

Hada T1: Control de versiones

git: Un sistema de control de versiones distribuído.

Objetivos del tema

- Conocer los conceptos básicos asociados a los *Sistemas de Control de Versiones*.
- Aprender a usar **git** localmente de manera individual.
- Aprender a usar **git** en grupo
- Conocer y saber cómo usar **git** con **github** (en remoto).



Usar un Sistema de
Control de Versiones
es como tener una
máquina del tiempo

para tus documentos/código



Mucha gente utiliza un sistema de control de versiones muy básico

- Normalmente la gente guarda diferentes versiones del mismo archivo

miprograma.c / miprogramav2.c / programaFINAL.c / programaFINALfinal.c

- o incluso usa fechas:

miprograma2018-12-02.c / miprograma2018-12-03.c

¿POR QUÉ QUEREMOS GUARDAR VERSIONES ANTERIORES?

EN GRUPOS (1 minuto)

- **¿CUÁLES SON LOS PROBLEMAS QUE NOS PODEMOS ENCONTRAR USANDO ESTE MÉTODO?**
- **¿qué pasa cuando se trabaja colaborativamente en un documento?**



Problemas

1. **CUÁLES** son los cambios entre las diferentes versiones
2. **CUÁNDO** se hizo un cambio concreto



Por lo tanto... ¿cómo podemos recuperar una versión concreta de un archivo?



Cuando trabajamos colaborativamente...



1. ¿Cómo **comunicamos los cambios** que hemos hecho a las otras personas del grupo?
2. ¿Cómo podemos **mezclar/fusionar diferentes versiones**?
3. ¿Cómo sabemos **QUIÉN** hizo una modificación concreta?

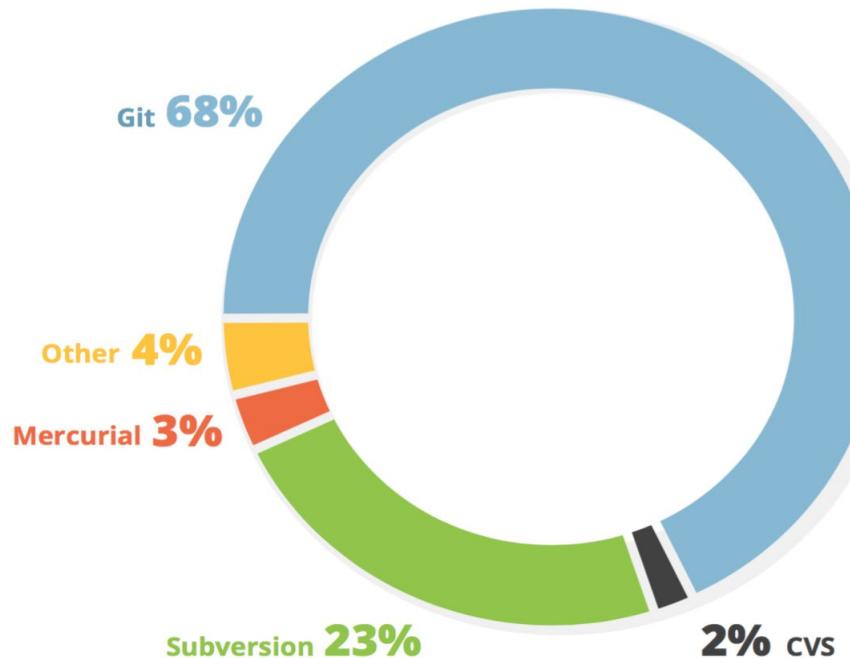
Documentos/Código colaborativo



¡NO SON SISTEMAS DE CONTROL DE VERSIONES!

Historial de versiones ⇒ temporal (en dropbox los últimos 30),
cambios imprecisos, solo para documentos (no código)
a veces es suficiente

Sistemas de Control de Versiones (scv) más comunes



SCV



- Documentos
- Proyectos software: lenguajes de programación, conjuntos de datos...
- Más control (comandos específicos).
- Versión de la historia más útil.

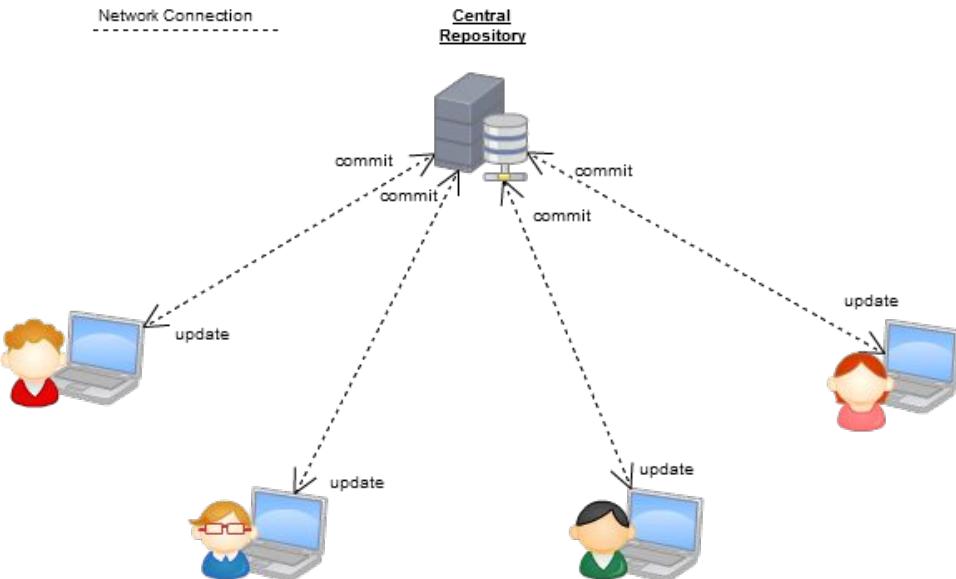


Git: repositorio local

Por lo tanto,
¿qué nos proporcionan
los *sistemas de
control de versiones*
(SCV)?

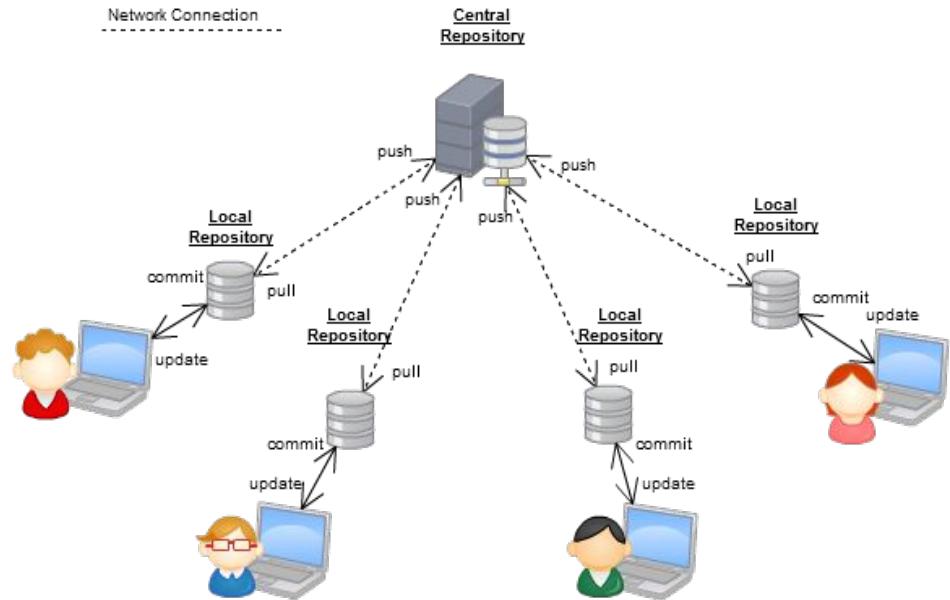
- Gestionar automáticamente los cambios que se realizan sobre uno o varios ficheros de un proyecto.
- Restaurar cada uno de los ficheros de un proyecto a un estado anterior (no solo al inmediatamente anterior).
- Permitir la colaboración de diversos programadores en el desarrollo de un proyecto.

SCV Centralizado



- Repositorio único y centralizado respecto al que se sincronizan los clientes
- Problemas:
 - un único punto de fallo
 - no permite trabajar sin conexión
- Similar a trabajar con Dropbox
 - ¡OJO! Dropbox no es un SCV

SCV Distribuido



- Cada cliente tiene su repositorio local como una copia maestra remota (previniendo la pérdida de datos)
- Puedes trabajar sin conexión (siempre tienes tu repositorio contigo)

Conceptos generales de los SCV I

- **Repositorio:** Es la copia maestra donde se guardan todas las versiones de los archivos de un proyecto. En el caso de git se trata de un directorio. Cada desarrollador tiene su propia copia local de este directorio.
- **Copia de trabajo:** La copia de los ficheros del proyecto que podemos modificar.

Conceptos generales de los SCV II

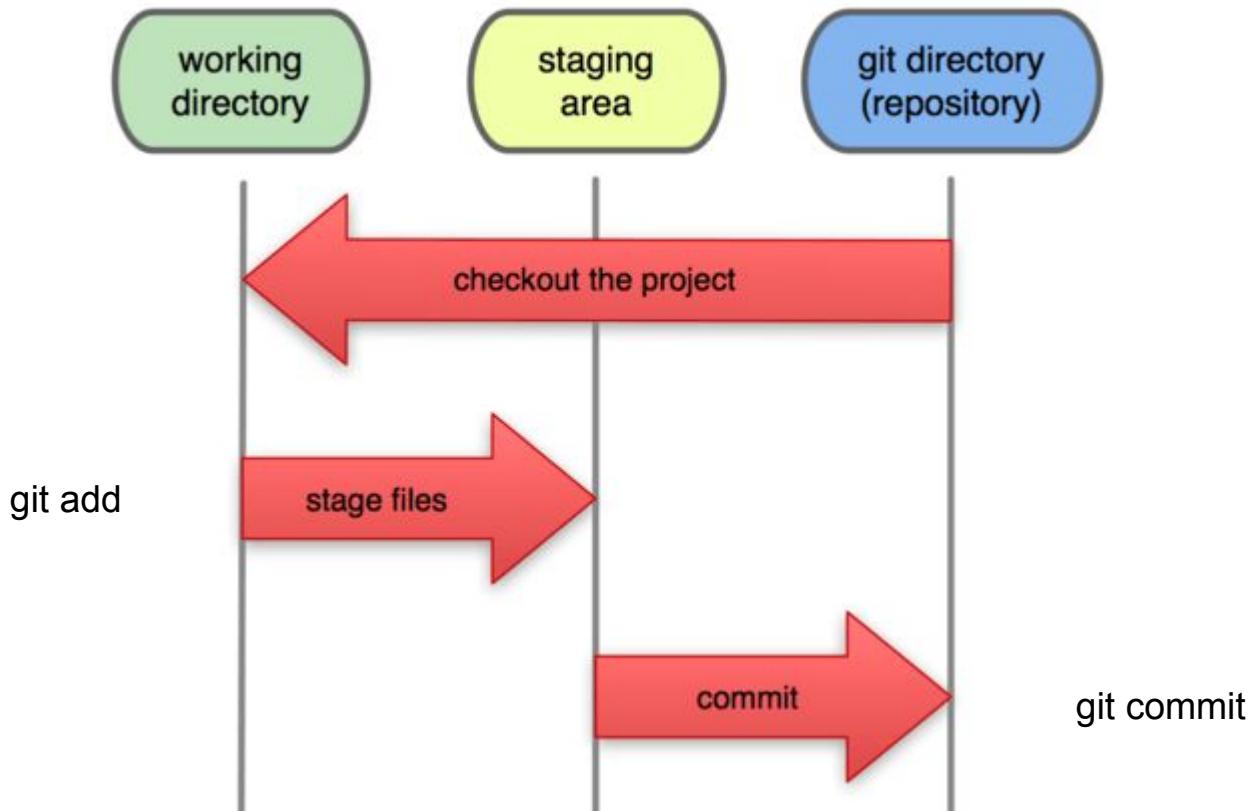
Check Out / Clone: La acción empleada para obtener una copia de trabajo desde el repositorio. En los scv distribuidos -como Git- esta operación se conoce como *clonar* el repositorio porque, además de la copia de trabajo, proporciona a cada programador su copia local del repositorio a partir de la *copia maestra* del mismo.

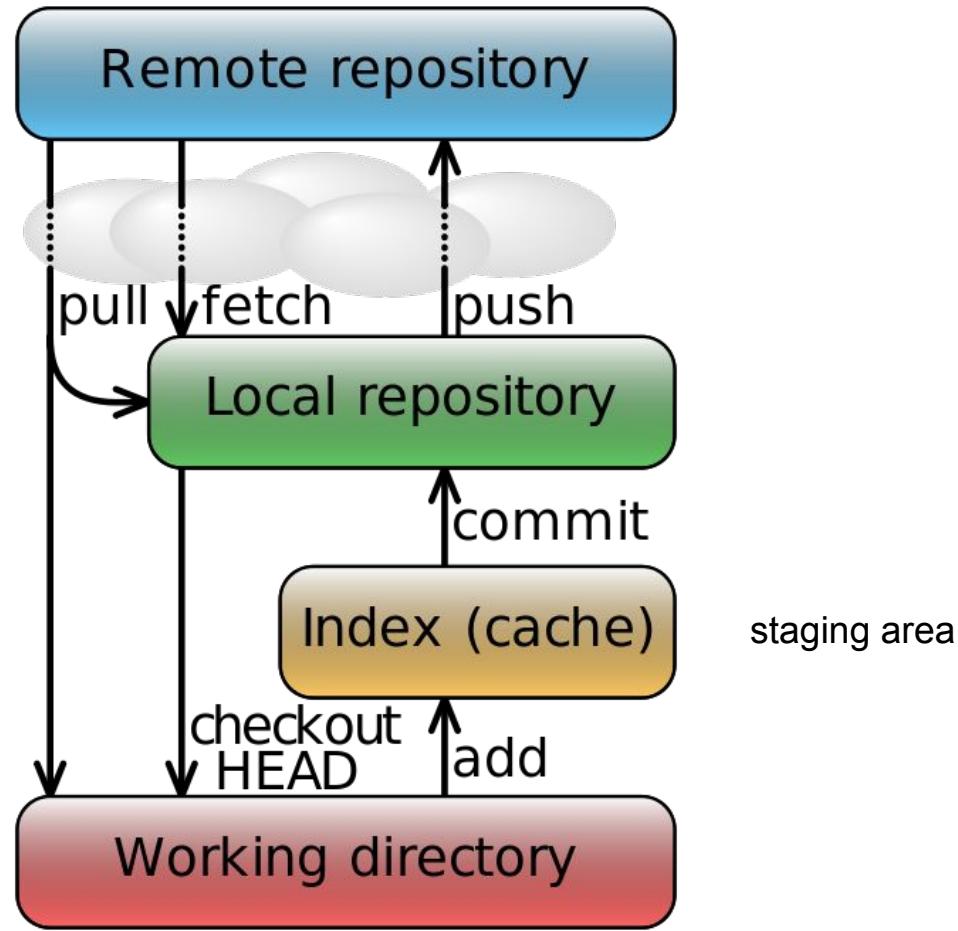
Check In / Commit: La acción empleada para llevar los cambios hechos en la copia de trabajo a la copia local del repositorio. Esto crea una nueva *revisión* de los archivos modificados. Cada commit debe ir acompañado de un “Log Message” el cual es un comentario, una cadena de texto que explica el commit, que añadimos a una revisión cuando hacemos el commit.

Conceptos generales de los SCV III

- **Push:** La acción que traslada los contenidos de la copia local del repositorio de un programador a la copia maestra del mismo.
- **Update/Pull/Fetch+Merge/Rebase:** Acción empleada para actualizar nuestra copia local del repositorio a partir de la copia maestra del mismo, además de actualizar la copia de trabajo con el contenido actual del repositorio local.
- **Conflicto:** Situación que surge cuando dos desarrolladores hacen un *commit* con cambios en la *misma región del mismo fichero*. El **scv** lo detecta, pero es el programador el que debe corregirlo.

Local Operations





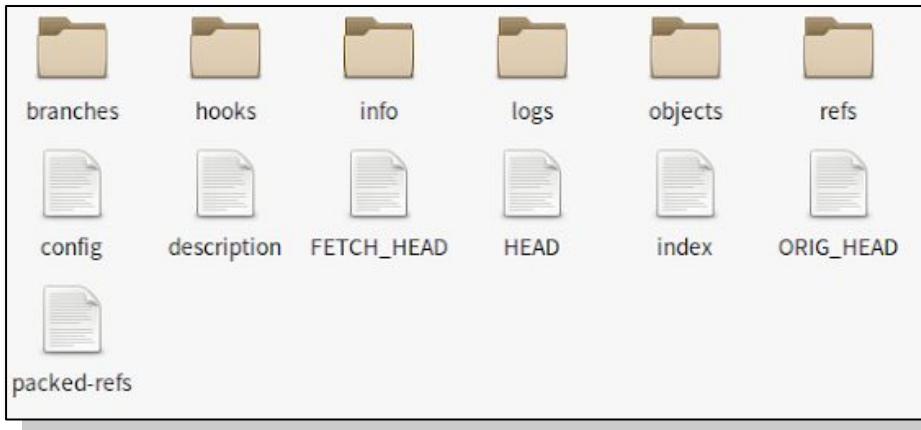
Git: Historia

- Los desarrolladores de *linux* emplean BitKeeper hasta 2005.
- BitKeeper es un **scv** distribuído. Git también lo es, al igual que Darcs, Mercurial, SVK, Bazaar y Monotone.
- Linus comienza el desarrollo de *git* el 3 de abril de 2005, lo anunció el día 6 de abril.
- Git se auto-hospeda el 7 de abril de 2005.
- El primer núcleo de *linux* gestionado con *git* se libera el 16 de junio de 2005, fue el **2.6.12**.

Git: Implementación

- La parte de bajo nivel ([plumbing](#)) se puede ver como un sistema de ficheros direccionable por el contenido.
- Por encima incorpora todas las herramientas necesarias que lo convierten en un **scv** más o menos amigable ([porcelain](#)).
- Cuenta con aplicaciones escritas en **C** y en **shell-script**. Con el paso del tiempo algunas de estas últimas se han reescrito en **C**.
- Los elementos u objetos en los que *git* almacena su información se identifican por su valor [SHA-1](#).

Git: Directorio .git



Uso I

- La orden principal es **git**.
 - Comprobamos qué versión tenemos instalada:
 > **git --version**
 git version 2.8.0
- Creamos un repositorio, se puede hacer de dos modos:
 1. > **mkdir Proyecto; cd Proyecto; git init**
 2. > **mkdir Proyecto; git init Proyecto**
- Añadimos archivos y guardamos:
 1. > **git add .**
 2. > **git status**
 3. > **git commit -m 'Primer commit.' -m "Descripcion detallada."**
 4. > **git commit -a**

Uso II

- Configuración:

- archivo “`.git/config`” ⇒ particular del proyecto actual

1. `> git config user.name "nombre apellidos"`

2. `> git config user.email "usuario@email.com"`

- archivo “`~/.gitconfig`” ⇒ general para todos los proyectos del usuario

3. `> git config --global user.name "nombre apellidos"`

4. `> git config --global user.email "usuario@email.com"`

Información

- **Git status:**
 - muestra el estado del directorio de trabajo y el index/staging area/caché
 - estado de los archivos:
 - untracked: no incluidos en el repositorio bajo control de versiones
 - **committed: confirmados** y almacenados en tu copia local del repositorio
 - modified: modificados respecto a la copia local del repositorio
 - (un)staged: (no) preparados para enviar a tu copia local del repositorio

Información II

- **Git log:**
 - muestra el estado del repositorio (confirmaciones - commits - en orden cronológico inverso)
 - -p: diferencias entre confirmaciones (commits)
- **Git show:**
 - muestra el último commit (confirmación) con diferencias detalladas
- **Git diff:**
 - muestra las diferencias entre el directorio de trabajo y el commit (confirmación) más reciente

Etiquetas

- Podemos etiquetar confirmaciones (commits) para marcar puntos importantes
 - Por ejemplo, puntos donde se crea una versión nueva (v1.0, v1.1, etc.)
-
- > git tag tagname commit //crear etiqueta
 - > git tag -a -m “mensaje” tagname commit //etiqueta anotada
 - > git tag -l //lista todas las etiquetas
 - > git tag -d tagname //elimina una etiqueta
 - > git show tagname
//muestra la información de la etiqueta y el objeto de referencia

Descartar cambios

1. Si por error has subido al repositorio algún archivo o directorio que no debería estar, puedes eliminarlo con el siguiente comando, que mantiene la copia local en tu ordenador de ese archivo o directorio:

```
git rm --cached nombre-del-archivo-o-directorio
```

2. O puedes deshacer cambios de archivos con seguimiento desde la última confirmación (el último git commit)

```
git reset --hard
```

3. O puedes traer cambios de archivos y ramas específicas

```
git checkout path-to-file or branch
```

Cómo organizar un repositorio - ¿Qué NO incluir? I

- No incluyas archivos que se puedan generar a partir de otros ya incluidos. Estos archivos incluyen binarios, ejecutables (.exe), etc.
 - En los proyectos de visual studio, este tipo de archivos se almacenan en las carpetas bin y obj
 - Tampoco los archivos de proyecto (.csproj), archivos de solución (.sln).
- No incluyas la BBDD
- Archivos de configuración, especialmente aquellos que puedan cambiar de un entorno a otro o por cualquier razón

Cómo organizar un repositorio - ¿Qué NO incluir? II

- Archivos binarios grandes
 - Grande con respecto a la cantidad de memoria RAM libre disponible
 - Evita archivos de más de 1GB
- Evita crear repositorios muy grandes (cuando sea posible)
 - Estos repositorios ralentizan muchas operaciones de git por la cantidad de archivos y por el tamaño de los mismos
 - Grande depende del tamaño de la RAM y otros factores.
- Más consejos en:
<https://sethrobertson.github.io/GitBestPractices/#misc>
<https://sethrobertson.github.io/GitBestPractices/#donot>

.gitignore

- Se puede definir en el archivo .gitignore los ficheros y carpetas que queremos NO incluir (y por tanto no rastrear) en el repositorio
- Ejemplo de .gitignore

