Quien programa mejor:
{
~ Lexus
= GT++
~ Jose el del Tupe
}

En C#:
{
~ Es correcta la expresion float a = 3,5;
= Es necesario inicializar las variables
~ La conversion implicita tiene perdida de informacion
}

En C#:
{
= Existe un recolector de basura
~ Debemos destruir las variables manualmente
~ Microsoft dice que esta basado en Java
}

En C#:
{
~ No se pueden manejar excepciones
~ Se puede hacer un try sin catch
= El bloque finally se ejecuta siempre
}

En C#:
{
~ No se puede hacer casting
~ Con la conversion implicita perdemos datos
= Con la conversion explicita perdemos datos
}

En el modelo de capas:
{
= La Interfaz llama al EN
~ El CAD llama al EN
~ El EN llama a la Interfaz
}

Donde se guarda la aplicacion WEB:
{
~ En una Carpeta
= En un Directorio Virtual
~ En LocalHost
}

RAD son las siglas de:
{
= Rapido Desarrollo de Aplicaciones
~ Rapido Analisis y Desarrollo
~ Rapido Diseño de Aplicaciones

}

Es obligatorio establecer la etiqueta runat = "server" en ASP.NET:
{
~ NO
= SI
~ Depende de la etiqueta
}

Al establecer la etiqueta runat = "server":
{
~ El formulario no debe ser procesado en el servidor
~ El formulario lo envia el servidor
= El formulario se debe procesar en el servidor
}

Un proyecto WEB:
{
= Sirve para aplicaciones web avanzadas
~ Sirve para paginas web sencillas
~ No podemos referenciar DLLs
}

Las paginas maestras:
{
~ Solo puede haber 1 pagina maestra
~ Proporcionan seguridad a la pagina
= Proporcionan coherencia a la pagina
}

En las master pages (paginas maestras):
{
= Definimos el contenido comun y los contenedores de contenido
~ Hacemos referencia a otras paginas maestras
~ Creamos el contenido de los contenedores
}

El enlace entre el mensaje del evento y el mensaje especifico se lleva a cabo mediante:
{
~ Emisor de eventos
= Delegado de eventos
~ Receptor de eventos
}

Los delegados son:
{
= Puntero a funcion
~ Puntero a mensaje
~ Puntero a destino
}

Un Boton es de tipo:
{

~ No Postback
~ Evento de cache
= Evento de envio
}

Un Textbox es de tipo:
{
~ Evento de cache
~ No Postback
= Ambas son Correctas
}

Se puede transformar No Postback a Postback:
{
~ Los eventos No Postback ya son Postback
= Asignando AutoPostback = true
~ Asignando PostBack = true
}

Se puede generar estilo de web sin usar archivo CSS:
{
= Si, con codigo inline
~ No, tiene que existir este archivo
~ Ninguna de las anteriores
}

Con el comando Response.Redirect():
{
= Podemos enviar parametros dentro de la web
~ Solo podemos acceder a paginas en el directorio raiz
~ Ninguna de las anteriores
}

Para leer parametros en la Web usamos:
{
~ Commander
= Request
~ Application
}

Un menu estatico es aquel que:
{
~ Son estaticas las porciones especificadas
~ El control del menu es variable
= El control del menu esta expandido completamente
}

Un menu dinamico es aquel que:
{
= Son estaticas las porciones especificadas
~ Toda la estructura es siempre visible
~ El control del menu esta espandido completamente
}

Con RequiredFieldValidator Validamos:
{
~ La cadena sea numerica
= La cadena no sea vacia
~ La cadena sea alfabetica
}

Los controladores de validacion pueden ser:
{
~ Estaticos
~ Dinamicos
= Los dos
}

En los controladores de validacion (marcar la falsa):
{
= No se puede controlar un rango de datos
~ Se puede controlar una expresion
~ Podemos definir nuestro propio control
}

La validacion de un correo electronico es:
{
~ \D+@\D+\.\D+
= \S+@\S+\.\S+
~ \w+@\w+\.\w+
}

En el objeto sesion:
{
~ Al acabar el timeout se cierra sesion
~ Al cerrar el navegador se borran los datos
= Ambas son correctas
}

Para cerrar sesion usamos:
{
= Session.Abandon
~ Session.Remove
~ Session.Close
}

