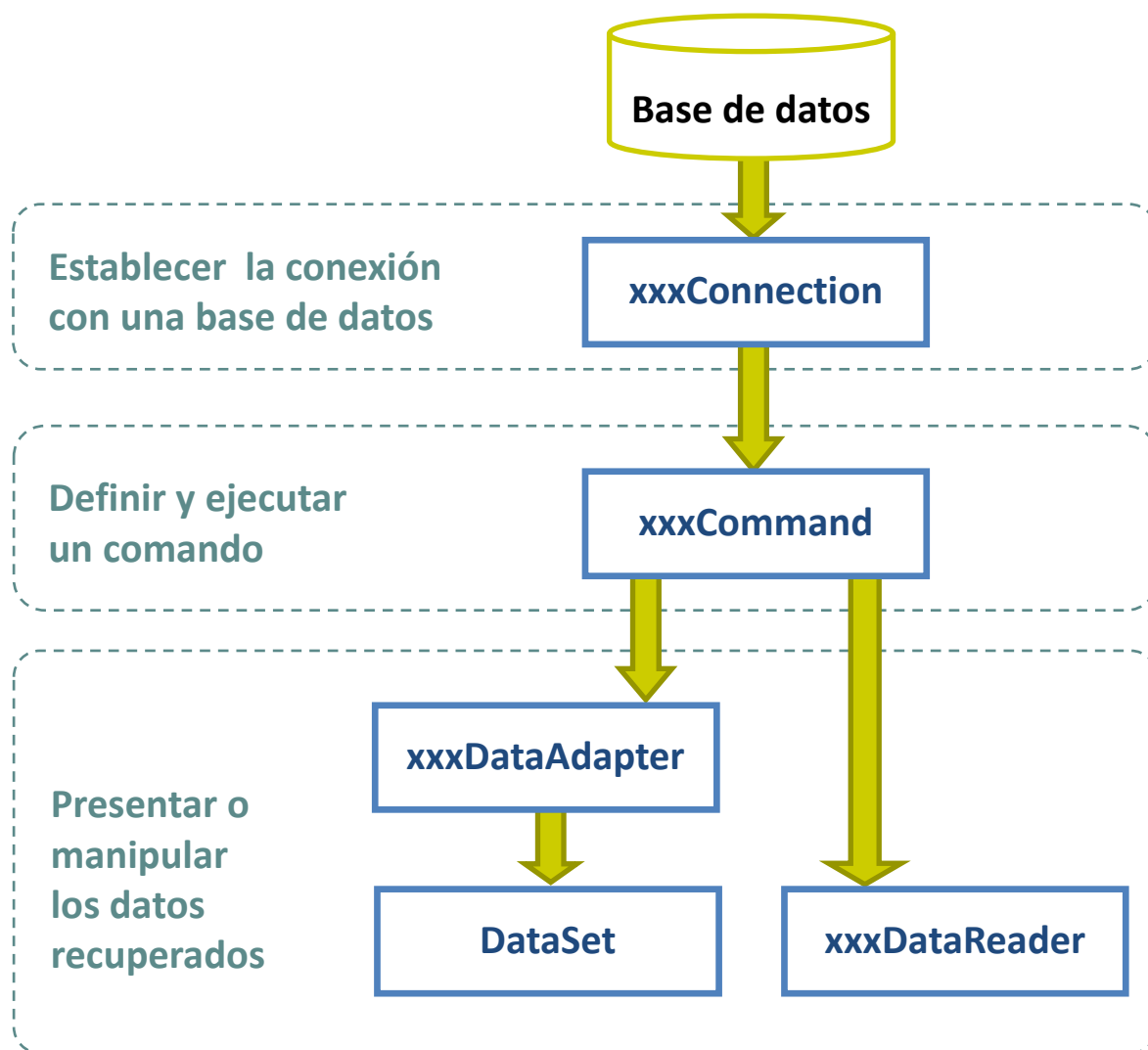


No se profundiza en el estudio de objetos Dataset

5.8 Manipulación de datos con ADO.NET



Esquema de uso de objetos de ADO.NET



Se recomienda emplear objetos `xxxDataReader` para recuperar datos

xxxConnection: permite manejar una conexión a una base de datos desde una aplicación Web

xxxCommand: realiza la ejecución de comandos contra la base de datos

xxxDataReader: proporciona acceso de solo lectura y solo avance a los datos recuperados con una consulta

xxxDataAdapter: facilita el intercambio de datos entre la base de datos y un objeto `DataSet`

xxxDataSet: proporciona una copia local de una colección de datos recuperados mediante un `DataAdapter`