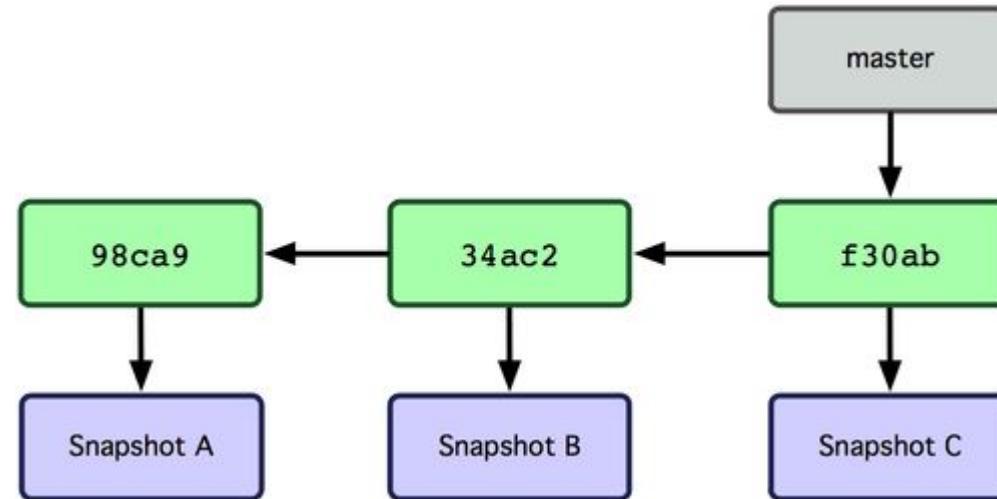
```
#archivos de la solución  
*.sln
```

```
#carpeta Debug  
[Dd]ebug/
```

```
#archivo de BBDD  
*.mdf
```

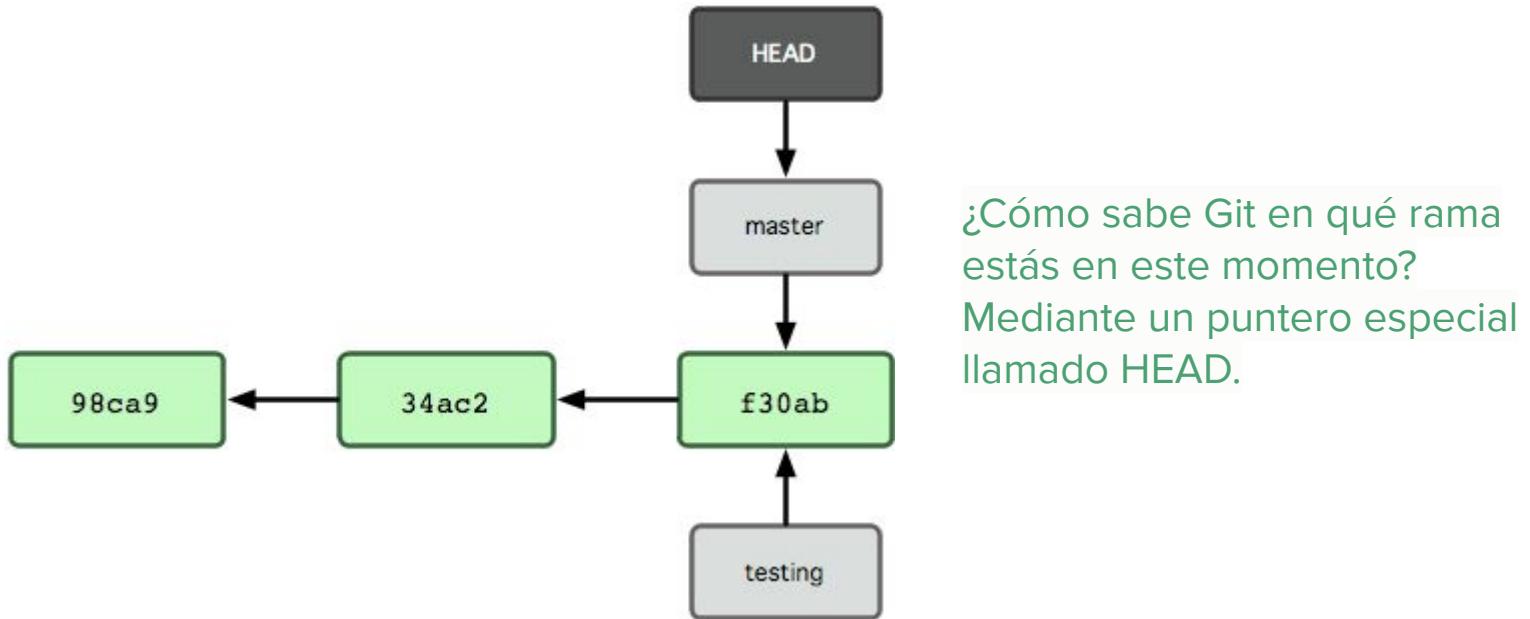
Ramas I

La rama por defecto se llama **master**. Con los primeros cambios confirmados, la rama principal **master** se creará apuntando a esta confirmación (=commit). Por cada commit que hagamos, la rama avanzará automáticamente. La rama **master** siempre apuntará la última confirmación (commit) realizado.



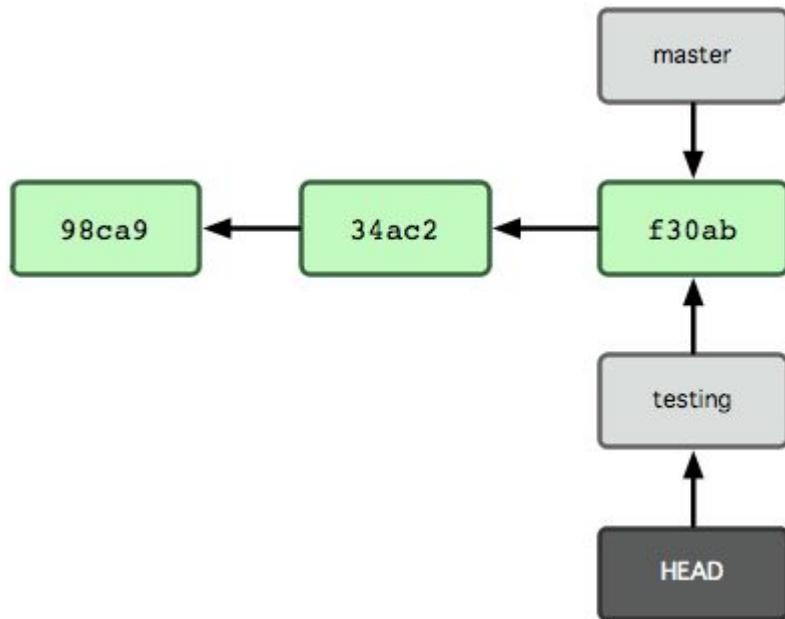
Ramas II

- **git branch testing:** Este comando creará un nueva rama llamada **testing** con un puntero en a la misma confirmación (commit) donde estés actualmente.



Ramas III

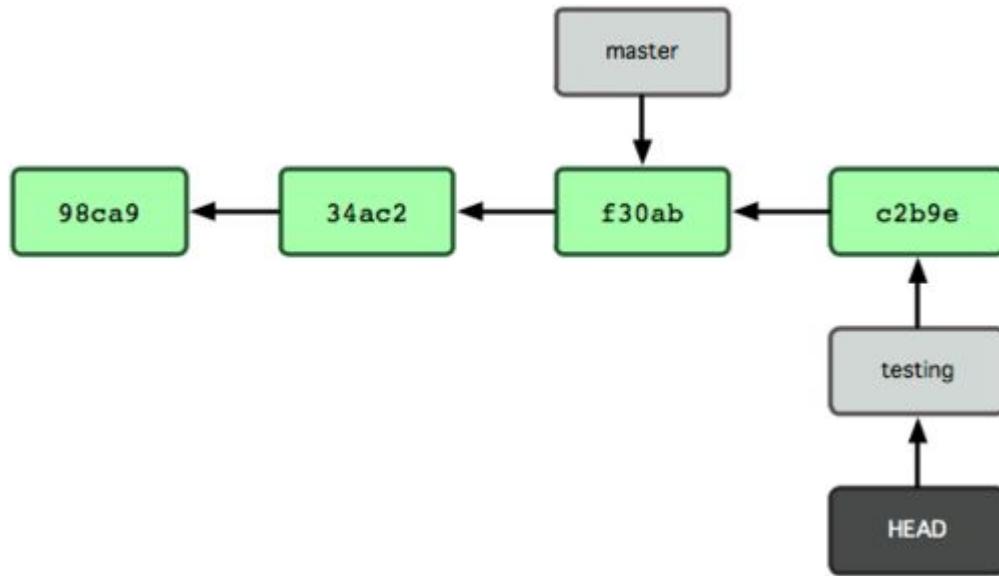
- **git checkout testing:** para saltar de una rama a otra



Ahora hemos cambiado a la rama testing, que acabamos de crear.
Por lo tanto, el puntero HEAD apunta a la rama testing.

Ramas IV

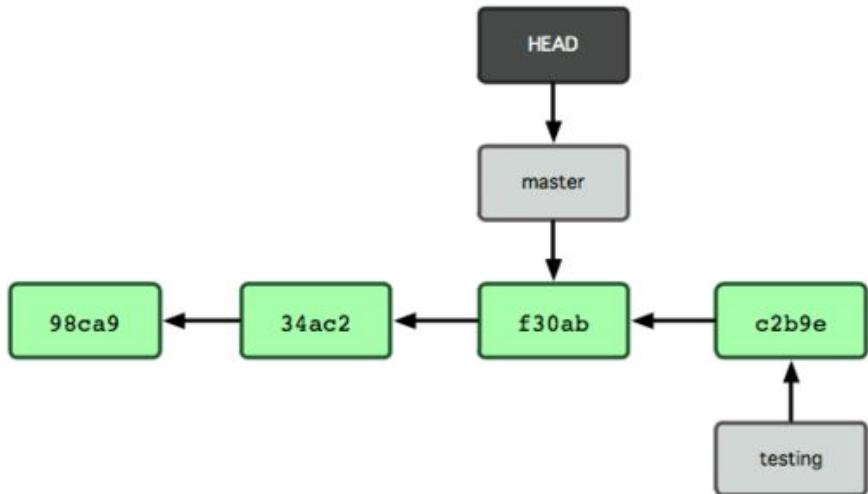
- Si hacemos modificaciones en la rama testing



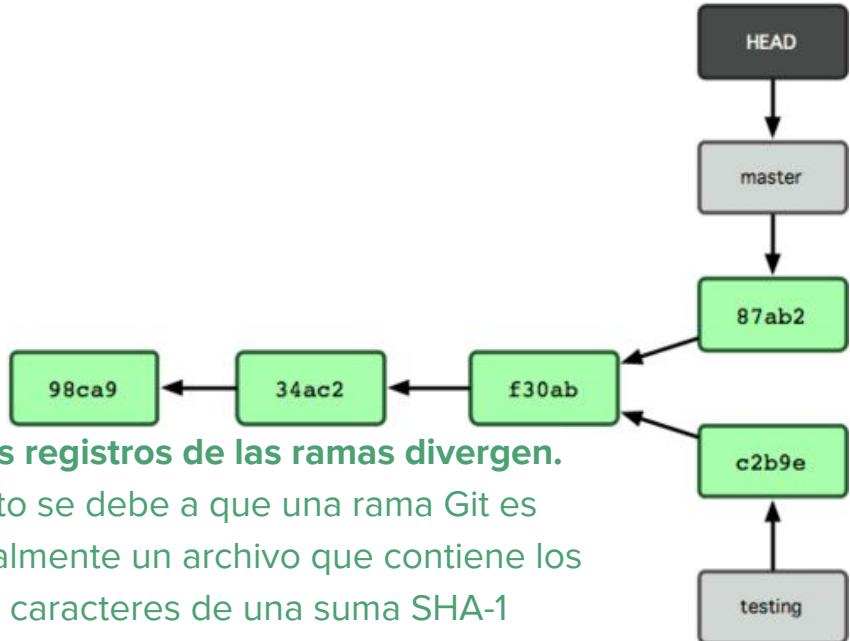
La rama testing avanza
La rama master se mantiene en
el mismo lugar ⇒ confirmación
(commit)

Ramas V

- Cambiamos a master



- Cambiamos a master
y modificamos

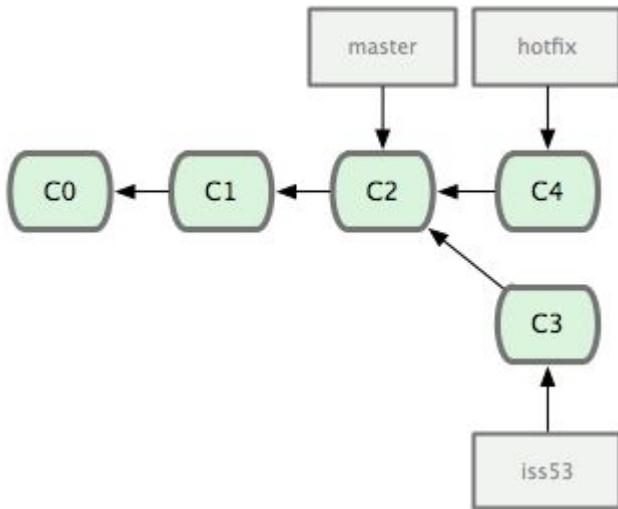


Los registros de las ramas divergen.

Esto se debe a que una rama Git es realmente un archivo que contiene los 40 caracteres de una suma SHA-1 (representando la confirmación de cambios a la que apunta), por eso no cuesta nada crear y eliminar ramas en Git.

Ramas VI

- **git checkout master**
- **git merge hotfix**

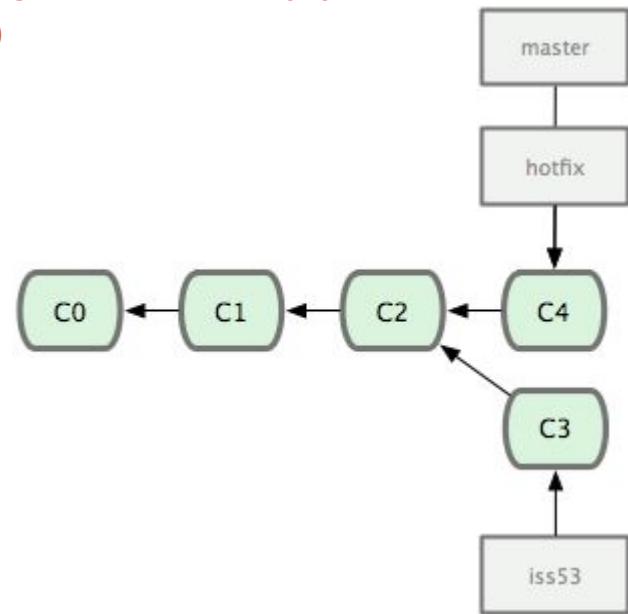


Updating f42c576..3a0874c

Fast forward

README | 1 -

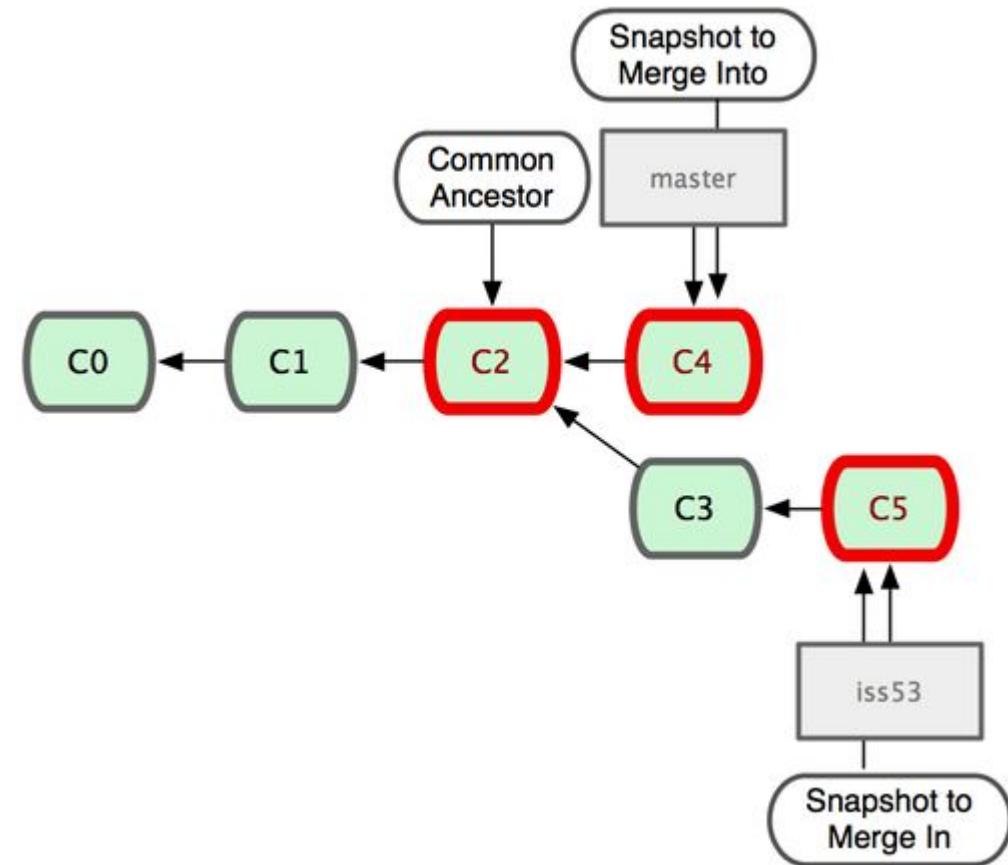
1 files changed, 0 insertions(+), 1 deletions(-)



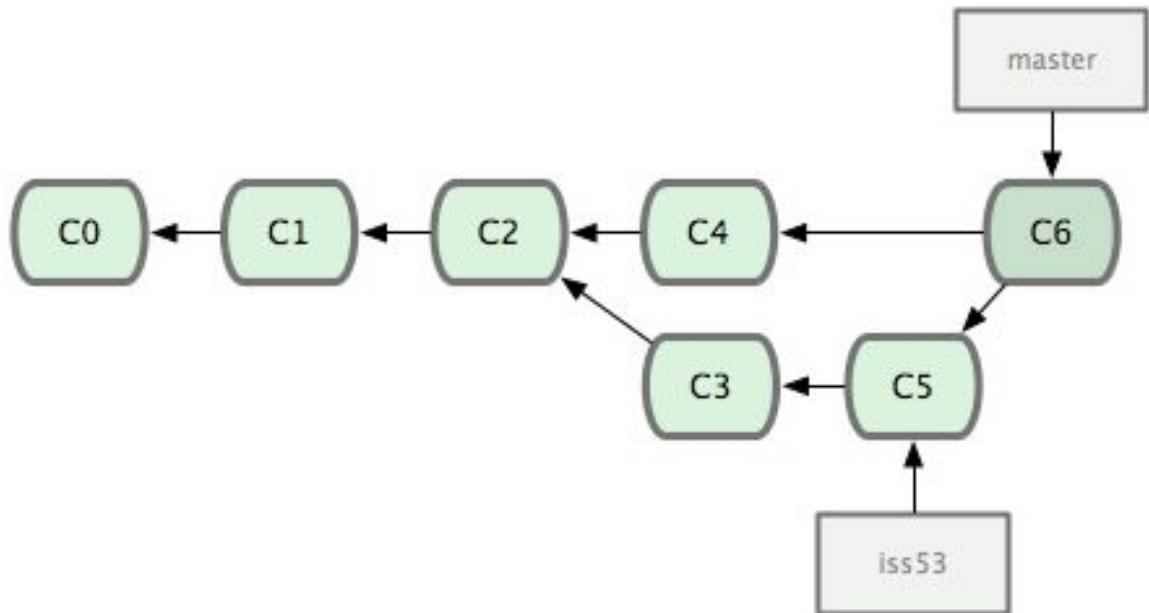
Git ha movido el puntero hacia delante, ya que la confirmación de la rama master (commit C2) apuntaba a la siguiente confirmación (commit C4, rama hotfix)

Ramas VII

Git automáticamente identifica el ancestro común para realizar la fusión de ramas



Ramas VIII



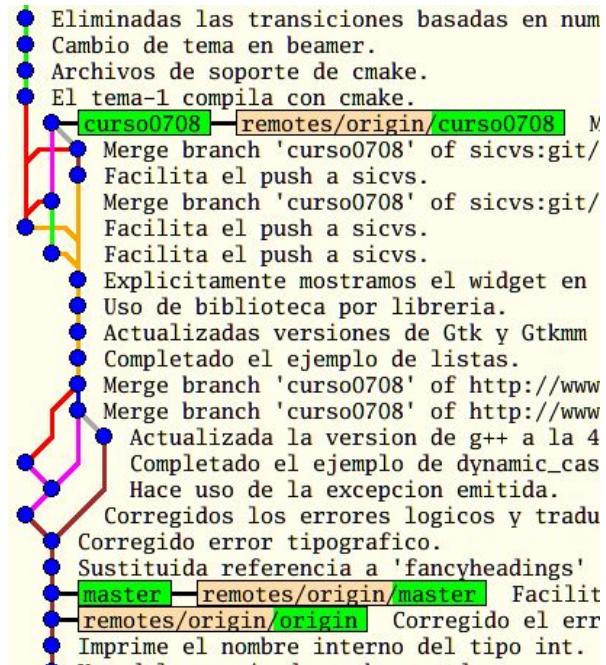
Ahora que todo tu trabajo está ya fusionado con la rama principal, ya no necesitas la rama iss53. Por lo que puedes borrarla.

En lugar de simplemente avanzar el puntero de la rama, **Git crea una nueva instantánea (snapshot) resultado de la fusión a tres bandas; y automáticamente crea una nueva confirmación de cambios (commit) que apunta a ella**. Nos referimos a este proceso como "fusión confirmada". Y se diferencia en que tiene más de un padre.

Ramas IX

- Comandos

1. > git branch [-a] [-r]
2. > git show-branch
3. > git checkout [-b] [rama-de-partida]
4. > gitk --all



Ramas X

- Reppositorios remotos
 1. > git remote add nombre protocolo
 2. > git remote add origin IP:ruta/hasta/repo
 3. > git clone IP:ruta/hasta/repo
- Operaciones con repositorios remotos
 1. > git pull [origin] [rama]
 2. > git push [repo] [rama]
 3. > git checkout -b rama origin/rama-remota
 4. > git fetch
 5. > git merge
 6. > git pull = git fetch + git merge
 7. > git rebase otra-rama

Uso V

- Stash

```
1. > git stash [list | show | drop | ...]
```

Echa un vistazo a este tutorial sobre [git stash](#).