Cuales estan implementados como colecciones o conjuntos de pares nombre-valor:
{
~ Session
~ Application
= Ambas
}

Si queremos asignar variables para todos los usuarios usaremos:
{
~ Session
= Application
~ Ambas

}

Las variables application pueden ser bloqueadas:
{
= Si
~ No
~ Depende del criterio a seguir
}

El archivo Global.asax:
{
~ Contiene etiquetas HTML
~ Contiene etiquetas XML
= No contiene etiquetas ASP.NET
}

Los cambios en Global.asax:
{
= Requieren reinicio de aplicacion
~ Requieren compilacion de aplicacion
~ Se ejecutan con normalidad
}

En el entorno conectado usaremos:
{
~ Dataset
= Datareader
~ Dataconect
}

En el entorno desconectado usaremos:
{
= Dataset
~ Datareader
~ Dataconect
}

Los Objetos Connection y Command:
{
= Tienen prefijo
~ No tienen prefijo
~ Tienen prefijo, como dataset
}

El archivo web.config:
{
~ Basado en ASP.NET
~ Basado en HTML
= Basado en XML
}

Para realizar un SELECT:
{
~ ExecuteNonQuery

~ ExecuteCommand
= ExecuteReader
}

Con DataAdapter:
{
~ Debemos gestionar en todo momento la apertura y cierre de la BBDD
= La gestion por abrir y cerrar la BBDD es automatica
~ La gestion por abrir y cerrar la BBDD es automatica utilizando CommandBuilder
}

Con DataAdapter:
{
~ Debemos gestionar en todo momento la apertura y cierre de la BBDD
~ La gestion por insertar, borrar y actualizar la BBDD es automatica
= La gestion por insertar, borrar y actualizar la BBDD es automatica utilizando CommandBuilder
}

GridView:
{
~ Con asistente
~ Con codigo
= Los dos
}

La concurrencia pesimista es:
{
= Cuando una fila es leida, esta queda bloqueada para su lectura para cualquier otro que la demande hasta que aquel que la posee la libere.
~ Las filas estan disponibles para su lectura en todo momento, estas pueden ser leidas por distintos usuarios al mismo tiempo.
~ Esta tecnica implica que no existe control. El ultimo cambio en escribirse es el que permanece.
}

La concurrencia last win es:
{
~ Cuando una fila es leida, esta queda bloqueada para su lectura para cualquier otro que la demande hasta que aquel que la posee la libere.
~ Las filas estan disponibles para su lectura en todo momento, estas pueden ser leidas por distintos usuarios al mismo tiempo.
= Esta tecnica implica que no existe control. El ultimo cambio en escribirse es el que permanece.
}

La concurrencia positiva es:
{
~ Cuando una fila es leida, esta queda bloqueada para su lectura para cualquier otro que la demande hasta que aquel que la posee la libere.
= Las filas estan disponibles para su lectura en todo momento, estas pueden ser leidas por distintos usuarios al mismo tiempo.
~ Esta tecnica implica que no existe control. El ultimo cambio en escribirse es el que permanece.

}

La concurrencia optimista es:
{
= Una cota de ADA que nos raya en este tipo de preguntas
~ Las filas estan disponibles para su lectura en todo momento, estas pueden ser leidas por distintos usuarios al mismo tiempo.
~ Esta tecnica implica que no existe control. El ultimo cambio en escribirse es el que permanece.
}

Cuando tenemos que hacer un acceso complicado usaremos:
{
~ DataReader
~ DataAdapter
= DataSet
}

Si trabajamos con mas de una BBDD usaremos:
{
= DataSet
~ DataReader
~ DataAdapter
}

En las Cookies:
{
~ Los datos se borran siempre cuando el usuario cierra la ventana del navegador
= Las cookies no se pierden cuando se cierra el navegador (a no ser que el usuario las borre)
~ Una cookie se representa por la clase Cookie
}

En las Cookies:
{
~ Los datos se borran siempre cuando el usuario cierra la ventana del navegador
~ Las cookies no se pueden borrar con fechas ya expiradas
= Una cookie se representa por la clase HttpCookie
}

La extension de los controles de usuario es:
{
~ aspx
= ascx
~ asdx
}