Establecer la conexión con una base de datos

- Se utiliza una instancia a un objeto **Connection** para conectar una aplicación Web con un origen de datos mediante una cadena de conexión
- Miembros más significativos del objeto Connection son:
 - Método **Open()**. Abre una conexión con la base de datos utilizando los valores de inicialización que se especifican en la Cadena de conexión (propiedad `ConnectionString`)
 - Método **Close()**. Cierra la conexión con la base de datos y se destruye la instancia del objeto Connection
 - Método **CreateCommand()**. Crea y devuelve un objeto Command asociado a la conexión
 - Propiedad **ConnectionString**. Obtiene o establece la **Cadena de conexión** utilizada para inicializar y abrir un origen de datos. Ejemplo de Cadena de conexión:

```
string strCadenaConexion = "Data Source=.\SQLEXPRESS;AttachDbFilename=" +  
    Server.MapPath("~/App_Data/Tienda.mdf") +  
    ";Integrated Security=True;Connect Timeout=30;User Instance=True";
```



Establecer la conexión con una base de datos

- Características de la **Cadena de conexión**
 - Es un valor de tipo string que:
 - Contiene información de inicialización de un origen de datos
 - Se transfiere como un parámetro desde un proveedor de datos .NET Framework hacia el origen de datos
 - La sintaxis de la cadena de conexión depende del propio proveedor de datos
 - Una vez validado el origen de datos, se aplican las opciones especificadas en la cadena de conexión y se abre la conexión
 - Los errores de sintaxis generan una excepción en tiempo de ejecución
- Ejemplo. Conexión a una base de datos almacenada en un servidor de base de datos de SQL Server:

```
string strCadenaConexion = "Persist Security Info=False;" +  
    "Integrated Security=true;initial Catalog=AdventureWorks;Server=MSSQL1";
```




Definir y ejecutar un comando

- Una vez establecida una conexión, se puede utilizar un objeto **Command** para ejecutar comandos de manipulación y devolver los resultados obtenidos
- Los miembros más significativos de un objeto Command son:
 - Método **ExecuteReader()**. Devuelve un objeto DataReader cuyo contenido es el resultado de la ejecución de una operación de consulta sobre la base de datos que se ha establecido mediante una instrucción SELECT
 - Método **ExecuteScalar()**. Devuelve un dato simple que suele ser el resultado de una consulta de totales o datos agregados: suma, cuenta, etc.
 - Método **ExecuteNonQuery()**. Ejecuta un comando de manipulación de datos que emplea instrucciones INSERT, UPDATE o DELETE. Devuelve un entero (Int32) con el número de filas afectadas por la ejecución de la instrucción
 - Método **Dispose()**. Libera todos los recursos utilizados por el objeto en el servidor Web
 - Método **ExecuteXMLReader()**. Devuelve un objeto XmlReader. Sólo está disponible para un objeto SqlCommand (proveedor de datos de SQL Server)



Definir y ejecutar un comando

- Propiedad **CommandText**. Obtiene o establece la instrucción SQL, el nombre de la tabla o el procedimiento almacenado que se ejecuta contra el origen de datos
- Propiedad **Connection**. Obtiene o establece la conexión que utiliza la instancia
- Propiedad **CommandType**. Obtiene o establece un valor que indica el tipo de comando que se ha incluido en la propiedad CommandText

Valor Propiedad CommandType	Establece que se ejecutará
CommandType.Text	Sentencia SQL. Valor predeterminado
CommandType.TableDirect	Tabla de la base de datos
CommandType.StoredProcedure	Procedimiento almacenado de la base de datos