- Bisect

```
1. > git bisect [help | start | bad | good | ...]
```

Echa un vistazo a este tutorial sobre [git bisect](#).

- Herramientas gráficas

1. [gitk](#)
2. [git gui](#)
3. [git view](#)
4. [gitg](#)
5. [giggle](#)
6. [gource](#)

- Cualquier IDE o editor actual (Atom, Sublime, VisualStudio, VisualStudioCode, etc...) dispone de un plug-in para git.

Interacción con otros SCVs

- Git puede interactuar con otros scvs. Además puede hacerlo en ambos sentidos (recuperar y guardar información).
- Por ejemplo, en los casos de CVS y Subversion echa un vistazo a:
 1. > `git cvsimport --help`
 2. > `git svn --help`

Casos de uso I

- ¿Cómo creo una rama local que *siga* los cambios en una remota al hacer `pull`?
 - `git branch --track ramalocal origin/ramaremota`
- ¿Se puede crear una rama que no parte del último commit de otra?...**sí**
 - `git branch --no-track feature3 HEAD~4`
- ¿Quién hizo qué *commit* en un fichero del proyecto?:
 - `git blame fichero`
- ¿Cómo creo una rama para resolver un bug y lo integro de nuevo en la rama principal?:
 - `git checkout -b fixes`
hack...hack...hack
 - `git commit -a -m "Crashing bug solved."`
 - `git checkout master`
 - `git merge fixes`

Casos de uso II

- He modificado localmente el fichero “src/main.cs” y no me gustan los cambios hechos. ¿Cómo lo devuelvo a la última versión bajo control de versiones?:
 - `git checkout -- src/main.cs`
- ¿Y un directorio completo, p.e. a la *penúltima versión* de la rama “test”?:
 - `git checkout test~1 -- src/`
- ¿Y si he modificado varios ficheros y quiero dejar todo como estaba antes de la modificación?...tenemos varias maneras:
 1. `git checkout -f`
 - o también:
 2. `git reset --HARD`

Casos de uso III

- ¿Se puede deshacer un `commit' que es una mezcla (*merge*) de varios “commits”?...**sí**, hay que elegir cuál o cuáles de los commits que forman la mezcla así:
 1. **git revert HEAD~1 -m 1**

En este ejemplo estaríamos deshaciendo sólo el primero de los “commits” que formaban este “merge”.
- ¿Cómo puedo obtener un archivo tal y como se encontraba en una versión determinada del proyecto?... de varias maneras:
 1. **git show HEAD~4:index.html > oldIndex.html**

o también así:
 2. **git checkout HEAD~4 -- index.html**

Casos de uso IV

- ¿De qué maneras distintas puedo ver los cambios que ha habido en el repositorio?:
 1. `git diff`
 2. `git log --stat`
 3. `git whatchanged`
- ¿Cómo puedo saber cuántos commits ha hecho cada miembro del proyecto en la rama actual?:
 1. `git shortlog -s -n`
- ¿Y en todas las ramas?:
 1. `git shortlog -s -n --all`
- ¿Cómo puedo corregir el mensaje de explicación del último commit que he hecho?:
 1. `git commit --amend`
Abre el editor por defecto y nos permite modificarlo.

Ejercicio sencillo

Lo mejor es que pruebes git con algún código tuyo, p.e. de alguna práctica de otra asignatura que ya tengas hecha. Sigue estos pasos:

1. Elige un directorio que contenga el código de esa práctica. Cambiate a él.
2. Inicia el repositorio en este directorio.
3. Añade los archivos que haya previamente en él.
4. Haz el primer “commit” de los archivos recién importados.
5. Haz una modificación a uno o varios de ellos. Comprueba cuáles han cambiado, cómo lo han hecho. Añádelos al siguiente commit.
6. Contribuye los cambios creando el “commit”.
7. Crea una rama en el proyecto y cámbiate a ella automáticamente (mira las opciones de *checkout*).
8. Haz cambios y commits en esta rama.
9. Vuelve a la rama “master”.

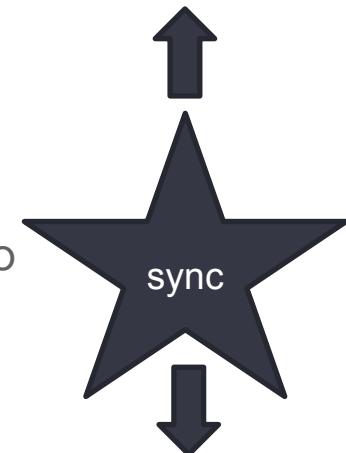
Bitbucket

GitLab
GitHub

Repositorios remotos para git

- Existen varios servicios de repositorio remoto para SCV Git
- Permiten sincronizar el repositorio local con el remoto
- Proveen una interfaz web que trabaja sobre git
- Son una plataforma social para compartir trabajo y conocimiento
- En esta asignatura nos centraremos en GitHub

repositorio remoto



Git: repositorio local

Github I

- Github es una plataforma de desarrollo colaborativo para alojar proyectos software haciendo uso del SCV Git.
- Podemos crear proyectos en ella y almacenarlos de manera gratuita si son:
 - **públicos** ⇒ son visibles para todos y *todo el mundo puede clonarlos*
 - **privados** ⇒ son visibles solo para tí y para los colaboradores que tú elijas, pero únicamente haciendo uso de una *cuenta de pago*.

Github II

- Desde hace un tiempo Github permite hacer uso gratuito de ciertos recursos que normalmente forman parte de la *cuenta de pago*, p.e. poder crear proyectos “privados”.
- Sólo necesitas una cuenta de email institucional de la UA (@alu.ua.es) asociada a la cuenta de Github empleada y darte de alta en [Github-Education](#).

Conflictos I

- Los SCV permiten gestionar las contribuciones uno o varios desarrolladores
- Si los desarrolladores editan el mismo contenido, pueden ocurrir conflictos
 - Cuando trabajas solo, los conflictos pueden aparecer al trabajar en varios ordenadores.
 - Cuando trabajas con varias personas, los conflictos pueden surgir al editar el mismo archivo.
- Para minimizar la aparición de conflictos, cada desarrollador suele trabajar en una rama separada
 - Por eso combinar ramas y lidiar con conflictos son tareas comunes de git

Conflictos II

- La mayor parte del tiempo, el comando `git merge` sabe cómo integrar los cambios entre ramas de forma automática (y sencilla para tí)
- Pero si surge un conflicto, `git` no sabe automáticamente qué cambio elegir
- El comando `git merge` marcará el archivo como conflictivo e interrumpirá el proceso. Es tu responsabilidad como desarrollador resolver ese conflicto.
- Los conflictos se pueden producir al inicio de o durante el proceso de combinar ramas (`git merge`).

Resolver conflictos I

- Git falla al inicio del proceso de combinar ramas (merge)
 - El fallo se produce cuando Git ve diferencias entre el directorio de trabajo y el staging area del proyecto.
 - No se pueden combinar las ramas porque esos cambios pendientes podrían ser sobrescritos por los commits que se van a combinar.
 - Cuando esto pasa, no hay conflictos con otros desarrolladores, si no conflictos con cambios locales pendientes.
 - Para resolverlo deberás usar: git stash, git checkout, git commit or git reset.
 - Este fallo produce mensaje de un error similar a:

```
error: Entry '<fileName>' not uptodate. Cannot merge.  
(Changes in working directory)
```

Resolver conflictos II

- Git falla durante del proceso de combinar ramas (merge)
 - En este caso el conflicto se produce entre la rama local y la rama remota a combinar
 - Indica un conflicto con el código remoto
 - Son los más habituales
 - Git tratará de combinar ambas ramas automáticamente pero necesitará que tú resuelvas los archivos conflictivos
 - Para resolverlo, deberás usar los comandos:
 - `git merge --abort`: saldrá del proceso de combinación y volverá a la rama al estado anterior, como si nada hubiera pasado
 - `git reset`: resetea los archivos conflictivos
 - Este fallo produce mensaje de un error similar a:

`error: Entry '<fileName>' would be overwritten by merge.`

Ejemplo de conflicto I

- Creamos un nuevo repositorio con una rama master y un archivo merge.txt

```
$ mkdir git-merge-test
$ cd git-merge-test
$ git init .
$ echo "this is some content to mess with" > merge.txt
$ git add merge.txt
$ git commit -m "we are committing the initial content"
[master (root-commit) d48e74c] we are committing the initial content
 1 file changed, 1 insertion(+)
create mode 100644 merge.txt
```

Ejemplo de conflicto II

- Creamos una nueva rama conflictiva: new_branch_to_merge_later
- Sobreescribimos el contenido del archivo merge.txt
- Confirmamos los cambios

```
$ git checkout -b new_branch_to_merge_later
$ echo "totally different content to merge later" > merge.txt
$ git add merge.txt
$ git commit -m"edited the content of merge.txt to cause a conflict"
[new_branch_to_merge_later 6282319] edited the content of merge.txt to cause a
conflict
1 file changed, 1 insertion(+), 1 deletion(-)
```

Ejemplo de conflicto III

- Cambiamos a la rama master, añadimos nuevo contenido al archivo txt y hacemos commit.

```
$ git checkout master
Switched to branch 'master'
$ echo "content to append" >> merge.txt
$ git add merge.txt
$ git commit -m"appended content to merge.txt"
[master 24fbe3c] appended content to merge.txt
1 file changed, 1 insertion(+)
```

- ¿Qué pasa si ejecutamos `$ git merge new_branch_to_merge_later` ?

-  **Tenemos un conflicto**

Auto-merging merge.txt

CONFLICT (content): Merge conflict in merge.txt

Automatic merge failed; fix conflicts and then commit the result.

Resolución de conflicto I

Git ofrece un mensaje descriptivo para avisar del conflicto ocurrido y cómo solucionarlo. Podemos obtener más información con el comando:

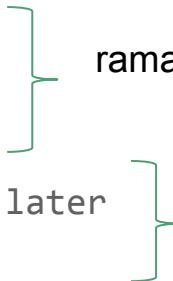
```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)
Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   merge.txt
```

El proceso de combinación no ha finalizado por el conflicto.

Resolución de conflicto II

El archivo conflictivo ha modificado su estado

```
$ cat merge.txt
<<<<< HEAD
this is some content to mess with
content to append
=====
totally different content to merge later
>>>>> new_branch_to_merge_later
```



rama actual \Rightarrow master

rama a combinar \Rightarrow new_branch_to_merge_later

Resolución de conflicto III

Para resolver el conflicto, simplemente modifica el archivo

```
$ cat merge.txt
this is some content to mess with
content to append
totally different content to merge later
```



rama actual ⇒ master
conflicto resuelto

Incluye estos cambios en el staging area:

```
$git add merge.txt
$git commit -m "merged and resolved the conflict in merge.txt"
```

¿Qué pasa si ejecutamos `$ git merge new_branch_to_merge_later` ?

- Ya no es necesario, ahora no tenemos conflicto y la rama está actualizada
Already up to date.
- Ahora podríamos enviarlo al repositorio remoto

Resolución de conflicto IV

Los conflictos también los puedes ver y resolver en IDEs como VisualStudio

Team Explorer - Resolve Conflicts Merge - vctmp2866...05.app.b6a7bbbd.ts* ×

Accept Merge | ← → | [] [] [] [] [] []

1 Conflicts (0 Remaining)

Source: FabrikamTypeScriptApp\app.ts;develop Target: FabrikamTypeScript Virtual Proj

```
1 class Greeter {
2     element: HTMLElement;
3     span: HTMLElement;
4     timerToken: number;
5
6     constructor(element: HTMLElement) {
7         this.element = element;
8         this.element.innerHTML += "The time is currently: ";
9         this.span = document.createElement('span');
10        this.element.appendChild(this.span);
11        this.span.innerText = new Date().toUTCString();
12    }
13
14    start() {
15        this.timerToken = setInterval(() => this.span.innerText =
16    }
17
18    stop() {
19        clearTimeout(this.timerToken);
}
```

Result: FabrikamTypeScriptApp\app.ts

TypeScript Virtual Projects Greeter

```
1 class Greeter {
2     element: HTMLElement;
3     span: HTMLElement;
4     timerToken: number;
5
6     constructor(element: HTMLElement) {
7         this.element = element;
8         this.element.innerHTML += "The time is currently: ";
9         this.span = document.createElement('span');
10        this.element.appendChild(this.span);
11        this.span.innerText = new Date().toUTCString();
12    }
}
```

Flujo de trabajo en Git I

Un flujo de trabajo git es un protocolo que se debe seguir para trabajar de forma consistente y productiva

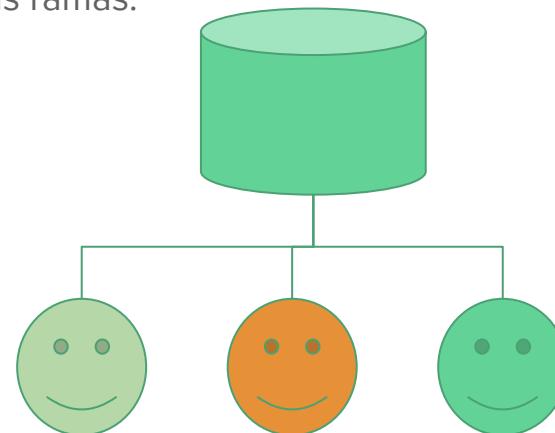
Git proporciona mucha flexibilidad para trabajar y no hay un proceso estandarizado al que atenerse. Es recomendable que cada organización diseñe el suyo.

Sin embargo, existen diversos flujos disponibles públicamente de los que partir como:

- El más simple: Centralized Workflow
- El más usado en Github: Gitflow Workflow

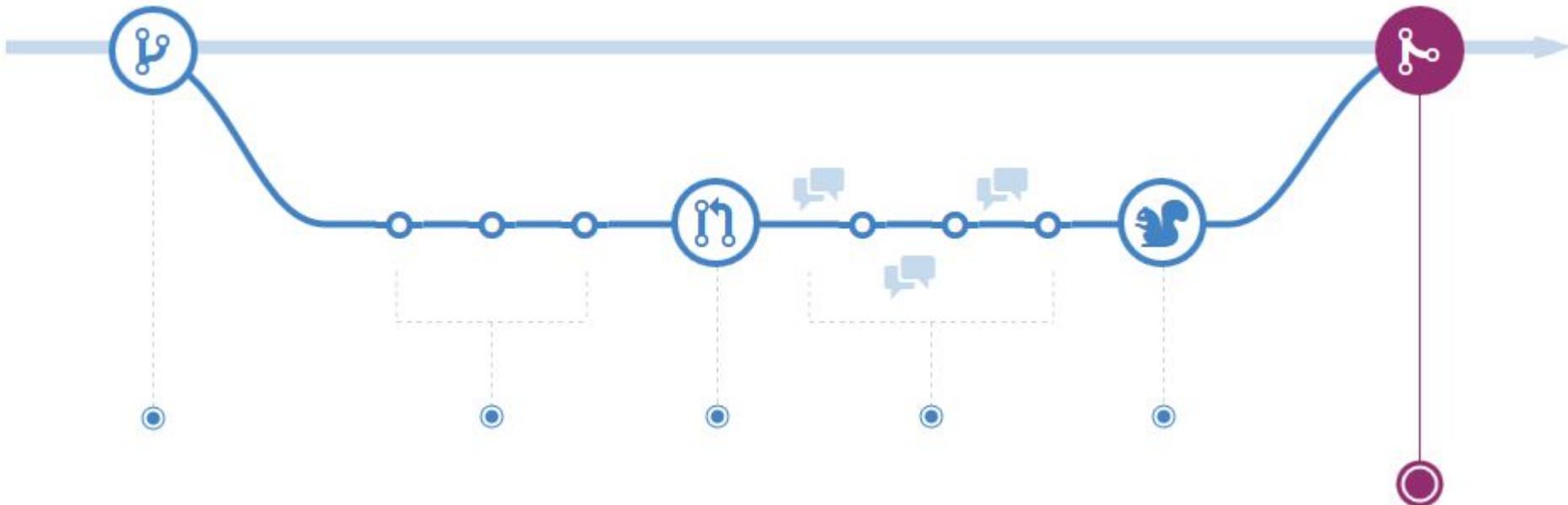
Flujo de trabajo en Git II

- Centralized Workflow:
 - Similar al protocolo usado en SCV centralizados.
 - Usa un repositorio central como punto de entrada único para todos los cambios del proyecto.
 - Todos los cambios se incluyen en la rama master.
 - No se necesitan otras ramas.



Flujo de trabajo en Git III

- Gitflow Workflow:



Flujo de trabajo en Git III

- Gitflow Workflow:

1. **Crea una rama**

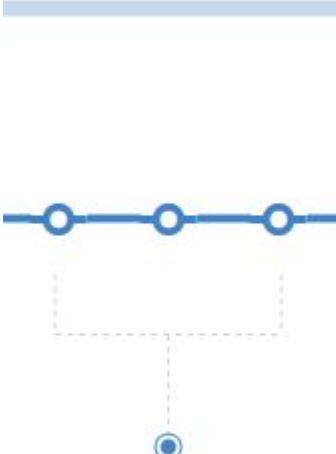
- Crear una rama para cada nueva idea a implementar.
- Los cambios de esta nueva rama no afectan a la rama master
- Puedes experimentar y confirmar cambios (commit), sabiendo que la nueva rama no se combinará hasta que esté lista



Flujo de trabajo en Git III

- Gitflow Workflow:

2. Añadir commits

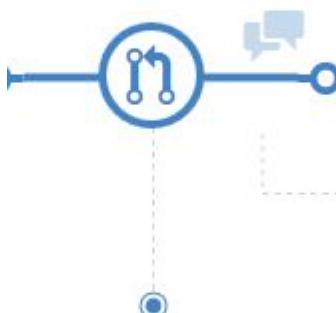
- 
- Cuando la rama está creada, puedes hacer cambios.
 - Cada vez que añadas, edites o elimines un archivo, debes hacer un commit.
 - Así registras tu progreso sobre una nueva funcionalidad en esta rama y para tus compañeros es transparente por qué haces cada cosa.
 - Cada commit tiene un mensaje que describe un único cambio para permitir volver atrás fácilmente, si por ejemplo encuentras un bug

Flujo de trabajo en Git III

- Gitflow Workflow:

3. Abrir una Pull Request

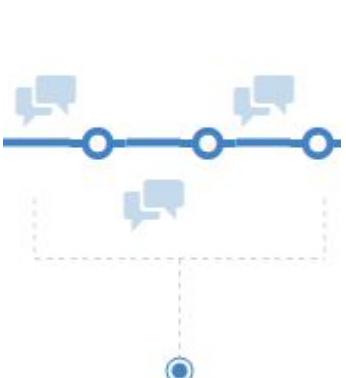
- Los Pull Requests iniciar una discusión sobre tus commits ya que todo el mundo puede ver tus cambios exactos, que serán aceptados si aceptan la petición de combinación (merge)
- Se puede abrir un Pull Request en cualquier momento del desarrollo para iniciar una conversación
- Son útiles para contribuir en proyectos de código abierto



Flujo de trabajo en Git III

- Gitflow Workflow:

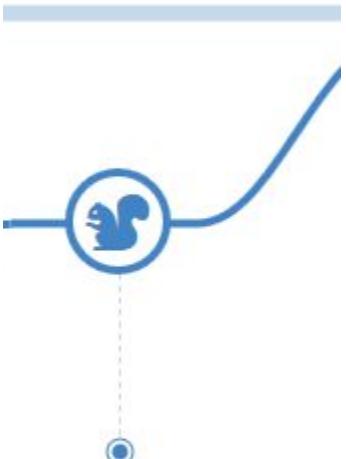
4. Discute y revisa tu código

- 
- El encargado de revisar tus cambios puede hacer preguntas o comentarios sobre tus cambios hasta que todo esté en orden (no faltan test, el código no tiene errores, etc.).
 - Puedes continuar subiendo cambios a tu rama según los comentarios sobre tus commits
 - GitHub muestra tus nuevos commits y cualquier comentario adicional.

Flujo de trabajo en Git III

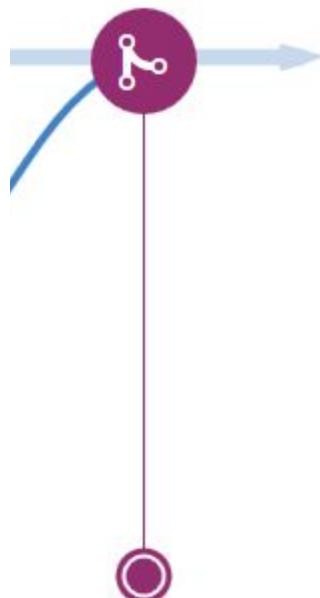
- Gitflow Workflow:

5. Despliegue

- 
- Github te permite desplegar tu rama en el sistema de producción antes de combina la nueva rama con master.
 - Si tu rama causa problemas en producción, puedes volver atrás y desplegar la rama máster en producción.

Flujo de trabajo en Git III

- Gitflow Workflow:



6. Combina (Merge)

- Cuando tus cambios funcionan en producción, es el momento de combinar la rama master con la nueva rama.
- Una vez combinadas, cada Pull Request preserva los cambios históricos de tu código de manera que aunque pase el tiempo, cualquiera puede consultarlos y entender por qué se tomaron las decisiones.

Puedes ampliar información sobre este proceso en:

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

<https://nvie.com/posts/a-successful-git-branching-model/>

<https://guides.github.com/introduction/flow/>

Flujo de trabajo en Git IV

- HADA Workflow:
 - Para la práctica en grupo:
 - la asignatura tiene su propio flujo de trabajo para que varios desarrolladores colaboren
 - diferente según si eres el responsable del grupo o no

Webs de interés

1. [git](#)
2. [git guide](#)
3. [Carl's Worth tutorial](#)
4. [git-for-computer-scientists](#)
5. [gitmagic](#)
6. [freedesktop](#)
7. [gitready](#)
8. [progit](#)
9. [winehq](#)
10. Presentación en [vídeo de git](#) hecha por *Linus Torvalds*
11. Vídeo de [uso de git/github desde Windows](#) con el intérprete de órdenes y con [Visual Studio Code](#) y Visual Studio 2015.

Hada T2: Programación Dirigida por Eventos

Un nuevo paradigma de programación.

Departamento de Lenguajes y Sistemas Informáticos Universidad de Alicante

Objetivos del tema

- Conocer el paradigma de la *Programación Dirigida por Eventos*.
- Aprender a programar bajo este nuevo paradigma.
- Saber y entender cómo se puede realizar en **C#**.
- Conocer, distinguir y saber emplear los conceptos de *evento* y *callback*.

Preliminares

- En términos de la estructura y la ejecución de una aplicación representa lo opuesto a lo que hemos hecho hasta ahora: ***programación secuencial***.
- La manera en la que escribimos el código y la forma en la que se ejecuta éste está determinada por los sucesos (**eventos**) que ocurren como consecuencia de la interacción con el mundo exterior.
- Podemos afirmar que representa un nuevo ***paradigma de programación***, en el que todo gira alrededor de los eventos o cambios significativos en el estado de un programa.

Programación secuencial vs. dirigida por eventos I

- En la ***programación secuencial*** le decimos al usuario lo que puede hacer a continuación, desde el principio al final del programa.
- El tipo de código que escribimos es como éste:

```
repetir
    presentar_menu ();
    opc = leer_opcion ();
    ...
    si (opc == 1) entonces accion1 ();
    si (opc == 2) entonces accion2 ();
    ...
hasta terminar
```

Programación secuencial vs. dirigida por eventos II

- En la ***programación dirigida por eventos*** indicamos:
 1. ¿Qué cosas -eventos- pueden ocurrir?
 2. Lo que hay que hacer cuando estos ocurran
- El tipo de código que escribimos es como éste:

```
son_eventos (ev1, ev2, ev3...);  
...  
cuando_ocurra ( ev1, accion1 );  
cuando_ocurra ( ev2, accion2 );  
repetir  
...  
hasta terminar
```

Programación secuencial vs. dirigida por eventos III

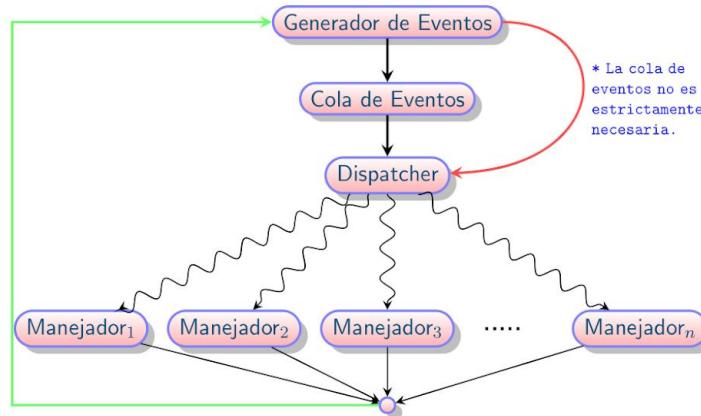
- A partir de este punto **los eventos pueden ocurrir en cualquier momento** y marcan la ejecución del programa.
- Aunque no lo parezca plantean un problema serio: **el flujo de la ejecución del programa escapa al programador.**
- El usuario (como fuente generadora de eventos) toma el control sobre la aplicación.
- Esto implica tener que llevar cuidado con el diseño de la aplicación teniendo en cuenta que **el orden de ejecución** del código no lo marca el programador y, además, **puede ser distinto cada vez.**

Esqueleto de una aplicación dirigida por eventos I

- Al principio de la misma llevamos a cabo una iniciación de todo el sistema de eventos.
- **Se definen todos los eventos que pueden ocurrir.**
- Se prepara el generador o generadores de estos eventos.
- **Se indica qué código se ejecutará en respuesta a un evento producido**
-ejecución diferida de código-.
- Se espera a que se vayan produciendo los eventos.
- **Una vez producidos son detectados por el “dispatcher” o planificador de eventos**, el cual se encarga de invocar el código que previamente hemos dicho que debía ejecutarse.

Esqueleto de una aplicación dirigida por eventos II

- Todo esto se realiza de forma ininterrumpida hasta que finaliza la aplicación.
- A esta ejecución ininterrumpida es a lo que se conoce como el **bucle de espera de eventos**.
- Las aplicaciones con un interfaz gráfico de usuario siguen este esquema de programación que acabamos de comentar. Gráficamente sería algo así:



Analogía

- Voy a un restaurante de comida rápida vs saco dinero de un cajero

La diferencia clave entre estos escenarios es que en el primero el siguiente cliente puede ser servido mientras espero mi comida (me avisarán). En el segundo, quien quiera usar el cajero deberá esperar a que yo acabe (secuencial).

El principio de HollyWood I

- Las pautas de uso de la *programación dirigida por eventos* se pueden resumir con el llamado *principio de Hollywood*.
- Este es muy sencillo de entender: **No nos llame...ya le llamaremos**
- Se emplea sobre todo cuando se trabaja con *frameworks*.
- El flujo de trabajo se parece a esto:
 1. En el caso de trabajar con un *framework* implementamos un interfaz y en el caso más sencillo escribimos el código a ejecutar más adelante.
 2. Nos registramos...es decir, indicamos de algún modo cuál es el código a ejecutar posteriormente.
 3. Esperamos a que se llame -al código registrado previamente- cuando le corresponda, recuerda: **No nos llame...ya le llamaremos**.

El principio de HollyWood II

- El programador ya no *dicta* el flujo de control de la aplicación, sino que son los eventos producidos los que lo hacen.
- Puedes consultar más información sobre él [aquí](#).
- Si quieres ampliar más tus conocimientos sobre él debes saber que a este principio también se le conoce por otros nombres:
 1. Inversión de control: [IoC](#).
 2. Inyección de dependencias [DI](#).
- Puedes ampliar más información sobre estas técnicas de programación en asignaturas como Programación-3.

¿Necesitamos un lenguaje de programación especial? I

- No, La ejecución diferida de código tiene sus orígenes en el concepto de **Callback**.
- En Lenguaje **C** un *callback* no es más que un puntero a una función.
- En la propia biblioteca estándar de **C** hay varios ejemplos de ello.
 - Hagamos un pequeño ejercicio: en la biblioteca estándar de **C** (`#include <stdlib.h>`)

disponemos de la función “*qsort - ordena un vector*”. El prototipo de la misma es:

```
void qsort(void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *));
```

- Identifica cada uno de sus parámetros. Trata de entender qué es el parámetro “*compar*”. **Propón un posible valor para ese parámetro.**

Parámetros

- **base** -- Es el puntero al primer elemento del array a ordenar.
- **nmemb** -- Este es el número de elementos en el array al que apunta *base*.
- **size** -- Es el tamaño en bytes de cada elemento del array.
- **compar** -- Esta es la función que compara dos elementos.
- **callback** es un trozo de código ejecutable que se le pasa como argumento a otro código, del cual se espera que llame *call back (execute)* a la función pasada como argumento cuando sea conveniente.

¿Necesitamos un lenguaje de programación especial? II

- Cuando tengas hecho todo este estudio trata de crear un programa ejecutable mínimo que te sirva para comprobar cómo funciona qsort.

```
int int_cmp (const void* pe1, const void* pe2) {  
    int e1 = *((const int*) pe1);  
    int e2 = *((const int*) pe2);  
    return e1-e2;  
}  
  
int main () {  
    int v[4] = {4,0,-1,7};  
  
    for (int i = 0; i < 4; i++)  
        printf("v[%d]=%d\n", i, v[i]);  
  
    qsort (v, 4, sizeof(int), int_cmp);  
    printf("\n");  
    for (int i = 0; i < 4; i++)  
        printf("v[%d]=%d\n", i, v[i]);  
    return 0;  
}
```

Ejecución diferida
Ventaja: Podría llamar a
cualquier método con la misma
signatura

Delegados en C# I

- Se pueden usar delegados p.e. para pasar métodos como argumentos de otros métodos (repasa el ejemplo de *qsort* en **C** visto antes).
- En lugar de simplemente pasar el método como parámetro, primero declaramos una variable del tipo **delegado**.
- Un delegado es un **tipo de dato** que representa una referencia a un método con una serie de parámetros y un tipo de resultado.

Delegados en C# II

- Podemos declarar variables de este tipo y asignarles un valor, el cual es un método cuya signatura se corresponde con la del delegado -se permite el uso de varianza en el tipo del resultado-.
- Posteriormente podemos invocar estos métodos a través del *delegado*.
- A un delegado le podemos asignar métodos de clase (static) y de instancia.
 - *Un delegado también puede contener una referencia a un objeto, así que podemos apuntar a una función/método en una instancia concreta de la clase. Un puntero a función no permite esto.*

Delegados en C#: Declaración

- La sintaxis para declarar un delegado es la siguiente:

```
public delegate int PerformCalculation(int x, int y);
```

- Necesitamos el espacio de nombres System.Delegate

Delegados en C# ejemplo

- Un ejemplo de uso:

```
delegate Persona menor (Persona p1, Persona p2);
class Persona {
    ...
    //métodos que serán parámetros de una función
    public static Persona nombreMenor (Persona p1, Persona p2){...}
    public static Persona edadMenor (Persona p1, Persona p2) {...}

    public static void muestraPrecede (Persona p1, Persona p2,
                                     menor cf) {
        Person p = cf(p1, p2); ←
        if (p != null) {
            Console.WriteLine ("{0} with age {1}.", p.name, p.age);
        }
    }
    // DATOS // PROPIEDADES C#
    public string nombre {get; set;}
    public int edad {get; set;}
}
```

1.declaración

2. invocación

```
public static void Main () {
    Persona juan = new Persona ("Juan", 20);
    Persona andres = new Persona ("Andres", 30);
    Persona.muestraPrecede (juan, andres,
                           Persona.edadMenor );
    Persona.muestraPrecede (juan, andres,
                           Persona.nombreMenor );
}
```

3. instanciaión

Propiedades C#

- Una propiedad es un miembro que proporciona un mecanismo flexible para **leer, escribir o calcular el valor de un campo privado.**
- Las propiedades se pueden usar como si fueran miembros de datos públicos, pero en realidad son métodos especiales denominados *descriptores de acceso*.
- Esto permite acceder fácilmente a los datos a la vez que proporciona la seguridad y la flexibilidad de los métodos.

Propiedades con campos de respaldo

- Para devolver el valor de la propiedad se usa un descriptor de acceso de propiedad get, mientras que para asignar un nuevo valor se emplea un descriptor de acceso de propiedad set.
- Estos descriptores de acceso pueden tener diferentes niveles de acceso.
- La palabra clave value se usa para definir el valor que va a asignar el descriptor de acceso set.

```
using System;
class TimePeriod
{
    private double _seconds;
    public double Hours
    {
        get { return _seconds / 3600; }
        set {
            if (value < 0 || value > 24)
                throw new ArgumentOutOfRangeException(
                    $"{nameof(value)} must be between 0 and 24.");
            _seconds = value * 3600;
        }
    }
}
```

```
class Program
{
    static void Main()
    {
        TimePeriod t = new TimePeriod();
        // The property assignment causes the 'set' accessor to be
        // called.
        t.Hours = 24;

        // Retrieving the property causes the 'get' accessor to be
        // called.
        Console.WriteLine($"Time in hours: {t.Hours}");
    }
}
// The example displays the following output:
//      Time in hours: 24
```

Propiedades implementadas automáticamente

- En algunos casos, los **descriptores de acceso de propiedad get y set** simplemente asignan un valor a un campo de respaldo o recuperan un valor de él sin incluir ninguna lógica adicional. Mediante las propiedades **implementadas automáticamente**, puede simplificar el código y conseguir que el compilador de C# le proporcione el campo de respaldo de forma transparente.
- Una propiedad implementada automáticamente se define mediante las palabras clave `get` y `set` sin proporcionar ninguna implementación.

```
using System;

public class SaleItem
{
    public string Name
    { get; set; }

    public decimal Price
    { get; set; }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        var item = new SaleItem{ Name = "Shoes", Price = 19.95m };
        Console.WriteLine($"{item.Name}: sells for
{item.Price:C2}")
    }
    // The example displays the following output:
    //      Shoes: sells for $19.95
}
```

Funciones lambda en C#

- C# permite crear funciones anónimas (**sin nombre**) también llamadas funciones lambda.
- **Se definen en el momento que se van a usar.**
- Podemos hacerlo de diferentes maneras:

```
// Create a delegate.  
delegate void Tdel(int x);  
  
// Instantiate the delegate using an anonymous method.  
Tdel d = delegate(int k) { /* ... */ };
```

Especificamos los parámetros de entrada (si hay) a la izquierda del operador lambda => y situamos la expresión o bloques de instrucciones al otro lado.

```
delegate int Tdel(int i);  
static void Main(string[] args) {  
    Tdel myDelegate = x => x * x;  
    int j = myDelegate(5); //j = 25  
}  
  
// Num. of param. ≠ 1 → use of parenthesis  
1. (x, y) => x == y  
2. (int x, string s) => s.Length > x  
3. () => SomeMethod()
```

cero parámetros de entrada

A veces, es difícil o imposible para el compilador deducir los tipos de datos. Cuando esto pasa, se pueden especificar explícitamente.

Marco de Prog. dirigida por eventos en C# y .Net

- El modelo empleado en C# para hacer uso de esta técnica es el siguiente:
 - Introduce un elemento nuevo, el evento (**event**).
 - **Los eventos representan los cambios de estado de un objeto y a ellos les podemos “conectar” el código que queramos que se ejecute cuando se generen (handlers).**

Un evento es un mecanismo de comunicación entre objetos

Marco de Prog. dirigida por eventos en C# y .Net

- Cada vez que se produzca un **evento** para un objeto concreto de una clase, **se ejecutarán todos los *handlers* que le hayamos asociado**, secuencialmente uno tras otro y no tiene porqué ser en el orden en que se conectaron.

Ejemplo simple

```
public class VideoEncoder  
{  
    public void EncodeVideo(Video video)  
    {  
        //encode logic  
        _mailService.send(new Mail())  
    }  
}
```

si queremos añadir un nuevo método pq queremos enviar ahora un SMS, hay que recompilar todo

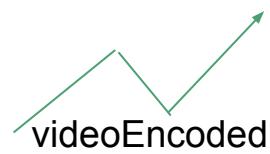
podemos usar eventos para solucionarlo

Emisor - Receptor

EMISOR DEL EVENTO

Publisher

videoEncoder class



RECEPTOR DEL EVENTO

Subscriber

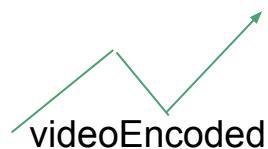
mailService class

Emisor - Receptor

EMISOR DEL EVENTO

Publisher

videoEncoder class



RECEPTOR DEL EVENTO

Subscriber

mailService class

RECEPTOR DEL EVENTO

Subscriber

MessageService class

Ejemplo simple

```
public class VideoEncoder  
{  
    public void EncodeVideo(Video video)  
    {  
        //encode logic  
        OnVideoEncoded();  
    }  
}
```

añadimos un nuevo método que deberá **NOTIFICAR A LOS SUSCRIPTORES**

enviándoles un mensaje

Esto en C#: *invocando a un método de los suscriptores*

Pero cómo sabe OnVideoEncoded a qué método llamar?

- Necesitamos un **acuerdo o contrato** entre los *publishers* (emisor) y los *subscribers* (receptor)
- ***un método con una firma específica***
- ***Event handler***

```
public void OnVideoEncoded (object sender, EventArgs a)
```

Emisor - Receptor

EMISOR DEL EVENTO

Publisher

videoEncoder class



RECEPTOR DEL EVENTO

Subscriber

mailService class

RECEPTOR DEL EVENTO

Subscriber

MessageService class

- *Event handler*

```
public void OnVideoEncoded  
(object sender, EventArgs a)
```

Delegados

- Con un delegado le indicamos al Publisher a qué método puede llamar cuando se invoca el evento.
- Acuerdo/contrato entre el publisher (emisor) y el subscriber (receptor).
- **Determina la firma del método manejador** (event handler) en el subscriber.

Pasos

- Definir el delegado

```
public delegate void VideoEncodedEventHandler(object sender, EventArgs args);
```

- Definir el evento basándonos en el delegado

```
public event VideoEncodedEventHandler videoEncoded;
```

- Lanzar el evento

```
videoEncoded(this, EventArgs.Empty);
```

Ejemplo simple

```
public class VideoEncoder{  
  
    public delegate void VideoEncodedEventHandler(object sender, EventArgs args);  
  
    public event VideoEncodedEventHandler videoEncoded;  
  
    public void EncodeVideo(Video video) { //encode logic  
  
        OnVideoEncoded();  
  
    }  
  
    protected virtual void OnVideoEncoded() //notify subscribers  
  
    {if videoEncoded!= null //if we have any subscriber  
  
        videoEncoded(this, EventArgs.Empty);  
  
    }
```

Vamos a crear subscribers (handlers)

```
public class MailService

{

    public void OnVideoEncoded(object source,
EventArgs e)

    {

        Console.WriteLine("MailService: enviando email");

    }

}
```

```
class Program{

    static void Main(string[] args){

        Video video= new Video();

        VideoEncoder v=new VideoEncoder(); //publisher

        MailService ms=new MailService(); //subscriber

        v.videoEncoded+=ms.OnVideoEncoded;

        //pointer to that method, we do the subscription

        v.Encode(video);

    }

}
```

Vamos a crear subscribers (handlers)

```
public class MessageService

{

    public void OnVideoEncoded(object source,
EventArgs e)

    {

        Console.WriteLine("MessageService: enviando
sms");

    }

}
```

```
class Program{

    static void Main(string[] args){

        Video video= new Video();

        VideoEncoder v=new VideoEncoder(); //publisher

        MailService ms=new MailService(); //subscriber

        MessageService mg = new MessageService();

        v.videoEncoded+=ms.OnVideoEncoded;

        v.videoEncoded+=mg.OnVideoEncoded;

        //pointer to that method, we do the subscription

        v.Encode(video); }
```

Parámetros en el evento

```
public class VideoEventArgs: EventArgs
```

```
{
```

```
    public Video v { get; set;}
```

```
}
```

Ejemplo simple

```
public class VideoEncoder{  
  
    public delegate void VideoEncodedEventHandler(object sender, VideoEventArgs args);  
  
    public event VideoEncodedEventHandler videoEncoded;  
  
    public void EncodeVideo(Video video) { //encode logic  
  
        OnVideoEncoded(video);  
  
    }  
  
    protected virtual void OnVideoEncoded(Video video) //notify subscribers  
  
    {if videoEncoded!= null //if we have any subscriber  
  
        videoEncoded(this, new VideoEventArgs(){Video=video;});  
  
    }  
}
```

Modificamos los subscribers (handlers)

```
public class MessageService
```

```
{
```

```
    public void OnVideoEncoded(object source,  
        VideoEventArgs e)
```

```
{
```

```
    Console.WriteLine("MessageService: enviando  
    sms");
```

```
}
```

```
public class MailService
```

```
{
```

```
    public void OnVideoEncoded(object source,  
        VideoEventArgs e)
```

```
{
```

```
    Console.WriteLine("MailService: enviando email");
```

```
}
```

Otro ejemplo de eventos en C# I

- Vamos a crear una clase que representa una **persona** y otra que representa un **vehículo**.
- En la clase vehículo se crearán un número concreto de personas que viajarán en él.
- Los objetos de la clase persona generarán el evento **cinturonQuitado** cada vez que uno de los pasajeros del vehículo se quite el cinturón de seguridad.

Resumen clases del ejemplo

- Program.cs → método main
 - Llama al constructor de vehiculo que genera la lista de personas y aleatoriamente “quita el cinturon” de una persona poniendo a false la propiedad cinturonQuitado.
- Vehiculo
 - “construye” la lista de personas. Conecta el manejador al evento y lo invoca. El manejador también se define aquí. **(Nunca se invoca directamente al manejador, se llamará cuando se lance el evento).**
- Persona
 - Aquí se define el evento.
- cinturonQuitadoArgs : EventArgs
 - *Clase especial EventArgs para guardar info sobre cambios en el cinturón.*

Ejemplo de eventos en C# II

```
public class Persona {  
    public Persona (string nombre, bool cinturon)  
    {  
        this.nombre = nombre;  
        this.cinturon = cinturon;  
    }  
  
    public string nombre { get; set; }  
  
    private bool _cinturon;  
    public bool cinturon  
    {  
        get { return _cinturon; }  
        set {  
            _cinturon = value;  
  
            if (!_cinturon && cinturonQuitado!=null) {  
                //Lanzar evento velocidad  
                cinturonQuitado(this, new CinturonQuitadoArgs(value));  
            }  
        }  
    }  
}
```

```
↓  
  
    public void quitarCinturon ()  
    {  
        this.cinturon = false;  
    }  
//definición evento  
public event EventHandler  
<CinturonQuitadoArgs> cinturonQuitado;  
  
// Fin de la clase Persona  
}  
↓
```

declaración evento

Declaración evento en C#

```
public event EventHandler <CinturonQuitadoArgs> cinturonQuitado;
```

- visibilidad + palabra clave *event*
- Delegado **EventHandler** + **EventArgs** (o tipo derivado)
- nombre del evento

Qué representa **EventHandler** ???