- Es importante tener en cuenta que se empleará:
 - **ExecuteReader()** para recuperar datos de una base de datos (SELECT)
 - **ExecuteNonQuery()** para modificar datos mediante instrucciones de inserción (INSERT), eliminación (DELETE) y actualización (UPDATE) de datos



Presentar y manipular los datos

- Los objetos **DataReader** y **DataAdapter** proporcionan acceso a una colección de datos almacenados en memoria que han sido recuperados de una base de datos, mediante la ejecución de un comando de consulta de tipo **SELECT**
 - Además, un objeto DataAdapter:
 - Hace de puente entre un objeto DataSet y un origen de datos para poder Insertar, actualizar y eliminar los datos almacenados en memoria temporalmente en el objeto DataSet
 - Dado que un objeto DataAdapter proporciona un acceso a datos de tipo desconectado, es necesario realizar una operación actualización de datos para producir reflejo de las modificaciones que se realicen sobre los datos del DataSet sobre el origen de datos
- Por razones de rendimiento, **se recomienda el uso de objetos DataReader** para recuperar datos y tenerlos disponibles en memoria principal



Presentar o manipular los datos

- Los miembros más significativos de un objeto **DataReader** son:
 - Método **Read()**. Avanza el apuntador de registro en un objeto DataReader hasta el siguiente registro de la secuencia de datos
 - Inicialmente se sitúa antes del primer registro
 - Devuelve *false* si ha alcanzado el final de la secuencia de filas, en caso contrario devuelve *true*
 - Método **Close()**. Cierra un objeto DataReader y destruye la instancia del objeto
 - Método **GetValues()**. Obtiene el valor de la columna especificada como parámetro de la fila actual del objeto DataReader en su formato nativo
 - También se disponen de métodos específicos según el tipo de datos del valor del campo a obtener: GetString, GetDecimal, GetDateTime, etc.
 - Propiedad **HasRows**. Obtiene un valor lógico que indica si el objeto DataReader contiene una o varias filas (*true*) o ninguna fila (*false*)
 - Propiedad **FieldCount**. Devuelve el número de campos en la fila actual



Presentar o manipular los datos

- Los miembros más significativos de un objeto **DataAdapter** son:
 - Método **Fill()**. Agrega filas de datos en un objeto DataSet. Una vez ejecutado el método Fill() la base de datos se cierra automáticamente (modo desconectado)
 - Método **Update()**. Llama a las instrucciones tipo INSERT, UPDATE o DELETE respectivas para cada fila insertada, actualizada o eliminada en un objeto DataSet que utiliza el objeto DataAdapter, produciéndose reflejo de las modificaciones sobre el origen de datos
 - Propiedad **InsertCommand**. Obtiene o establece un procedimiento almacenado o una instrucción SQL para insertar nuevos registros en un conjunto de datos
 - Propiedad **UpdateCommand**. Obtiene o establece un procedimiento almacenado o una instrucción SQL para actualizar registros en un conjunto de datos
 - Propiedad **DeleteCommand**. Obtiene o establece un procedimiento almacenado o una instrucción SQL para eliminar registros en un conjunto de datos
 - Propiedad **SelectCommand**. Obtiene o establece un procedimiento almacenado o una instrucción SQL para seleccionar registros de un conjunto de datos



Establecer el espacio de nombres en el código lógico

- La palabra reservada **using** del lenguaje de programación C#, permite establecer en el código el espacio de nombres a emplear

Proveedor de datos	Espacio de nombres	Instrucción using a emplear en el código lógico
SQL Server	System.Data.SqlClient	<code>using System.Data.SqlClient;</code>
Oracle	System.Data.OracleClient	<code>using System.Data.OracleClient;</code>
ODBC	System.Data.Odbc	<code>using System.Data.Odbc;</code>
OLE DB	System.Data.OleDb	<code>using System.Data.OleDb;</code>

- Además, si se utilizan objetos DataSet, debe incluirse el espacio de nombres genérico System.Data