La **signatura de métodos/funciones** que pueden conectarse con ese evento + **parámetros del evento**

Delegados y eventos

- No necesitamos declarar un delegado, podemos utilizar el delegado EventHandler definido en la biblioteca de clases de C#

```
delegate void EventHandler(object sender, EventArgs e);
```

Delegados y eventos II

- Por tanto... el prototipo de los métodos manejadores de eventos deben coincidir con el prototipo del delegado EventHandler.
- Por tanto, por defecto los manejadores de evento devuelven **void** y tienen dos parámetros:
 - **Objeto que lanza el evento**
 - **Argumentos del evento:** información específica del evento (*EventArgs o tipo derivado*)

Ejemplo de eventos en C# III

```
public class CinturonQuitadoArgs : EventArgs
{
    public bool cinturon { get; set; }
    public CinturonQuitadoArgs (bool cinturon)
    {
        this.cinturon = cinturon;
    }
}
```

Ejemplo de eventos en C# IV

```
public class Vehiculo
{
    List<Persona> personas { get; set; }

    public Vehiculo (int numeroPersonas)
    {
        personas = new List<Persona>();
        for (int i = 0; i < numeroPersonas; i++) //se crea un cjto de personas
        {
            Persona p = new Persona("Persona" + i.ToString(), true);
            p.cinturonQuitado += cuandoCinturonQuitado; // Conexión del callback al evento!
            personas.Add(p);
        }
    }
}
```

conexión del evento y
manejador



Ejemplo de eventos en C# V

```
public void quitarCinturonAleatorio ()  
{  
    Random rand = new Random();  
    int posicion = rand.Next(0, personas.Count - 1);  
    personas[posicion].quitarCinturon(); // Forzamos a que se emita el evento!  
}  
  
private void cuandoCinturonQuitado (object sender, CinturonQuitadoArgs args)  
{  
    Persona p = (Hada.Persona) sender;  
    Console.WriteLine("¡¡Alguien se ha quitado el cinturón!!");  
    Console.WriteLine("Persona: " + p.nombre);  
    Console.WriteLine("Cinturon: " + args.cinturon);  
}  
  
// Fin de la clase Vehiculo  
}
```

invoca evento (llamada a quitarCinturon)

handler

Ejemplo de eventos en C# VI

```
// Main para probar el código de eventos

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Main");
        Vehiculo v = new Vehiculo(4);
        v.quitarCinturonAleatorio(); //Lanza el evento
    }
}
```

Ejemplo de eventos en C# VII

```
public class Persona {  
    public Persona (string nombre, bool cinturon)  
    {  
        this.nombre = nombre;  
        this.cinturon = cinturon;  
    }  
  
    public string nombre { get; set; }  
  
    private bool _cinturon;  
    public bool cinturon  
    {  
        get { return _cinturon; }  
        set {  
            _cinturon = value;  
  
            if (!cinturon) {  
                //Lanzar evento velocidad  
                cinturonQuitado(this, new CinturonQuitadoArgs(value));  
            }  
        }  
    }  
}
```

invoca evento (llamada a quitarCinturon)

```
↓  
  
    public void quitarCinturon ()  
    {  
        this.cinturon = false;  
    }  
//definición evento  
public event EventHandler  
<CinturonQuitadoArgs> cinturonQuitado;  
  
    // Fin de la clase Persona  
}
```

↓

Ejemplo de eventos en C# VIII

Recapitulando:

- **Definición evento:** Para cada evento que pueda generar una clase debemos definir una variable en la clase de tipo **EventHandler<T>**:

```
public event EventHandler<...> cinturonQuitado;
```

- **Parámetros del evento:** El tipo genérico **T** se instanciará al de la clase **EventArgs** o derivada de ella:

```
public event EventHandler<CinturonQuitadoArgs> cinturonQuitado;
```

Ejemplo de eventos en C# IX

Recapitulando:

- **Definición y asociación manejadores:**

A esta variable de tipo **evento** le conectaremos todos aquellos métodos que necesitemos, p.e.:

```
private void cuandoCinturonQuitado(object sender, CinturonQuitadoArgs e)...
p.cinturonQuitado += cuandoCinturonQuitado
```

- Lo invocaremos así: cinturonQuitado(this, new CinturonQuitadoArgs(value));

Hada T3: Introducción a las aplicaciones web

Aplicaciones web.

Objetivos

- Conocer la **arquitectura cliente/servidor**
- Comprender los conceptos de programación en el cliente y en el servidor
- Ventajas e inconvenientes de la arquitectura cliente/servidor
- Conocer las diferentes **tecnologías web**
- Conocer la arquitectura de n-capas
- Conocer los **servidores web** más utilizados
- Aprender una **metodología** en el desarrollo de las páginas web

Arquitectura cliente / servidor

- **Arquitectura cliente/servidor**
 - Arquitectura de n-capas
 - Aplicaciones web
 - Servidores web
 - Metodología de diseño
 - Ejercicios
-

Arquitectura cliente/servidor

- La arquitectura **cliente/servidor** es un modelo de diseño software en que los **servidores** comparten recursos o servicios, y los **clientes** solicitan dichos recursos
- Es una arquitectura típica de desarrollo de aplicaciones
- El **cliente** solicita y el **servidor** responde
- Es más ventajoso para **aplicaciones distribuidas** a través de una red de ordenadores
- La **capacidad de proceso** está dividida entre **clientes y servidores.**
- Facilita el diseño del sistema

Arquitectura cliente/servidor

- Diferentes requisitos para el **cliente** y el **servidor** (velocidad de procesamiento, memoria, capacidad de almacenamiento de datos, etc.)
- Un **servidor** puede dar servicio a miles de **clientes**
- **Cliente**
 - El software de la parte **cliente** se conoce como **front-end**
 - Inicia la petición
 - Espera y recibe la respuesta del **servidor**
 - Se puede conectar a un grupo de **servidores** de manera simultánea
 - Suele ofrecer un interfaz gráfico al usuario final

Arquitectura cliente/servidor

- **Servidor**

- El software de la parte del **servidor** se conoce como **back-end**
- Pasivo
- Espera las peticiones de los **clientes**
- Cuando recibe una petición, la procesa y devuelve la respuesta
- Normalmente acepta un gran número de conexiones simultáneas de los **clientes**

Arquitectura cliente/servidor

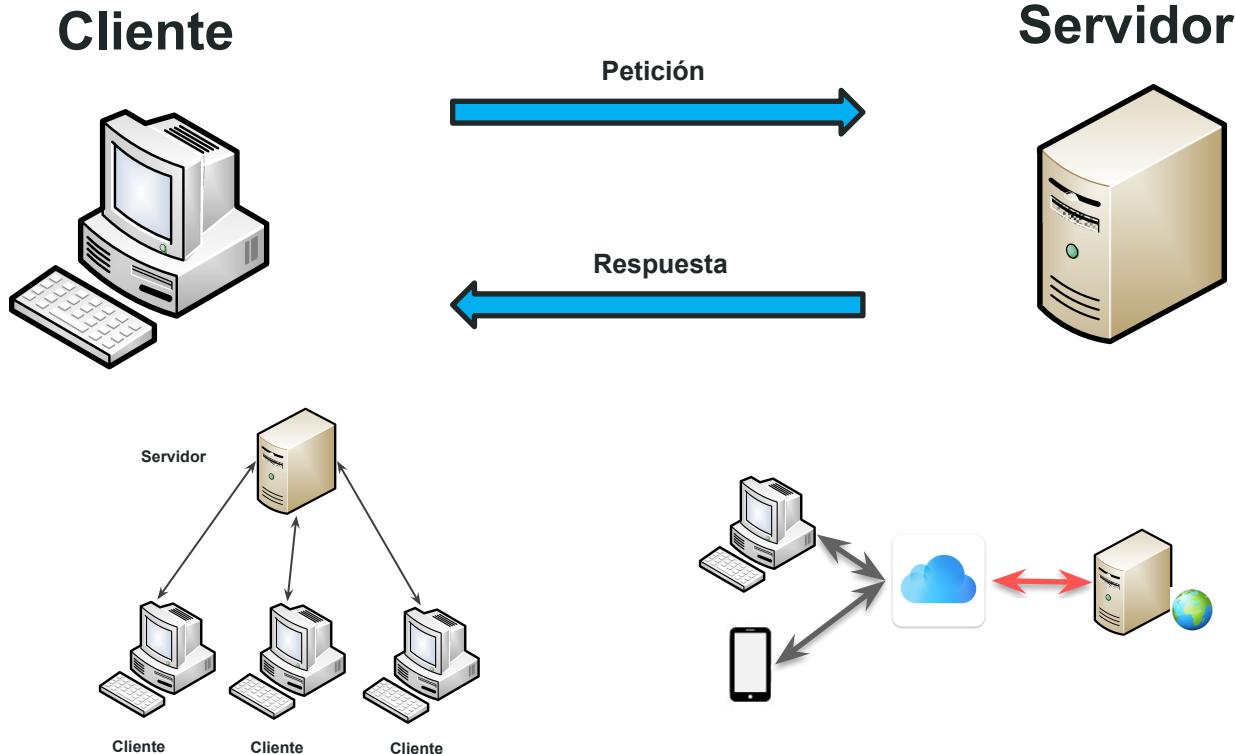
Ventajas

- + **Distribución de aplicaciones:** concurrencia de procesos
- + **Estandarización.** Los recursos están publicados y otras aplicaciones pueden acceder a ellos
- + **Portabilidad**
- + **Escalabilidad**
- + Posibilidad de utilizar clientes ligeros con requisitos mínimos de instalación
- + Entornos heterogéneos. El hardware o el sistema operativo puede ser diferente

Desventajas

- Aumenta la comunicación. Congestión tráfico de red
- **Falta de robustez.** Caídas del servidor

Arquitectura cliente/servidor



Arquitectura de n-capas

- Arquitectura cliente/servidor
 - **Arquitectura de n-capas**
 - Aplicaciones web
 - Servidores web
 - Metodología de diseño
 - Ejercicios
-

Arquitectura de n-capas

- **División de los componentes** de una aplicación en n niveles o capas lógicas.
- Pueden ser de 1 (aplicación individual), 2 o 3 o n capas.
- No implica separación física en distintos ordenadores de la red (una aplicación de 3 capas puede existir en un único ordenador).
- **Ventajas**
 - Permitir modificar una capa sin tener que modificar toda la aplicación.
 - Simplificación de la administración de los sistemas
 - Disponibilidad inmediata de cambios
 - Posibilidad de balanceo de carga de trabajo entre ordenadores

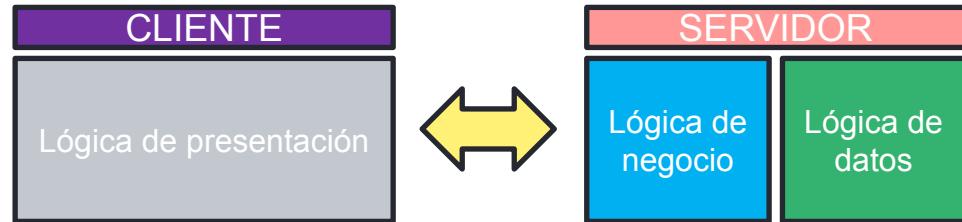
Arquitectura de n-capas

- Una arquitectura de 3 capas está dividida en:
 - **Interfaz de usuario (o lógica de presentación)**, componentes que interactúan con el usuario final. Interfaz gráfico (GUI) o basado en texto.
 - **Lógica de negocio**, contienen las reglas de negocio (funcionalidades) de nuestra aplicación. Corresponde con el núcleo de la aplicación.
 - **Persistencia (o lógica) de datos**, permite el acceso y almacenamiento de los datos

Modelos de distribución por capas

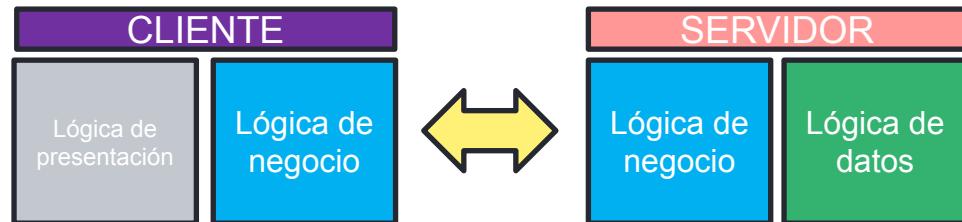
- Presentación distribuida

(una interfaz con validación de datos)



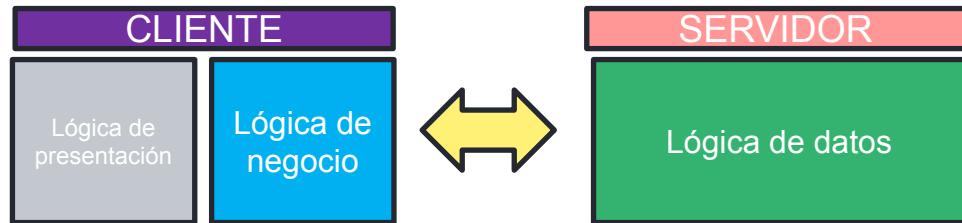
- Aplicación distribuida

(máxima flexibilidad)



- Datos distribuidos

(máxima carga en cliente y mayor ancho de banda)

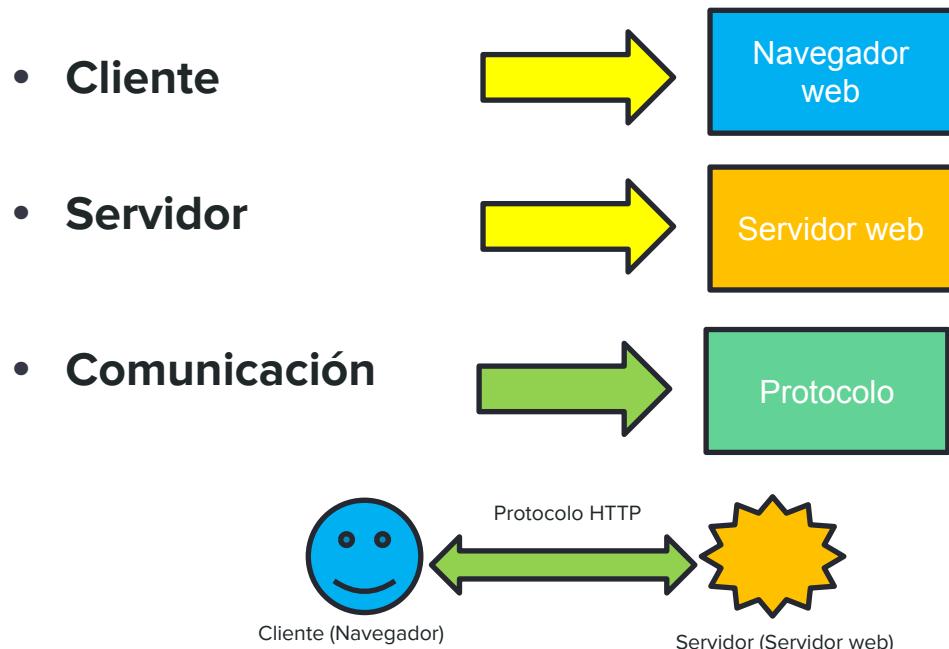


Aplicaciones web

- Arquitectura cliente/servidor
 - Arquitectura de n-capas
 - **Aplicaciones web**
 - Servidores web
 - Metodología de diseño
 - Ejercicios
-

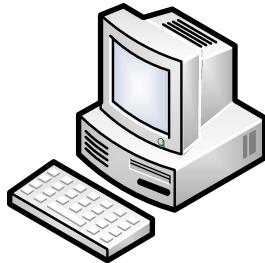
Introducción

- Una aplicación web es una aplicación informática que se ejecuta en el entorno web
- Basada en la arquitectura **cliente/servidor**.

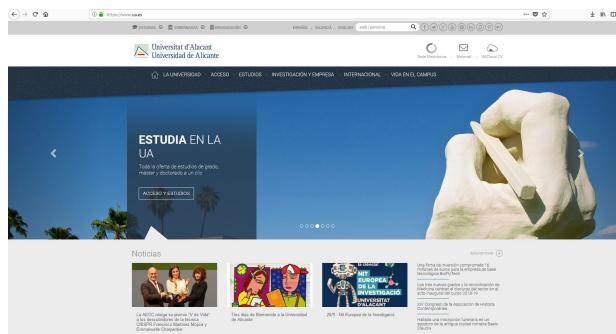
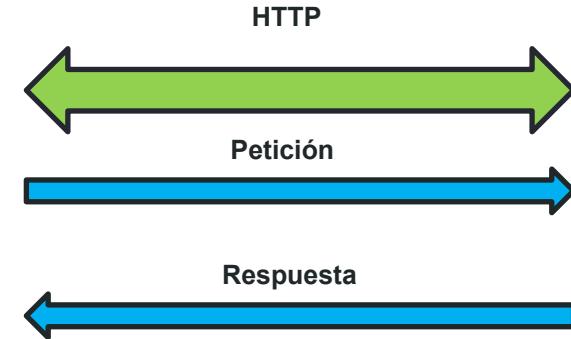


Introducción

Cliente



Navegador web



Servidor



Servidor web

Otros servicios

Base de datos

Aplicaciones

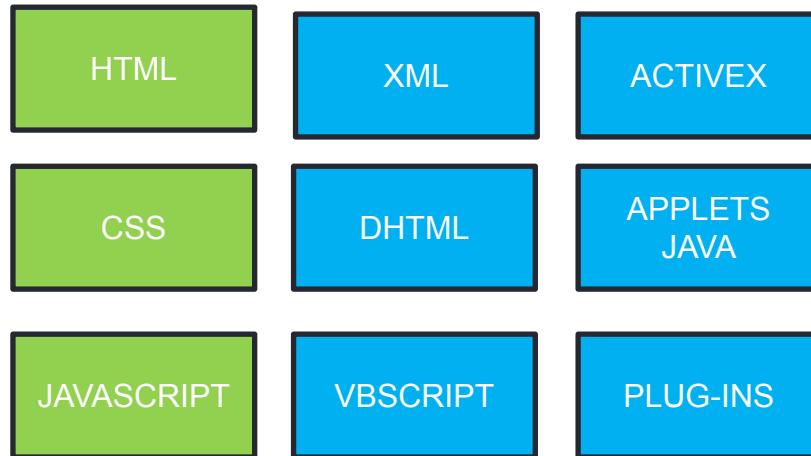
Ficheros

Impresoras

CORREO

Cliente

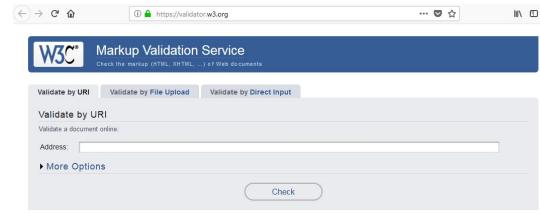
- Se encarga de gestionar las peticiones del usuario y la recepción del contenido de las páginas que provienen del servidor web
- Es capaz de interpretar los documentos HTML y otros recursos
- Tecnologías:



Tecnologías en cliente



- **HyperText Markup Language** (lenguaje de marcas de hipertexto)
- Lenguaje de programación para definir el contenido de una página web
- Versión actual: HTML 5.2
- **Define el formato del texto, posición, colores, tamaños**
- **Enlaces**: Permite añadir elementos externos a partir de referencias a la ubicación de dicho elementos (no se incrusta directamente en el código)
- Algunas **etiquetas**: <html>, <script>, <head>, <div>, , , , etc.
- Editores: Notepad, Atom, jedit, Visual Studio Code, Visual Studio, etc.
- W3C validator: <http://validator.w3.org>
 - Revisa el código fuente
 - Busca errores de sintaxis
 - Devuelve el resultado de la prueba de validación



This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.



Tecnologías en cliente

css

- **Cascading Style Sheet** (hojas de estilo en cascada)
- Es un lenguaje de diseño gráfico que **define la presentación** de un documento estructurado en un lenguaje de marcado como HTML
- Versión actual: CSS3
- **El estilo define la manera de mostrar los elementos HTML**
- El estilo se puede incluir en la misma página o en un fichero externo
- Permite establecer una separación entre la presentación (estilo) y el contenido de una página web
- **Ventajas:**
 - Posibilidad de mayor control en la presentación
 - Diferentes presentaciones para un mismo contenido
 - Menor carga de las páginas web
 - Programación independiente

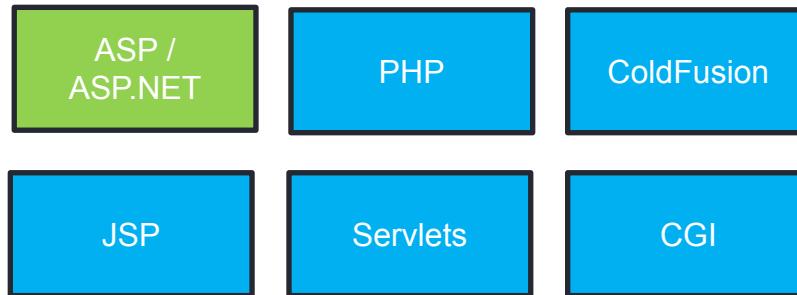
Tecnologías en cliente

JAVASCRIPT

- **Javascript** es un lenguaje de programación que está embebido dentro de una página web, para crear contenido e interactividad dinámica
- Lenguaje más estándar para **script** (más compatible)
- Durante muchos años fue el único método para incorporar dinamismo a las páginas web. En la actualidad hay otras tecnologías
- Sintaxis parecida a JAVA y C++
- Case sensitive (sensible a minúsculas y a mayúsculas)
- Incorpora el *Document Object model (DOM)*, que se utiliza para interactuar con la página web
- **Ejemplos de uso:** validar los datos de entrada de un usuario (comprobar datos correctos), proporcionar dinamismo en respuesta a las acciones del usuario (desplegar un menú al pasar el ratón), etc.

Servidor

- Software que espera las peticiones de los clientes
- En la aplicación del servidor habrá:
 - Conjunto de páginas estáticas (HTML)
 - Recursos multimedia (imágenes, documentos, videos , etc.)
 - Programas que se ejecutan y proporcionan páginas web dinámicas
- Tecnologías:



Tecnologías en servidor

PHP

- **Hipertext Preprocessor** es un lenguaje de Script, de **programación en el servidor** para desarrollo web **de contenido dinámico**
- Se suele combinar con el sistema gestor de base de datos MySQL
- El código de PHP se puede mezclar con el código HTML de la página web
- El código escrito en PHP es invisible al cliente, ya que es el servidor quien se encarga de ejecutar el código, y enviar el resultado HTML al cliente (navegador web)
- Es libre (licencia GPL), multiplataforma y tiene gran velocidad de respuesta
- Funciona con distintos servidores web: Apache, Nginx
- Manejo de excepciones (desde PHP5)
- Muy popular el desarrollo con la herramienta XAMPP (Apache, MariaDB, PHP y Perl)

Tecnologías en servidor

ASP

- **Active Server Pages (ASP):** Lenguaje de Script, con un intérprete (no compilado)
 - Es una tecnología de Microsoft, en el lado del servidor, para crear páginas web dinámicas
 - Utiliza el servidor IIS (Internet Information Server)
 - El código de ASP se puede mezclar con el código HTML de la página web
 - El código ASP se ejecuta en el servidor, y produce en su salida un código HTML (compatibilidad con navegadores)

Tecnologías en servidor

ASP.NET

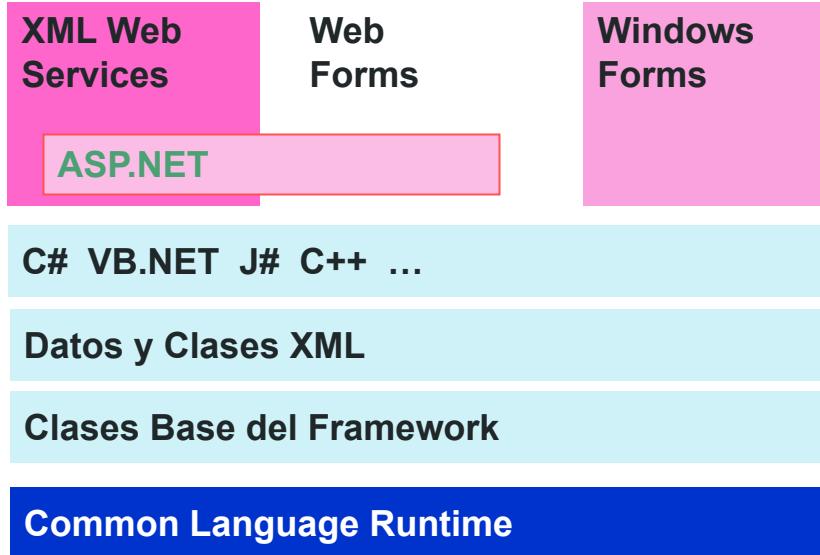
- **ASP. NET:** plataforma para construir aplicaciones Web.
- Forma parte del .NET Framework y contiene un conjunto de librerías
- Soporte de múltiples lenguajes: *Orientados a Objetos, compilados y dirigidos por Eventos. Utilizaremos C#*
- Para desarrollar aplicaciones ASP.NET necesitamos:
 - Editor o entorno de desarrollo (Microsoft Visual Studio)
 - **.NET Framework instalado**
 - Un servidor web con IIS



Tecnologías en servidor

ASP.NET

Framework de Microsoft .NET



Capa de clases destinadas a servicios web, páginas web y formularios windows

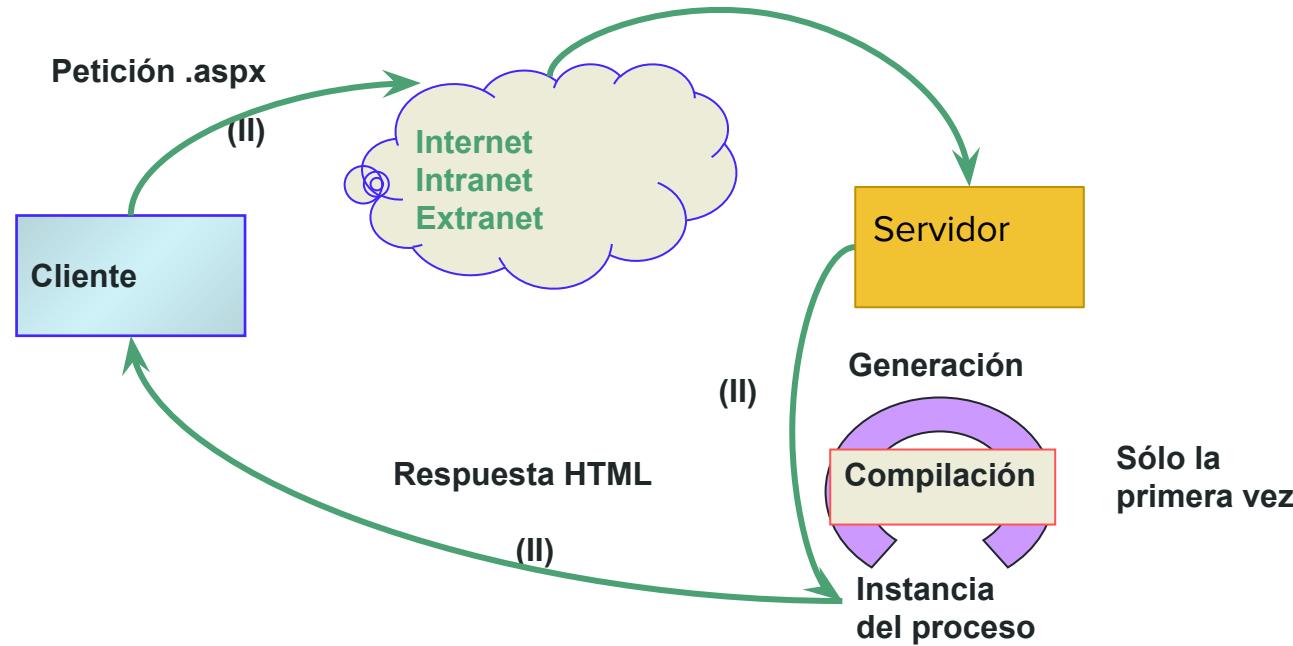
Capa de clases de datos y XML

Clases base del entorno

Tecnologías en servidor

ASP.NET

Llamada a una página ASP .NET



Servidores web

- Arquitectura cliente/servidor
 - Arquitectura de n-capas
 - Aplicaciones web
 - **Servidores web**
 - Metodología de diseño
 - Ejercicios
-

Internet Information Server (IIS)

- IIS es un servidor web
- IIS es un conjunto de servicios de Microsoft
- IS es una plataforma web unificada que integra:
 - el servidor web (HTTP/HTTPS),
 - ASP.NET, y
 - otros servicios como FTP, SMTP, NNTP
- También puede incluir PHP o Perl

Internet Information Server (IIS)

Ventajas

- + Ejecución de aplicaciones web en ASP, ASP.NET, y PHP en un mismo servidor
- + Confiable, seguro y fácil de utilizar
- + Posibilidad de agregar o eliminar componentes IIS integrados e incluso reemplazarlos por módulos personalizados
- + Aumenta la velocidad del sitio web mediante el almacenamiento en caché dinámico integrado y mejora de la compresión
- + Soporte técnico por parte de Microsoft

Desventajas

- No se recomienda su uso para desplegar aplicaciones en PHP, Python, Perl o Ruby (que se ejecutan de forma óptima en Linux y UNIX)
- Su licencia no es gratuita
- No es multiplataforma (sólo Windows)
- Código fuente propietario

APACHE

- Es un servidor web de código abierto multiplataforma
- Es el más utilizado en el mercado del hosting



Ventajas

- + Software de código abierto (instalación y configuración adaptable mediante módulos)
- + Sin coste de licencia
- + Buen soporte debido a que es el más utilizado por lo que muchos programadores contribuyen con mejoras
- + Multiplataforma (Windows, Linux y MacOS)
- + Buena interacción con PHP y MYSQL

Desventajas

- No posee interfaz gráfica
- No contiene una configuración estándar. Puede ser adaptada a cada aplicación
- No hay soporte técnico
- Difícil de administrar y configurar
- Pocas actualizaciones

Metodología de diseño de aplicaciones web

- Arquitectura cliente/servidor
 - Arquitectura de n-capas
 - Aplicaciones web
 - Servidores web
 - **Metodología de diseño**
 - Ejercicios
-

Metodología de diseño

- Es recomendable seguir una metodología en el diseño de una aplicación web.
- Existen diferentes metodologías.
- Nosotros proponemos la siguiente metodología:

1

Análisis de requisitos

Es un punto muy importante porque hay que analizar todo lo que nos han pedido y los objetivos

2

Elección de las tecnologías web a usar

HTML, CSS, ASP, ASP.NET, PHP, etc.

3

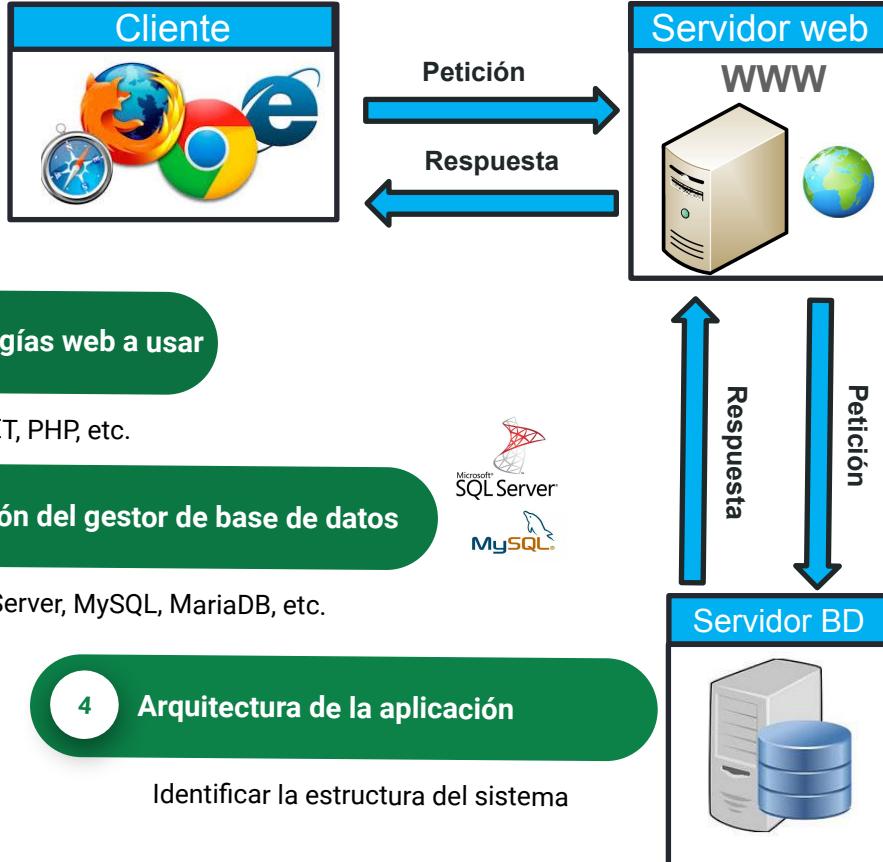
Elección del gestor de base de datos

SQL Server, MySQL, MariaDB, etc.

4

Arquitectura de la aplicación

Identificar la estructura del sistema



Metodología de diseño

5

Diseño de estructura y mapa de navegación web

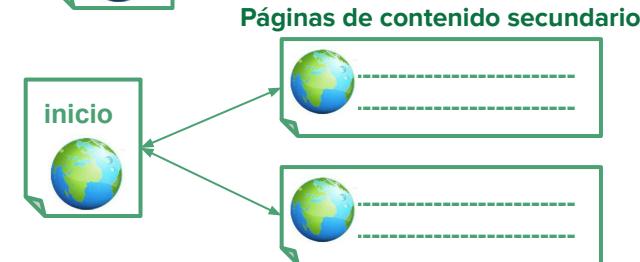
Se definen las páginas web que se van a utilizar y sus navegación en el sitio web



6

Creación del contenido de las páginas

Se implementan las páginas estáticas de la web



7

Diseño gráfico

Se añade la maquetación mediante CSS



8

Generación de scripts para controles en el cliente

Se implementa el código Javascript



JavaScript

Metodología de diseño

9

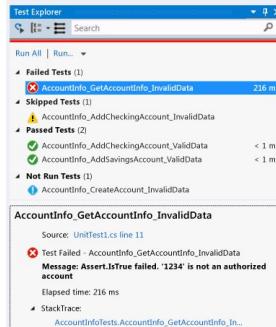
Diseño y desarrollo de páginas dinámicas

ASP /
ASP.NET

PHP

10

Validación y pruebas



Pruebas
funcionales

Pruebas de
Integración

Pruebas
Unitarias

11

Despliegue de la solución



Ejercicios

- Arquitectura cliente/servidor
 - Arquitectura de n-capas
 - Aplicaciones web
 - Servidores web
 - Metodología de diseño
 - **Ejercicios**
-

Ejercicios

1. ¿Qué son las páginas con extensión .htm? ¿Es lo mismo .html que .htm?
Razona la respuesta.
2. ¿Cuál es el lenguaje estándar para aplicar estilos de presentación a nuestras páginas web?
3. ¿En qué lugar se ejecuta el código JavaScript?
4. ¿Es mejor utilizar PHP o ASP? ¿Por qué?

Hada T4: Tecnologías web

Tecnologías web.

Objetivos

- Aprender las tecnologías fundamentales de programación web
- HTML
- CSS
- ASP.NET
- Programación de aplicaciones web con visual studio .NET

HTML

- **HTML**
 - ASP . NET
 - CSS
-

HTML

- Hypertext Markup Language
- **Hypertext**
 - Es texto ordinario al que se le incorporan funcionalidades adicionales como el formato, imágenes, multimedia y enlaces a otros documentos.
- **MarkUp**
 - **Es el proceso de tomar el texto ordinario e incorporar símbolos adicionales.**
Cada uno de estos símbolos identifica a un comando que le indica al navegador cómo mostrar ese texto.

Elementos HTML

- Los elementos son los componentes fundamentales de HTML
- Cuentan con dos propiedades básicas:
 - **Atributo**
 - **Contenido**
- En general se conforman con una *Etiqueta* de apertura y otra cierre
- Los **atributos** se colocan dentro la etiqueta de apertura, y el **contenido** se coloca entre la etiqueta de apertura y la de cierre

<ETIQUETA

ATRIBUTOS

>

CONTENIDO

</ETIQUETA>

Atributos de elementos

- Los atributos de un elemento son pares de nombres y valores separados por un '=' que se encuentran dentro de la etiqueta de apertura de algún elemento. Los valores deben estar entre comillas

```
<span id='iddeesteelemento' style='color:red;' title='Curso de HTML'>  
    Curso de HTML  
</span>
```

Estructura de página HTML

```
<HTML>
  <HEAD>
    <TITLE> Mi web! </TITLE>
  </HEAD>
  <BODY>
    <H1> Hello World </H1>
    <! Rest of page goes here. This is a comment. >
  </BODY>
</HTML>
```

Elementos principales

- Etiquetas que definen la estructura del documento
- **<HTML>... </HTML>**

Delimita el Documento HTML

- **<HEAD> ... </HEAD>**

Delimita el encabezado del Documento HTML

En general incluye los metadatos del documento y Scripts.

- **<BODY> ... </BODY>**

Delimita el Cuerpo del Documento HTML.

Es donde se incluyen los contenidos visibles del documento.

Algunos elementos posibles en HEAD

- **<TITLE> ... </TITLE>**

Define el título del documento HTML

- **<SCRIPT> ... </SCRIPT>**

Se utiliza para incluir programas al documento. En general se tratan de Javascripts.

- **<STYLE> ... </STYLE>**

Especifica un estilo CSS para ser utilizado en el documento.

- **<META> ... </META>**

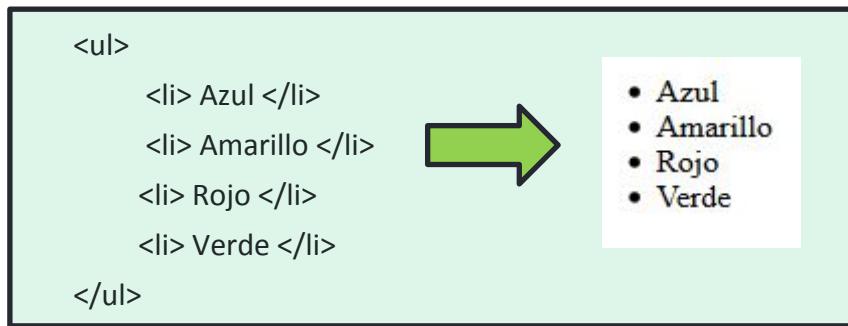
Permite especificar información de interés como: autor, fecha de publicación, descripción, palabras claves, etc.

Más etiquetas

Bloques de texto	Características texto	Listas	Scripts
<div>			<script>
<h1> ..<h6>			
<p>	<sub>		Semánticos
<blockquote>	<sup>	<dl>	<section>
<address>	<small>	<dt>	<article>
<pre>		<dd>	<header>
			<footer>
Imágenes y mapas	Tablas	Formularios	<nav>
	<table>	<form>	<aside>
<map>	<tr>	<input>	<figure>
<area>	<td>	<select>	<figcaption>
Enlaces	<th>	<option>	<summary>
<a>		<textarea>	

Listas no ordenadas (1)

- Listas no ordenadas ...
- Para añadir un elementos a una lista utilizaremos la etiqueta ...



Listas no ordenadas (2)

- El atributo ***type***, que permitía cambiar el tipo de marca (*type="circle"*, *type="square"*, *type="disc"*), no forma parte de HTML5 (uso de CSS), aunque sigue siendo compatible

```
<ul>
    <li type="circle"> Azul </li>
    <li type="square"> Amarillo </li>
    <li type="disc"> Rojo </li>
</ul>
```



- Azul
- Amarillo
- Rojo

- Utilización de CSS (Style)

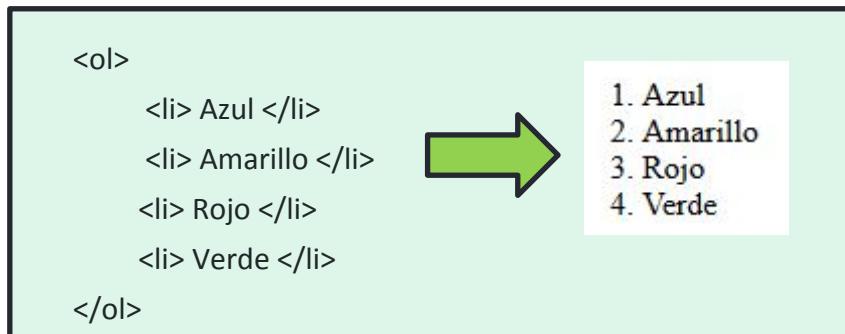
```
<ul>
    <li style="list-style-type:circle"> Azul </li>
    <li style="list-style-type:square"> Amarillo </li>
    <li style="list-style-type:disc"> Rojo </li>
</ul>
```



- Azul
- Amarillo
- Rojo

Listas ordenadas (1)

- Listas ordenadas ...
- Para añadir un elementos a una lista utilizaremos la etiqueta ...



Listas ordenadas (2)

- El atributo **type**, permite modificar el tipo de numeración (`type="1"`) por letras (`type="a"`) o números romanos (`type="i"`).
- También es posible modificar la lista para que el primer punto empiece por otro número (tipo de numeración de números), utilizando el atributo **start** en `ol` (`start="25"`, *etc.*).

```
<ol type="a">
  <li> Azul </li>
  <li> Amarillo </li>
  <li> Rojo </li>
  <li> Verde </li>
</ol>
```

a. Azul
b. Amarillo
c. Rojo
d. Verde



```
<ol start="25">
  <li> Azul </li>
  <li> Amarillo </li>
  <li> Rojo </li>
  <li> Verde </li>
</ol>
```

25. Azul
26. Amarillo
27. Rojo
28. Verde



Ejercicio 1

- Crea una página web que tenga el mismo aspecto que la siguiente imagen

PROGRAMACIÓN WEB

Tema 1: HTML

1. Listas
 - Listas ordenadas
 - Listas no ordenadas
2. Tablas
3. Formularios

Codigo...

```
<HTML>
```

```
  <HEAD>
```

```
    <TITLE> Ejercicio 1 HADA </TITLE>
```

```
  </HEAD>
```

```
  <BODY>
```

```
    <!lista ordenada>
```

```
      <!lista desordenada>
```

```
    </BODY>
```

```
</HTML>
```

Enlaces

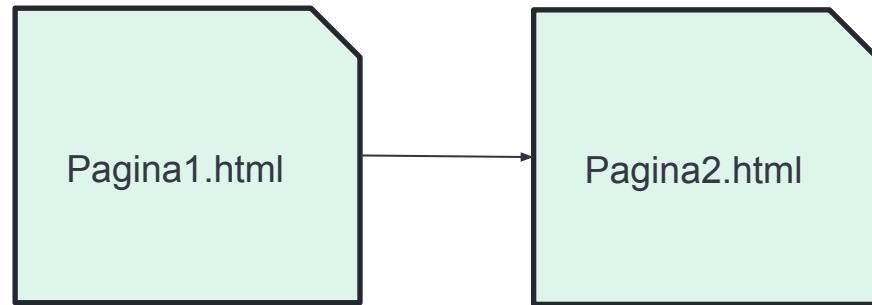
- La etiqueta `<a>...`, inserta un enlace.

```
<a href="destino_del_enlace">Texto del enlace </a>
```

- El destino del enlace puede ser a
 - Un lugar de la página en curso
 - Otra página del sitio web
 - Algún lugar de otra página del sitio web
 - Una página de otro sitio existente en la web
 - Una dirección de correo electrónico
 - Una archivo para descargas

Enlaces (2)

Enlace a otra página -- Cómo sería???