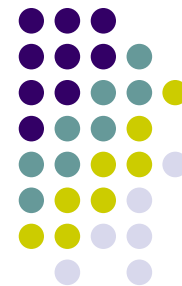
```
using System.Data;
```



Funcionalidades de acceso a datos para el desarrollo de aplicaciones Web

- Los proveedores de datos de .NET Framework de ADO.NET exponen clases que permiten realizar las dos funcionalidades básicas del acceso a datos, que son:
 - **Recuperar (Seleccionar) y presentar los datos** almacenados en la base de datos, para hacer la información accesible al usuario:
 - Ejecutando una instrucción de tipo SELECT definida en un objeto Command
 - Rescatando los datos almacenado temporalmente en un objeto DataReader
 - Especificando una presentación visual adecuada sobre la interfaz Web
 - **Modificar (Insertar, Eliminar o Actualizar) los datos** almacenados en la base de datos, para mantener la consistencia y actualidad de la información
 - Ejecutando una instrucción de tipo INSERT, DELETE o UPDATE definida en un objeto Command
- A continuación, se estudian diversos ejemplos que demuestran diversas formas de construcción del código lógico para poder realizar el acceso a datos mediante el uso de los objetos de ADO.NET

5.8 Manipulación de datos con ADO.NET



Recuperar (Seleccionar) y presentar datos

Ejemplo 1

```
protected void Page_Load(object sender, EventArgs e)
{
    int inNumeroFilas = 0;
    string strResultado, strError;
    string strCadenaConexion = "Data Source=(localdb)\\MSSQL15;AttachDbFilename=" +
        Server.MapPath("~/App_Data/Tienda.mdf") +
        ";Integrated Security=True;Connect Timeout=30";
    string strComandoSql = "SELECT IdCliente,NomCli,PobCli,TelCli,CorCli FROM CLIENTE;";
    SqlConnection conexion = new SqlConnection(strCadenaConexion);
    try
    {
        SqlCommand comando = new SqlCommand(strComandoSql, conexion);
        conexion.Open();
        SqlDataReader reader = comando.ExecuteReader();
        if (reader.HasRows)
        {
            strResultado = "<div class='tabla'>";
            strResultado += "<div class='fila'>";
            strResultado += "<div class='celda encabezado'>Código</div>";
            strResultado += "<div class='celda encabezado'>Nombre</div>";
        }
    }
}
```

Constructor
de comando

Ejecución de la
sentencia SELECT
de SQL



Recuperar (Seleccionar) y presentar datos


Ejemplo 1 (continuación)

```
strResultado += "<div class='celda encabezado'>Población</div>";
strResultado += "<div class='celda encabezado'>Teléfonos</div>";
strResultado += "<div class='celda encabezado'>E-mail</div>";
strResultado += "</div>";
while (reader.Read())
{
    strResultado += "<div class='fila'>";
    strResultado += "<div class='celda'>" + reader.GetString(0) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(1) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(2) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(3) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(4) + "</div>";
    strResultado += "</div>";
    inNumeroFilas++;
}
strResultado = strResultado + "</div>";
strResultado = strResultado + "<p>Número Filas: " + inNumeroFilas + "</p>";
lblResultado.Text = strResultado;
}
```




Recuperar (Seleccionar) y presentar datos

Ejemplo 1 (continuación)

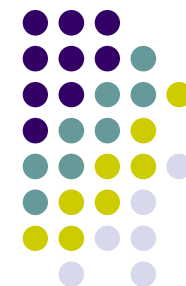


```
else
{
    lblMensajes.Text = "No existen registros resultantes de la consulta";
}
if (reader != null) reader.Close();
if (comando != null) comando.Dispose();
}
catch (SqlException ex)
{
    strError = "<p>Se ha producido errores en el acceso a la base de datos.</p>";
    strError = strError + "<div>Código: " + ex.Number + "</div>";
    strError = strError + "<div>Descripción: " + ex.Message + "</div>";
    lblMensajes.Text = strError;
    return;
}
finally
{
    conexion.Close();
}
}
```



Liberar recursos en el servidor Web

5.8 Manipulación de datos con ADO.NET



Recuperar (Seleccionar) y presentar datos

Ejemplo 1 (continuación)