El código del enlace en la página 1 es:

Tablas (1)

- Creación de una tabla <table>...</table>
- Filas <tr>...</tr>
- Celdas <td>...</td>
- Celdas cabecera <th>...</th>
- Antes de aparecer CSS, las tablas eran fundamentales. **Hoy en día no se recomienda su uso para organizar el diseño.**
- Usar CSS.
- Combinación de celdas: colspan, rowspan

```
<table>
  <tr>
    <th> Asignatura </th>
    <th> Créditos </th>
  </tr>
  <tr>
    <td> Programación 1</td>
    <td> 6 </td>
  </tr>
  <tr>
    <td> Base de datos </td>
    <td> 6 </td>
  </tr>
</table>
```



Asignatura	Créditos
Programación 1	6
Base de datos	6

Tablas (2)

Ejemplo de tabla

```
<body>
  <table>
    <tr>
      <th>Código</th>
      <th>Municipio</th>
    </tr>
    <tr>
      <td>1</td>
      <td>Agres </td>
    </tr>
    <tr>
      <td>2</td>
      <td>Alcoy </td>
    </tr>
    <tr>
      <td>3</td>
      <td>Torremanzanas </td>
    </tr>
  </table>
</body>
</html>
```



Código	Municipio
1	Agres
2	Alcoy
3	Torremanzanas

Tablas (2)

Ejemplo de tabla utilizando css

```
<!DOCTYPE html>
<head>
    <title> Ejemplo de tabla con CSS</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <style>
        table { text-align: center;
                 border: 1px solid black;
                 border-collapse: collapse;
                 width: 300 px; }
        td {border: 1px solid black;
            background-color: #99ccff; }
    </style>
</head>
```

Código	Municipio
1	Agres
2	Alcoy
3	Torremanzanas



Para la tabla (table):

- **Text-align:** alineación de las celdas de la tabla
- **Border:** borde de la tabla
- **Border-collapse:** para eliminar el espacio entre el borde de la tabla y los bordes de las celdas
- **Width:** ancho de la tabla

Para las celdas (td):

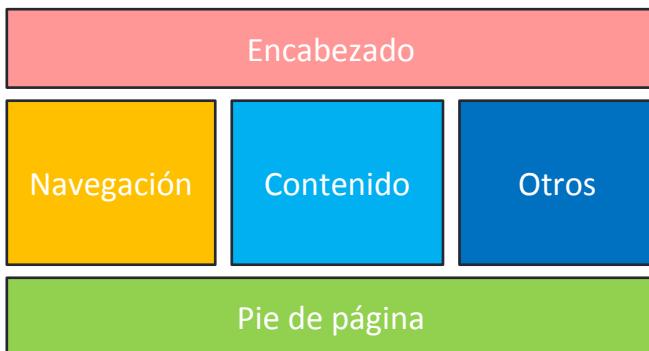
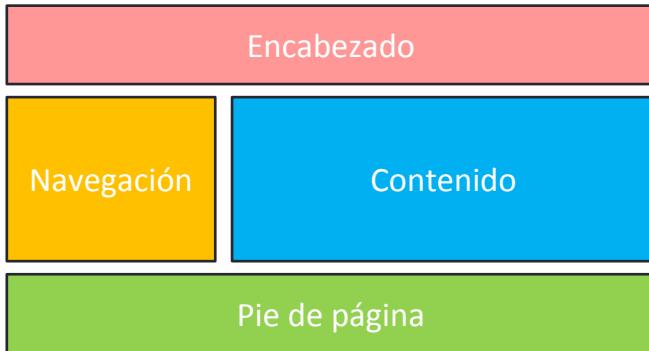
- **Border:** borde de la celda
- **Background-color:** color del fondo de la celda

Etiquetas de organización

- Son etiquetas que permiten formar bloques, que describen mejor las partes de un documento
- Todas las web están formadas por:
 - Encabezado de página
 - Herramientas de navegación
 - Contenido
 - Zona anexa de elementos asociados al contenido (publicidad)
 - Pie de página

Etiquetas de organización (2)

Ejemplos de organización en una web



Etiquetas de organización (3)

- **<header> ... </header>**

Agrupa los elementos del encabezado de la página
- **<nav> ... </nav>**

Indica los elementos del menu de navegación
- **<footer> ... </footer>**

Agrupa los elementos del pie de página
- **<aside> ... </aside>**

Indica que se tratan de elementos anexos al contenido

Etiquetas de organización (3)

- **<section> ... </section>**

Indica que una parte del contenido de la página se refiere a un tema en concreto

- **<article> ... </ article>**

Define un contenido del documento que posee una identidad independiente dentro de la página (artículo de un blog, un post en un foro, o un producto en la web de un comercio)

- **<figure> ... </figure>**

Permite incorporar una sección de imagen.

- **<figcaption> ... </figcaption>**

Contiene el texto explicativo de una imagen asociada

Etiquetas de organización (4)

Ejemplo

```
<!DOCTYPE html>
<html>
    <head>
        <title> Nuevas secciones en HTML 5 </title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <link rel="stylesheet" href="css/estilo.css">
    </head>
    <body>
        <header>
            
            <h1>EJEMPLO SENCILLO EN HTML5</h1>
        </header>

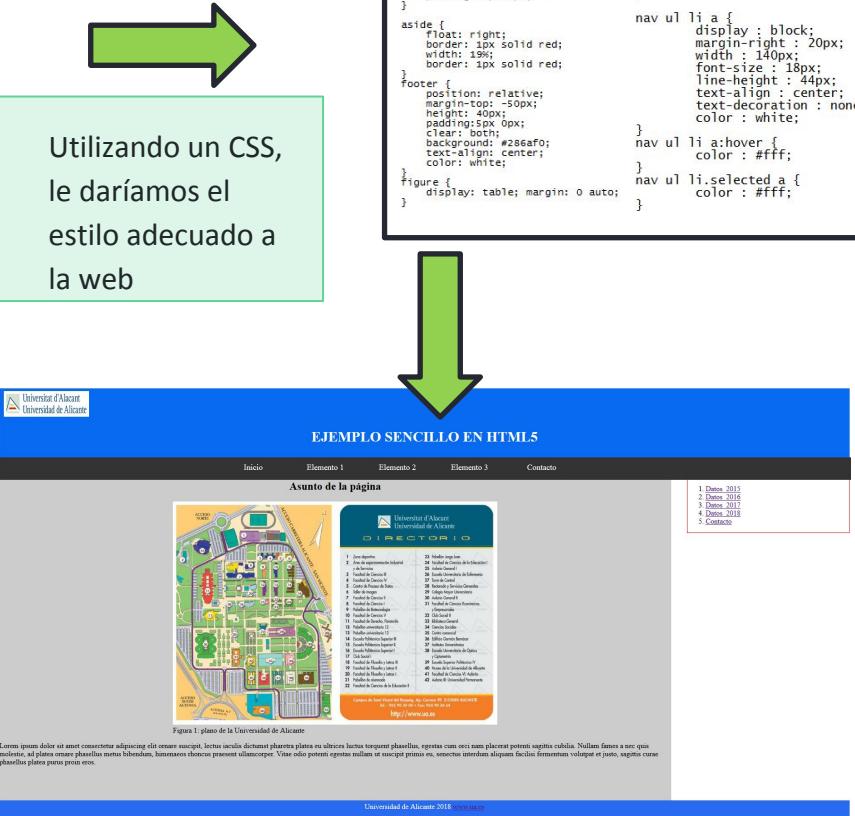
        <nav>
            <ul>
                <li> <a href="">Inicio</a> </li>
                <li> <a href="">Elemento 1</a> </li>
                <li> <a href="">Elemento 2</a> </li>
                <li> <a href="">Elemento 3</a> </li>
                <li> <a href="">Contacto </a> </li>
            </ul>
        </nav>

        <section>
            <h1> Asunto de la página </h1>
            <figure>
                
                <figcaption>Figura 1: plano de la Universidad de Alicante</figcaption>
            </figure>

            <article>
                <p>Lorem ipsum dolor sit amet consectetur adipiscing elit ornare suscipit, lectus iacu
placerat potenti sagittis cubilia. Nullam fames a nec quis molestie, ad platea ornare phasellus metus bibendum, himena
senectus interdum aliquam facilisi fermentum volutpat et justo, sagittis curae phasellus platea purus proin eros. </p>
            </article>
        </section>

        <aside>
            <h3>Archivos</h3>
            <ol>
                <li> <a href="">Datos_2015</a> </li>
                <li> <a href="">Datos_2016</a> </li>
                <li> <a href="">Datos_2017</a> </li>
                <li> <a href="">Datos_2018</a> </li>
                <li> <a href="">Contacto </a> </li>
            </ol>
        </aside>

        <footer>
            Universidad de Alicante 2018 <a href="http://www.ua.es">www.ua.es</a>
        </footer>
    </body>
</html>
```



Formularios

- Recaba la información introducida por el usuario
- La información es enviada a una página (normalmente al servidor) después de pulsar en un botón
- El servidor recibe la información como pares de datos, con cada valor asociado a cada nombre de control
- El servidor procesa dichos datos, y responde el resultado
- Nosotros utilizaremos los formularios de ASP. NET
- Se define mediante la etiqueta `<form> ... </form>`

Formularios (2)

Atributos de la etiqueta form

- name: asignar un nombre al formulario
- action: acción que debe realizar (web, correo, programa)
- enctype: especifica el formato
(application/x-www-form-urlencoded)
- method: el método para mandar la información

```
<!DOCTYPE html>
<head>
    <title> Ejercicio formulario 1</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
    <form method="get" action="ejercicio2.html">
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" size="20"> <br>
        <label for="username">Apellidos:</label>
        <input type="text" id="apellidos" size="40"> <br>
        <button>Enviar</button>
    </form>
</body>
</html>
```



Nombre:

Apellidos:

Formularios (3)

- Dos opciones para enviar la información (method):
 - GET. Los datos se envían (caracteres ASCII) dentro de la propia URL, unidos por &

```
http://www.miweb.com/index.aspx?id=1&nombre=pepe&apellidos=perez
```

Para enviar poco texto, datos que no requieran de seguridad, sencillez

- POST. Los datos no se envía en la URL y son invisibles para el usuario

```
http://www.miweb.com/index.aspx
```

Para enviar grandes campos de texto, imágenes, URL más legibles, seguridad de información

Formularios (4)

Campo de texto

```
<input type="text">
```



maxlength	size
readonly	required
placeholder	autofocus
Pattern	

```
<form method="get" action="ejercicio2.html">
    Nombre:
    <input type="text" name="nombre" placeholder="Nombre" size="20"> <br>
    Apellidos:
    <input type="text" name="apellidos" placeholder="Apellidos" size="40"> <br>
    E-Mail:
    <input type="text" name="mail" required> <br>
    Código postal:
    <input type="text" name="postal" pattern="[0-9]{5}"> <br>

    <button>Enviar</button>
</form>
```

Nombre:

Apellidos:

E-Mail:

Código postal:

Nombre:

Apellidos:

E-Mail:

Código postal:

Rellene este campo.

Nombre:

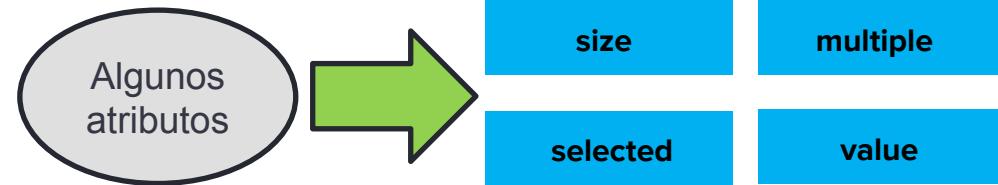
Apellidos:

E-Mail:

Código postal:

Ajustese al formato solicitado.

Formularios (5)



Lista desplegable

- <select> ... </select> indica el uso de una lista desplegable
- <option> ...</option> forma los elementos de la lista

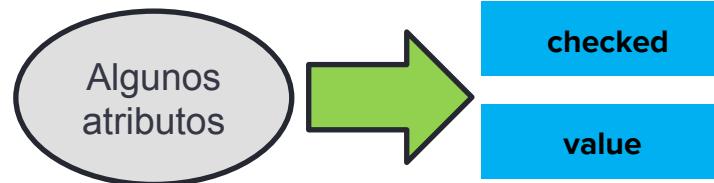
```
<!DOCTYPE html>
<head>
  <title> Ejercicio formulario 3</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <form action="">
    <p>Asignatura preferida: </p>
    <select>
      <option value="PR1">Programación 3 </option>
      <option value="SIO">Sistemas operativos</option>
      <option value="FDB">Diseño de base de datos</option>
      <option value="RED">Redes de computadores</option>
      <option value="HAD" selected>Herramientas avanzadas para el desarrollo de aplicaciones</option>
      <option value="ARC">Arquitectura de los compiladores</option>
    </select>
  </form>
</body>
</html>
```

A screenshot of a web browser showing a dropdown menu for "Asignatura preferida". The menu contains several options: "Herramientas avanzadas para el desarrollo de aplicaciones" (selected), "Programación 3", "Sistemas operativos", "Diseño de base de datos", "Redes de computadores", "Herramientas avanzadas para el desarrollo de aplicaciones" (highlighted in blue), and "Arquitectura de los compiladores". A green arrow points from the code block to the dropdown menu.

Formularios (6)

Botones de selección única

<input type="radio">



```
<!DOCTYPE html>
<head>
    <title> Ejercicio formulario 4</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>

<body>
    <form action="ejercicioFormulario4.html">
        <p>Sexo:</p>
        <input type="radio" name="sexo" value="H">Hombre <br>
        <input type="radio" name="sexo" value="M">Mujer <br>
        <button>Enviar</button>
    </form>
</body>
</html>
```

Sexo:

Hombre
 Mujer

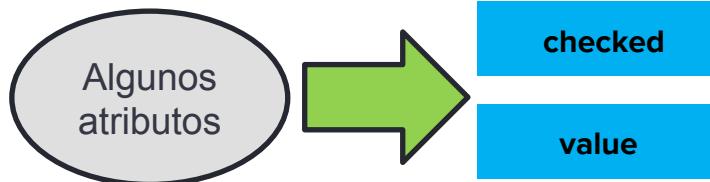
- Cuando pulsemos en el botón *Enviar*, se enviarán los datos con el valor establecido con *value*.

A green arrow points from the "Enviar" button in the form to two URLs:
ejercicioFormulario4.html?sexo=H
ejercicioFormulario4.html?sexo=M

Formularios (7)

Botones de selección múltiple

<input type="checkbox">



```
<!DOCTYPE html>
<head>
  <title> Ejercicio formulario 5</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <form action="ejercicioFormulario5.html">
    <p>Selección de asignaturas matriculadas:</p>
    <input type="checkbox" name="a1">Sistemas operativos <br>
    <input type="checkbox" name="a2" checked>Diseño de base de datos <br>
    <input type="checkbox" name="a3">Redes de computadores <br>
    <input type="checkbox" name="a4" checked>Herramientas avanzadas para el desarrollo de aplicaciones <br>
    <input type="checkbox" name="a5">Arquitectura de los computadores <br>
    <button>Enviar</button>
  </form>
</body>
</html>
```

Selección de asignaturas matriculadas:

- Sistemas operativos
 Diseño de base de datos
 Redes de computadores
 Herramientas avanzadas para el desarrollo de aplicaciones
 Arquitectura de los computadores

- Cuando pulsemos el botón *Enviar*, se enviarán los datos con el valor a on de cada opción seleccionada.
- Si tiene el attribute value reemplazará el on por el valor asociado.

ejercicioFormulario5.html?a2=on&a4=on

ejercicioFormulario5.html?a2=on&a4=HADA

Formularios (8)

Otras elementos del formulario

- Botón de envío
- Botón de anulación
- Área de texto
- Campos ocultos
- Contraseñas
- Campos de transferencia de archivos
- Etiquetas de campos

- Campo de texto URL
- Campo de texto con lista de sugerencia
- Campo de texto para dirección de correo
- Campo de texto numérico
- Campo de texto de búsqueda
- Campo de texto de fecha
- Campo de texto de color
- Cursores

ASP .NET

- HTML
 - **ASP . NET**
 - CSS
-

Índice

1. Introducción a las aplicaciones web ASP.NET
2. Formularios
3. Controles de servidor
4. Páginas maestras
5. Eventos de los controles del servidor
6. Navegación entre WebForms

Introducción a las aplicaciones web ASP.NET

1

Aplicaciones Web ASP.NET

- Combinación de archivos, páginas, manejadores, módulos y código ejecutable que puede invocarse desde un **directorio virtual**.
- Se dividen en varias páginas web.
- Comparten un conjunto de recursos y opciones de configuración común.
- Cada aplicación tiene su propio:
 - Conjunto de caché
 - Datos de estado de sesión

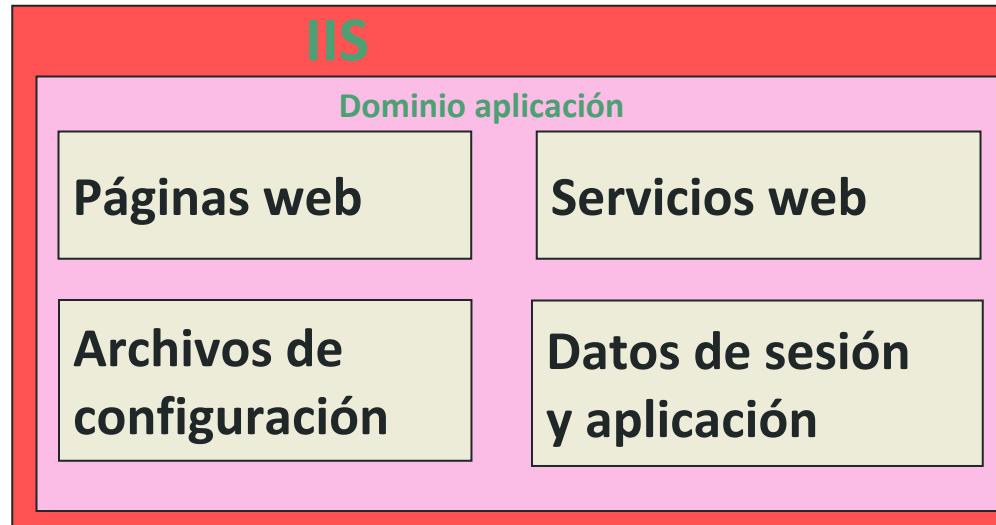
Dónde se guarda la aplicación??

→ Directorio Virtual

- Una aplicación web sólo existe en una localización que ha sido publicada por IIS como un **Directorio Virtual**.
- Un **directorio virtual** es un recurso compartido identificado por un alias que representa la localización física en el servidor.
- **//localhost** es la carpeta virtual raíz del ordenador
(\InetPub\wwwroot)

Directorio virtual

- Directorio virtual: estructura de agrupación básica que delimita una aplicación.
- Creación y administración desde IIS (Internet Information Server)



En VisualStudio...

Sitio Web o Proyecto Web?

- **Sitio Web:** conjunto de páginas Web independientes.
 - Para páginas Web sencillas (ej, página Web personal...)
- **Proyecto Web:** conjunto de páginas Web enlazadas con un archivo de proyecto.
 - Para aplicaciones avanzadas
 - Podemos referenciar DLLs, etc

Necesitamos Internet Information Server?

- Visual Studio dispone de su propio servidor de desarrollo, por lo que para hacer pruebas en nuestro ordenador no necesitamos tener instalado IIS.
- Sin embargo, para poder desplegar nuestra aplicación en un servidor, si lo necesitaríamos.

En Visual Studio...

Dónde se guarda la aplicación?

- File system
- C:\Documents and Settings\aaaa\Mis documentos\Visual Studio 2005\WebSites\WebSite2
 - http://localhost:3371/WebSite2/Default.aspx
- Local IIS: Carpeta del sitio web por defecto (<http://localhost>)
 - Por ejemplo C:\Inetpub\wwwroot\WebSite1

Code-inline vs Code-behind

- “Etiquetas” declarativas
 - HTML, controles del servidor, texto estático
- A diferencia de ASP, buena separación entre el código y las etiquetas

Único archivo
("Code-inline")

código

<etiquetas>

Form1.aspx

Archivos separados ("Code-behind")

<etiquetas>

Form1.aspx

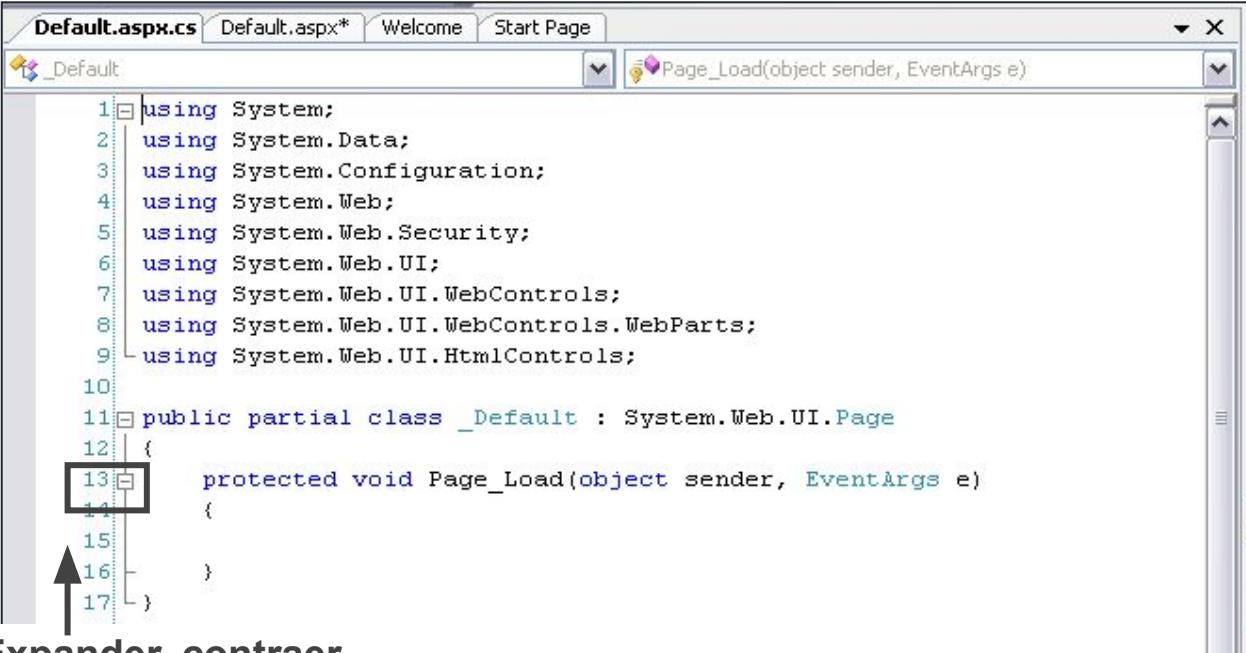
código

Form1.cs

Code-behind

- El código para manejar eventos se sitúa en un **archivo físico separado** de la página que contiene los controles de servidor y las etiquetas.
 - Extensión *.aspx*
 - Extensión *.cs, .vb ... (code-behind)*
- UTIL mantenerlo por separado :
Es común en grupos de proyectos tener
 - **diseñadores** trabajando en la IU de la aplicación
 - y **desarrolladores** en el comportamiento o código.

Ver código (C#)



```
Default.aspx.cs Default.aspx* Welcome Start Page _Default.cs
  Page_Load(object sender, EventArgs e)

1  using System;
2  using System.Data;
3  using System.Configuration;
4  using System.Web;
5  using System.Web.Security;
6  using System.Web.UI;
7  using System.Web.UI.WebControls;
8  using System.Web.UI.WebControls.WebParts;
9  using System.Web.UI.HtmlControls;
10
11 public partial class _Default : System.Web.UI.Page
12 {
13     protected void Page_Load(object sender, EventArgs e)
14     {
15     }
16 }
17
```

Expander, contraer...

Definición de una clase parcial

Procedimiento que responde al evento LOAD de la página

Visual Studio IDE showing the design and code-behind for a web page.