Cientes

Código	Nombre	Población	Teléfonos	E-mail
00123659Z	Beatriz Iraola López	Bilbao	686343211	beatriz@empresa.com
07899905V	Jesús García Soria	Zaragoza	660542392	suso@empresa.com
08659442S	Miguel Sogorb Fenoll	Alicante	654789995	msogorb@empresa.com
09897907K	Alvaro Fernández Moreno	Madrid	609667878	afm@empresa.com
10900801K	Alejandro Castro Hernández	Zamora	954678966	acastro@empresa.com
11265888J	Sandra Rostroll Trias	Barcelona	931123451	rostroll@empresa.com
11908762X	Rocío Esplá Sirvent	Alicante	965087790	morena@empresa.com
12667489B	Joaquín Porta Sabater	Barcelona	609765890	jporta@empresa.com
15437882H	Sergio Fernández Pérez	Madrid	912345673	ferper@empresa.com
15673999A	Eneko Ugalde Garmendia	Bilbao	606897654	eneko@empresa.com
21123456B	Luis Miguel Boronat Segarra	Valencia	953245367	boro@empresa.com
22348942H	Joaquín Miralles Bernat	Barcelona	935678921	jmiralles@empresa.com
A-48223342	DISTRIBUCIONES COLMAR, S.A.	Madrid	914897600	paquito@empresa.com
B-28697099	COMERCIAL ZAMORANA, S.L.	Madrid	916780987	pedro@empresa.com

Número Filas: 14

5.8 Manipulación de datos con ADO.NET



Recuperar (Seleccionar) y presentar datos

Ejemplo 2. Procesamiento mejorado y recomendado

```
protected void Page_Load(object sender, EventArgs e)
{
    int inNumeroFilas = 0;
    string strResultado, strError;
    string strCadenaConexion =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    string strComandoSql = "SELECT IdCliente,NomCli,PobCli,TelCli,CorCli FROM CLIENTE;";
    using (SqlConnection conexion = new SqlConnection(strCadenaConexion))
    {
        try
        {
            conexion.Open();
            SqlCommand comando = conexion.CreateCommand();
            comando.CommandText = strComandoSql;
            comando.CommandType = System.Data.CommandType.Text;
            SqlDataReader reader = comando.ExecuteReader();
            if (reader.HasRows)
            {
                strResultado = "<div class='tabla'>";
                strResultado += "<div class='fila'>";
                strResultado += "<div class='celda encabezado'>Código</div>";
            }
        }
    }
}
```

Nombre de la cadena de conexión definida en Web.Config

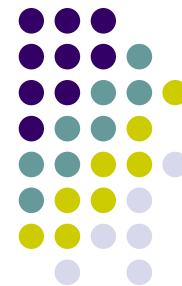
Método CreateCommand()



Recuperar (Seleccionar) y presentar datos


Ejemplo 2. Procesamiento mejorado y recomendado (continuación)

```
strResultado += "<div class='celda encabezado'>Nombre</div>";
strResultado += "<div class='celda encabezado'>Población</div>";
strResultado += "<div class='celda encabezado'>Teléfonos</div>";
strResultado += "<div class='celda encabezado'>E-mail</div>";
strResultado += "</div>";
while (reader.Read())
{
    strResultado += "<div class='fila'>";
    strResultado += "<div class='celda'>" + reader.GetString(0) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(1) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(2) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(3) + "</div>";
    strResultado += "<div class='celda'>" + reader.GetString(4) + "</div>";
    strResultado += "</div>";
    inNumeroFilas++;
}
strResultado = strResultado + "</div>";
strResultado = strResultado + "<p>Número Filas: " + inNumeroFilas + "</p>";
lblResultado.Text = strResultado;
}
```

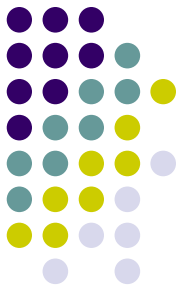


Recuperar (Seleccionar) y presentar datos

Ejemplo 2. Procesamiento mejorado y recomendado (continuación)



```
else
{
    lblMensajes.Text = "No existen registros resultantes de la consulta";
}
reader.Close();
}
catch (SqlException ex)
{
    strError = "<p>Se ha producido errores en el acceso a la base de datos.</p>";
    strError = strError + "<div>Código: " + ex.Number + "</div>";
    strError = strError + "<div>Descripción: " + ex.Message + "</div>";
    lblMensajes.Text = strError;
    return;
}
}
```



Modificar (Insertar, Eliminar y Actualizar) datos

Ejemplo 3. Inserción de un nuevo cliente

Nuevo Cliente

Código Cliente

Nombre

Dirección

Población

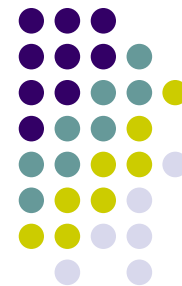
Código postal

Teléfono

Correo electrónico (Login)

Tema 6. Técnicas de Acceso a datos (II)
Ejemplo de Teoría sobre base de datos Tienda.mdf

5.8 Manipulación de datos con ADO.NET



Modificar (Insertar, Eliminar y Actualizar) datos

Ejemplo 3. Inserción de un nuevo cliente

```
protected void cmdInsertar_Click(object sender, EventArgs e)
{
    lblMensajes.Text = "";
    string StrError;
    string strCadenaConexion =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;

    string strComandoSql = "INSERT CLIENTE " +
        "(IdCliente,NomCli,DirCli,PobCli,CpoCli,TelCli,CorCli) VALUES (" +
        "'" + txtIdCliente.Text + "'," + txtNomCli.Text +
        "'," + txtDirCli.Text + "'," + txtPobCli.Text +
        "'," + txtCpoCli.Text + "'," + txtTelCli.Text +
        "'," + txtCorCli.Text + "');";

    using (SqlConnection conexion = new SqlConnection(strCadenaConexion))
    {
        try
        {
            conexion.Open();
            SqlCommand comando = conexion.CreateCommand();
            comando.CommandText = strComandoSql;
        }
    }
}
```

Acceso a los
valores de los
controles TextBox

5.8 Manipulación de datos con ADO.NET



Modificar (Insertar, Eliminar y Actualizar) datos

Ejemplo 3. Inserción de un nuevo cliente (continuación)

Ejecución de la
sentencia INSERT
de SQL

```
int inRegistrosAfectados = comando.ExecuteNonQuery();
if (inRegistrosAfectados == 1)
{
    lblMensajes.Text = "Registro insertado correctamente";
}
else
{
    lblMensajes.Text = "Error al insertar el registro";
}
}
catch (SqlException ex)
{
    StrError = "<p>Se ha producido errores en el acceso a la base de datos.</p>";
    StrError = StrError + "<div>Código: " + ex.Number + "</div>";
    StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
    lblMensajes.Text = StrError;
    return;
}
}
```



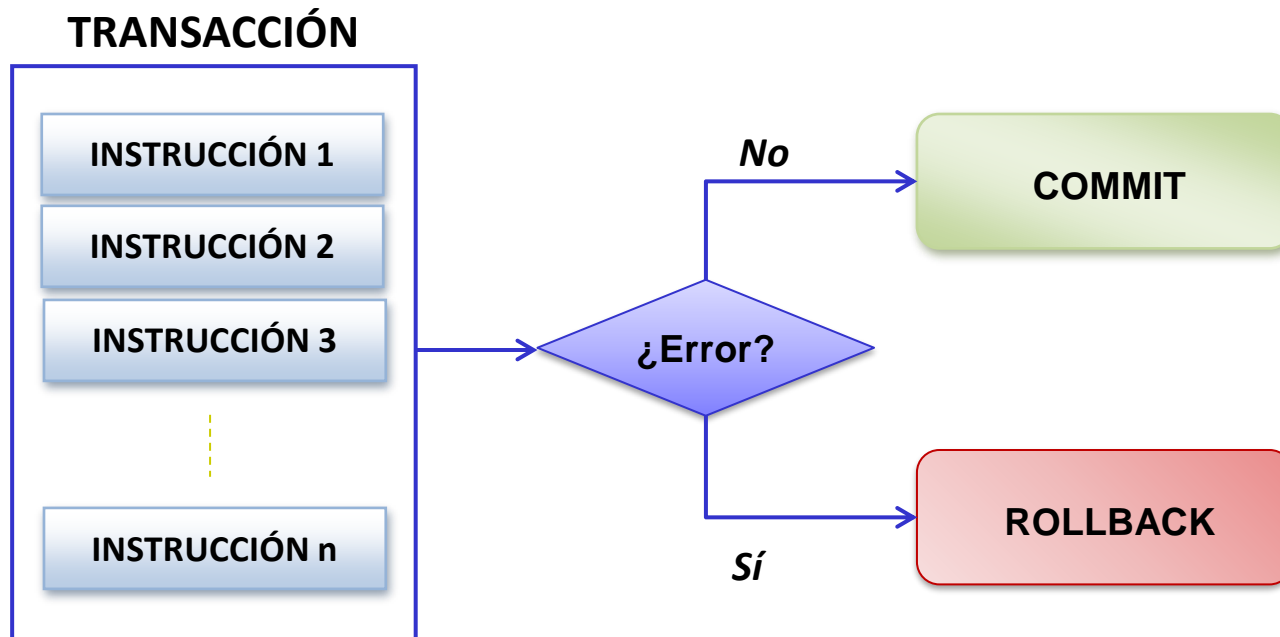
Modificar (Insertar, Eliminar y Actualizar) datos

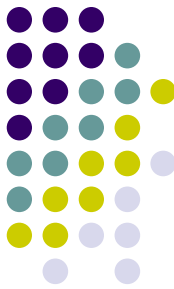
- Otros procesamientos de modificación de datos muy similares al del ejemplo 3 de inserción de registros, son:
 - **Eliminar** un registro existente, empleando una sentencia de SQL de tipo DELETE
 - **Actualizar** los campos de un registro existente, empleando una sentencia de SQL de tipo UPDATE
- En los procesamientos para eliminar o actualizar datos:
 - Se usa, igualmente, el método **ExecuteNonQuery()** del objeto SqlCommand
 - Las diferencias en los procesos de eliminación y actualización, provienen de:
 - El diseño de la presentación sobre la interfaz Web
 - La definición del valor de tipo string que especifica la sentencia SQL a ejecutar
 - Al tratarse de operaciones sobre registros existentes, es necesario:
 - Seleccionar, primeramente, el registro concreto que va a ser eliminado o actualizado
 - En segundo lugar, presentar los datos del registro a eliminar o a actualizar sobre la interfaz Web para poder confirmar la eliminación, o bien para poder editar los valores de los campos a actualizar, respectivamente



Concepto de Transacción

- Una transacción es un conjunto de instrucciones de manipulación de datos que se ejecutan de manera atómica, es decir, como si se tratara de una única instrucción
 - Todas las instrucciones de SQL (DML) que forman una transacción se ejecutan en una base de datos como una única instrucción, de manera que todas y cada una de ellas se **confirman (COMMIT)** o se **deshacen (ROLLBACK)** en grupo





Características de una transacción

- Las características que definen los objetivos de una transacción perfecta (Prueba o Test ACID) son las siguientes:
 - **Atomicidad (*Atomicity*)**. El grupo de instrucciones debe comportarse como un todo o ninguno. Si alguna de ellas no se pudiera ejecutar se debería poder revertir el conjunto de instrucciones ejecutadas
 - **Consistencia (*Consistecy*)**. Al confirmarse una transacción, la base de datos debe quedar en un estado consistente. Cualquier transacción llevará a la base de datos desde un estado válido a otro estado también válido.
 - **Aislamiento (*Isolation*)**. Los datos modificados durante una transacción no deben ser visibles para otros usuarios o procesos hasta que finalice y se hayan realizado todos los cambios de manera definitiva.
 - **Durabilidad (*Durability*)**. Los efectos de una transacción se almacenan de forma permanente

5.9 Transacciones

Ejemplo



```
SqlConnection con = new SqlConnection();
con.ConnectionString = "Data Source=MYCOMPUTER\\SQLEXPRESS;Initial " +
    "Catalog = Sample; Integrated Security = True";

SqlTransaction tran = null;
try
{
    con.Open();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = con;
    tran = con.BeginTransaction();
    cmd.Transaction = tran;
    cmd.CommandText = "Insert into VentaDetalle Values('" + w_cod + "'," +
        "'Polar','Dotze'," + w_cant + ")";
    cmd.ExecuteNonQuery();
    w_stock = w_stock - w_cant;
    cmd.CommandText = "Update Producto Set Stock=" + w_stock +
        " Where Cod='" + w_cod + "'";
    cmd.ExecuteNonQuery();
    tran.Commit();
    Label1.Text = "La transacción se ha realizado con éxito";
}
catch
{
    if (tran != null) tran.Rollback();
    Label1.Text = "La transacción ha fallado";
}
finally
{
    con.Close();
}
```

Define el comienzo de la transacción

Asociar a transacción al comando

Confirma la transacción

Deshace la transacción



Seguridad en aplicaciones Web interactivas

- Aspectos generales. **Funciones de seguridad principales:**
 - **Autenticación.** Ayuda a comprobar que un usuario es, precisamente, quien dice ser
 - La aplicación Web obtiene las credenciales (nombre y contraseña) de un usuario para su identificación, y las valida consultando a una autoridad determinada
 - Si las credenciales son válidas, la identidad que ha enviado las credenciales se considera como una identidad autenticada (o autenticada)
 - **Autorización.** Limita los derechos de acceso a los procesamientos de la aplicación Web mediante la concesión o negación de permisos específicos a una identidad autenticada
- En aplicaciones Web basadas en ASP.NET
 - La autenticación suele resolverse controlando el acceso a la aplicación Web a través de la consulta a una lista de usuarios y grupos que puede estar constituida por:
 - Un conjunto de tablas en la misma base de datos que maneja la aplicación
 - Una base de datos diferente y utilizado ASP.NET Identity como sistema de autenticación
 - La autorización suele resolverse asociando los grupos de usuarios definidos a los roles o casos de uso de la aplicación Web



Seguridad de acceso a datos

- Es uno de los subsistemas de seguridad más importantes de todos los que intervienen en el diseño del sistema de seguridad de una aplicación Web
 - Cuando una aplicación Web incorpora acceso a datos, entonces la aplicación debe proporcionar las credenciales a la base de datos como cualquier otro usuario o proceso
 - Los métodos de seguridad de acceso a datos desde la aplicación Web dependen de los mecanismos de seguridad disponibles en el almacén de datos subyacente
- Algunas formas de diseñar el acceso autenticado a **SQL Server**:
 - Usar la autenticación de Windows para obtener acceso al servidor SQL Server como usuario local de una aplicación de ASP.NET, mediante el usuario NETWORK_SERVICE
 - Es un método seguro y eficiente de acceso a datos almacenados en una base de datos
 - Usar la autenticación de SQL Server, pasando los valores del nombre de usuario y la contraseña de forma explícita en la cadena de conexión
 - Es una forma de acceso menos segura, puesto que los nombres de usuarios y contraseñas de SQL Server se especifican en el código



Seguridad de acceso a datos

- Algunas recomendaciones para proteger el acceso a datos desde una aplicación Web de ASP.NET, son:
 - Conectarse a una instancia de SQL Server usando la **autenticación de Windows** como un usuario local de una aplicación Web de ASP.NET (NETWORK_SERVICE), en lugar de usar la autenticación de SQL Server
 - Se evita exponer el identificador de usuario y la contraseña en la cadena de conexión
 - Asegurarse de que la identidad del proceso que está ejecutando ASP.NET, sea la cuenta predeterminada de ese proceso o una cuenta de usuario restringida
 - Se puede mejorar la seguridad de la aplicación siguiendo las buenas prácticas en materia de codificación y configuración del sistema, también es importante:
 - Comprender cómo interactúan entre sí los diversos subsistemas de seguridad disponibles en un contexto de desarrollo de aplicaciones Web con acceso a datos en el entorno del servidor Web
 - Mantener actualizado el sistema con las últimas actualizaciones de seguridad de Microsoft Windows, Internet Information Services (IIS), de servidor de base de datos utilizado y de cualquier otro software utilizado