Left Panel (Toolbox):

- Estándar
- Puntero
- A Label
- abl TextBox
- ab Button
- ab LinkButton
- ab ImageButton
- HyperLink
- DropDownList
- ListBox
- CheckBox
- CheckBoxList
- RadioButton
- RadioButtonList
- Image
- ImageMap
- Table
- BulletedList
- HiddenField
- Literal
- Calendar

Middle Panel (Properties):

Button1 System.Web.UI.WebControls.Button

(Expressions)	
(ID)	Button1
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
CausesValidation	True
CommandArgument	
CommandName	
CssClass	
Enabled	True
EnableTheming	True
EnableViewState	True
Font	
ForeColor	
Height	
OnClientClick	
PostBackUrl	
SkinID	
TabIndex	0
Text	Button
ToolTip	
UseSubmitBehavior	True
ValidationGroup	
Visible	True

Right Panel (Code-Behind):

```
using System.Web.Security;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.
{
    protected void Page_Load(object sender,
    {
        this.
    }
}
```

The Intellisense dropdown shows the available methods for the Button1 control:

- AppRelativeTemplateSourceDirectory
- AppRelativeVirtualPath
- AsyncTimeout
- BuildProfileTree
- Button1
- Cache
- ChildControlsCreated
- ClearChildControlState
- ClearChildState
- ClearChildViewState

Formulario ASP.NET

2

Formularios Web (I)

- Técnicas para Rápido Desarrollo de Aplicaciones (RAD)
- Podemos:
 - Arrastrar y soltar controles en un formulario
 - Escribir el código soporte
- La aplicación se desarrolla para un servidor web
- Los usuarios interactúan con la aplicación a través de un navegador
- Proporcionan una aproximación orientada a:

Objetos

Eventos

Gestión de estado

- pueden ejecutarse, virtualmente, sobre cualquier navegador compatible con HTML.
- **Programación del lado del servidor para manejar eventos del lado del cliente**

Formularios Web (III)

- Todos los controles de servidor deben aparecer dentro de una etiqueta <form>, y esta etiqueta tiene que contener el atributo **runat="server"**.
- Este atributo indica que el formulario se debe procesar en el servidor.
- También indica que los controles que contiene pueden ser accedidos por scripts del servidor:

```
<form runat="server">  
    ...HTML + controles de servidor  
</form>
```

- Una página

runat="server">

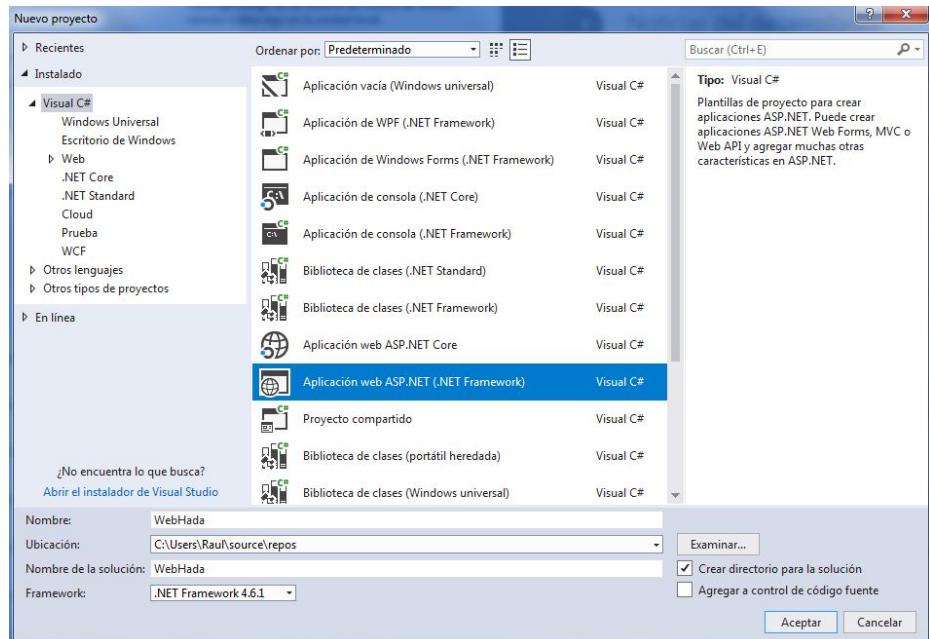
Aplicación web utilizando Web Forms

- **Vamos a crear nuestra primera aplicación web ASP.NET utilizando Web Forms.**
- Para ello utilizaremos **Visual Studio**, de manera que crearemos una aplicación web ASP.NET, y un formulario que permita al usuario introducir sus datos personales, validar los datos y si todo está correcto, cuando pulse en el botón *Enviar datos*, se enviarán los datos a un servidor.
- En el formulario web el usuario podrá introducir la siguiente información: **Nombre, apellidos, dirección, código postal, sexo y correo electrónico**

Aplicación web utilizando Web Forms

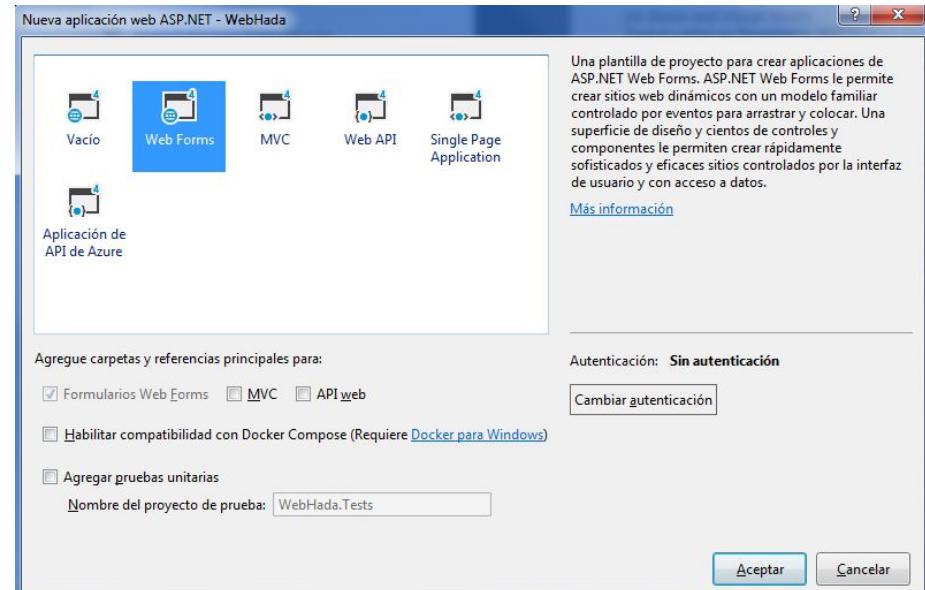
Creación del proyecto

- Para crear el proyecto desde el entorno de Visual Studio, hay que acceder al menú *Archivo -> Nuevo -> Proyecto*.
- Dentro del menú *Visual C#*, acceder al menú *Web* y seleccionar como tipo de aplicación **Aplicación web ASP.NET(.NET Framework)**.
- Damos un nombre al proyecto, en este caso **WebHada**.



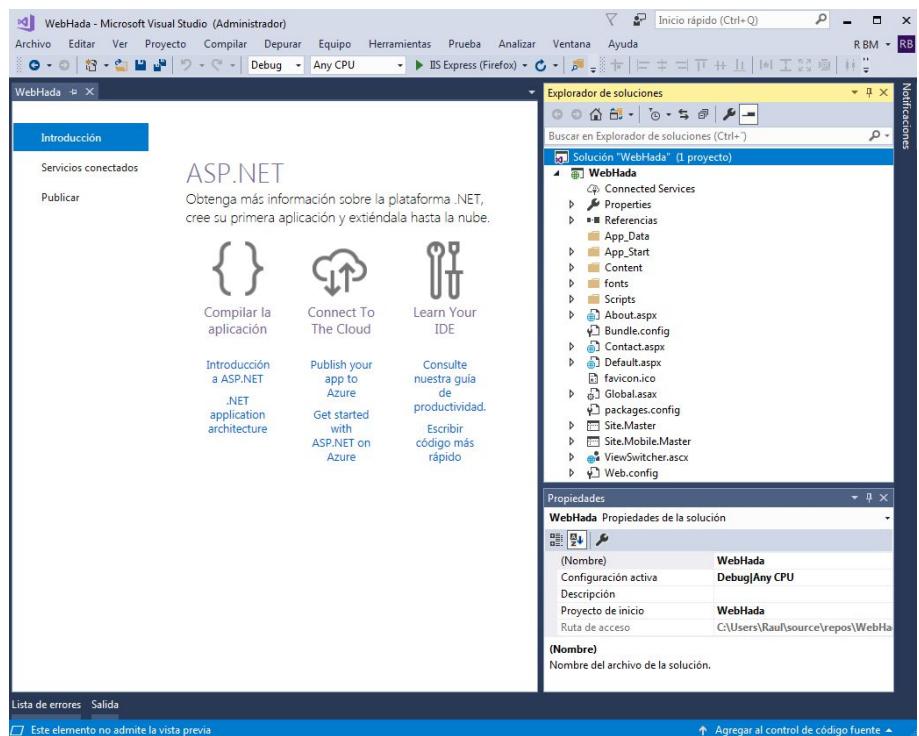
Aplicación web utilizando Web Forms

- A continuación, aparece la pantalla para seleccionar la plantilla en que se basa la aplicación web. En este caso seleccionamos *Web Forms*.
- Al pulsar en *Aceptar*, se creará el proyecto.
- Hay que tener en cuenta, que este proyecto se creará dentro de una solución con el mismo nombre que el proyecto.



Aplicación web utilizando Web Forms

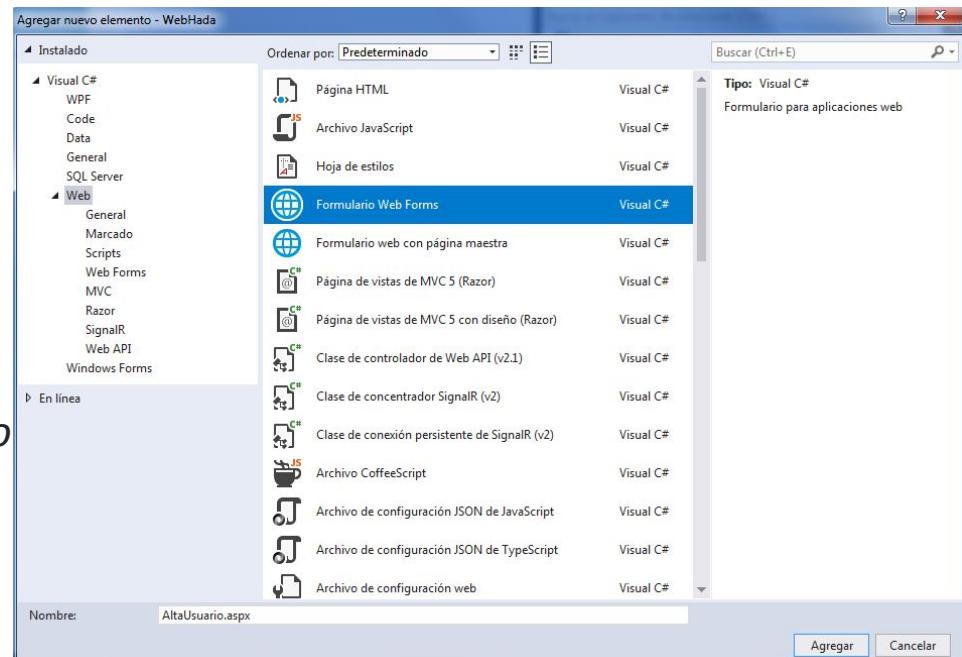
- Ya tenemos creado el proyecto base a partir del cual trabajaremos
- El explorador de soluciones, muestra el contenido de la solución (proyectos de la solución y elementos de cada proyecto).
- Propiedades de los elementos seleccionados



Aplicación web utilizando Web Forms

Creación del formulario

- Desde el explorador de soluciones, pulsamos con el botón derecho del ratón sobre el nombre del proyecto WebHADA.
- Aparece un menú contextual, del que elegiremos la opción *Agregar -> Nuevo elemento*.
- Elegiremos la opción **Formulario Web Forms**, y le daremos el nombre de *AltaUsuario.aspx*



Aplicación web utilizando Web Forms

Visualización del formulario

- El IDE dispone de tres botones, que permiten cambiar el modo de visualización del código de la aplicación:
 - **Código.** Muestra el código fuente equivalente que genera la página.
 - **Diseño.** Muestra una pantalla para diseñar nuestra aplicación de manera visual. Se pueden arrastrar los componentes (botones, etiquetas, imágenes, calendarios, datos, etc.) existentes en la barra de herramientas
 - **Dividir.** Muestra una parte de la pantalla el diseño y la otra parte el código.
- Cuando añadimos algún componente desde la pantalla de *Diseño*, automáticamente se genera el código equivalente. De la misma manera, si añadimos código desde la pantalla de *Código*, se genera automáticamente su equivalente a nivel visual en la pantalla *Diseño*.

Aplicación web utilizando Web Forms

Visualización del formulario

AltaUsuario.aspx

```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AltaUsuario.aspx.cs" Inherits="WebHada.Altausuario" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8   <title></title>
9 </head>
10 <body>
11   <form id="form1" runat="server">
12     <div>
13       </div>
14     </form>
15   </body>
16 </html>
```

AltaUsuario.aspx

```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AltaUsuario.aspx.cs" Inherits="WebHada.Altausuario" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8   <title></title>
9 </head>
10 <body>
11   <form id="form1" runat="server">
12     <div>
13       </div>
14     </form>
15   </body>
16 </html>
```

Solución "WebHada" [1 proyecto]

- WebHada
 - Connected Services
 - Properties
 - Referencias
 - App_Data
 - App_Start
 - Content
 - fonts
 - Scripts
 - About.aspx
 - AltaUsuario.aspx
 - Bundle.config
 - Com.aspx
 - Default.aspx
 - favicon.ico
 - Global.asax
 - packages.config
 - Site.Master
 - Site.Mobile.Master
 - ViewSwitcher.aspx

Propiedades

DOCUMENT

- Class
- Culture
- Debug
- EnableSessionState
- Id

Class

Define la clase del cuerpo de página.

Explorador de soluciones

Solución "WebHadas" [1 proyecto]

- WebHadas
 - Connected Services
 - Properties
 - Referencias
 - App_Data
 - App_Start
 - Content
 - fonts
 - Scripts
 - About.aspx
 - AltaUsuario.aspx
 - Bundle.config
 - Com.aspx
 - Default.aspx
 - favicon.ico
 - Global.asax
 - packages.config
 - Site.Master
 - Site.Mobile.Master
 - ViewSwitcher.aspx

Propiedades

DOCUMENT

- Class
- Culture
- Debug
- EnableSessionState
- Id

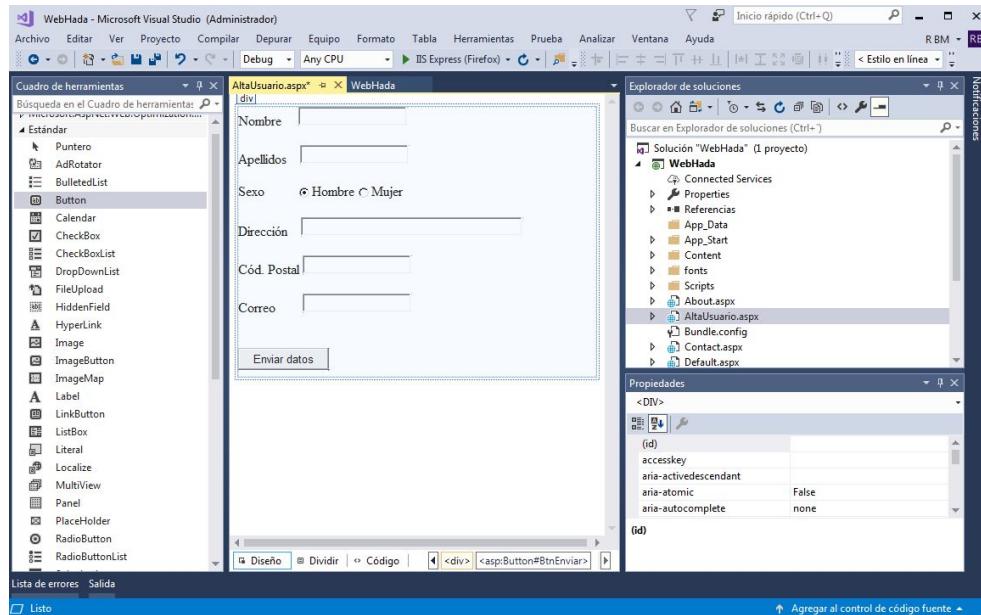
Class

Define la clase del cuerpo de página.

Aplicación web utilizando Web Forms

Modificar el formulario

- Incorporamos los elementos arrastrando los componentes de la barra de herramientas en la vista de diseño.
- Para visualizar la barra de herramientas pulsamos el menú Ver-Cuadro de herramientas



Aplicación web utilizando Web Forms

Código y ejecución

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AltaUsuario.aspx.cs" Inherits="WebHada.AltaUsuario" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div style="height: 318px">
        Nombre &nbsp;&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TBNOMBRE" runat="server"></asp:TextBox>
        <br />
        <br />
        Apellidos&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TBAPELLIDOS" runat="server"></asp:TextBox>
        <br />
        <br />
        Sexo&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        <asp:RadioButton ID="RBHombre" runat="server" Checked="True" Text="Hombre" />
        <asp:RadioButton ID="RBMujer" runat="server" Text="Mujer" />
        <br />
        <br />
        Dirección&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TBDIRECION" runat="server" Width="253px"></asp:TextBox>
        <br />
        <br />
        Cód. Postal <asp:TextBox ID="TBCODPOSTAL" runat="server"></asp:TextBox>
        <br />
        <br />
        Correo&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TBCORREO" runat="server"></asp:TextBox>
        <br />
        <br />
        <br />
        <asp:Button ID="BtnEnviar" runat="server" Text="Enviar datos" />
      </div>
    </form>
  </body>
</html>
```



Nombre

Apellidos

Sexo Hombre Mujer

Dirección

Cód. Postal

Correo

Controles de servidor

3

Los controles de servidor...

- Tienen propiedades que pueden ser establecidas
 - **declarativamente (en la etiqueta)**
 - o mediante programación (en el código).
- Junto con la página, **tienen eventos** que los desarrolladores pueden manejar
 - para ejecutar acciones específicas durante la ejecución de la página
 - o en respuesta a una acción del lado del cliente que envía la página al servidor.

Controles de servidor

- Soporte para características avanzadas: enlace de datos, plantillas..
- Se emplea el prefijo **asp:** junto con el atributo **runat="server"**.
`<asp:TextBox id="text" runat="server"/>`

Tipos de controles ASP.NET

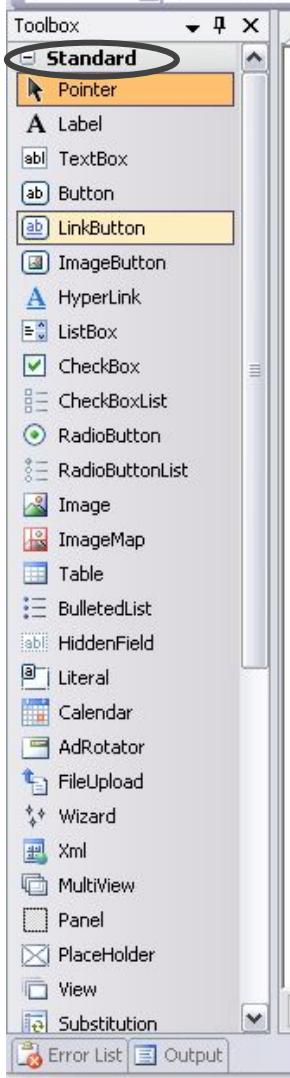
- 1. Controles HTML (no ASP.NET)
- **2. Controles estándar**
- **3. Controles de validación**
- 4. Controles de login
- **5. Controles de navegación**
- 6. Controles Webparts
- **7. Controles de datos**
- **8. Controles de usuario**

Principales controles web (estándar)

Label	Se utiliza para mostrar texto dinámico (cambiamos sus propiedades a través del código del servidor)
Hyperlink	Muestra un enlace a otra página.
TextBox	Permite introducir texto al usuario. Propiedad Textmode valores: Single, Multiline o Password.
Image	Sirve para mostrar una imagen en la página web.
Button	Se usa en un Formulario web para crear un control de tipo submit (evento OnClick) .
LinkButton	Apariencia de hipervínculo pero funciones de control de un botón (envío formulario)
ImageButton	Muestra una imagen que maneja eventos tipo click.
Checkbox	Sirven para añadir casillas de verificación a una página.
RadioButton	Crea un botón de radio individual en la página.

Principales controles web

DropDownList	Proporciona un buen método para que los usuarios elijan elementos de una lista en un espacio pequeño. Cada elemento se crea con un control ListItem.
ListBox	Propiedad SelectionMode: Single, Multiple. Cada elemento se crea con un control ListItem.
CheckBoxList	Permite presentar una lista de opciones pudiendo el usuario seleccionar varias.
RadioButtonList	Permite crear una lista de botones de radio de opciones excluyentes.
Panel	Puede usarse como contenedor de otros controles.
Table, TableRow, TableCell	Permiten crear dinámicamente una tabla mediante programación.



Controles estándar

- AdRotator
- PlaceHolder
- ImageMap
- BulletedList
- HiddenField
- FileUpload
- Wizard
- Xml
- MultiView
- Substitution...

TextBox

Página aspx

```
<form id="form1" runat="server">
<div>
<p>    Username: <asp:TextBox id="userTextBox" TextMode="SingleLine"
    Columns="30" runat="server" />
</p>
<p>    Password: <asp:TextBox id="passwordTextBox"
    TextMode="Password" Columns="30" runat="server" />
</p>
<p>    Comments: <asp:TextBox id="commentsTextBox"
    TextMode="MultiLine" Columns="30" Rows="10" runat="server" />
</p>
</div>
</form>
```

Username:	Sonia
Password:	*****
Comments:	Comentarios en varias lineas

Button

Página aspx

```
<form id="form1" runat="server">
<asp:Button id="BotonEnviar" Text="Enviar" runat="server" OnClick="WriteText" />
<asp:Label id="Label1" runat="server" />
</form>
```

Página aspx.cs

```
protected void WriteText(object sender, EventArgs e)
{
    Label1.Text = "Hola mundo";
}
```



ImageButton

Página aspx

```
<form id="form1" runat="server">
  <div>
    <asp:ImageButton id="BotonImagen" ImageUrl="~/garfield.gif" runat="server"
      OnClick="WriteText" />
    <asp:Label id="Label4" runat="server" />
  </div>
</form>
```

Página aspx.cs

```
protected void WriteText(object sender, ImageClickEventArgs e)
{
  Label4.Text = "Coordenadas:" + e.X + "," + e.Y;
}
```



Coordenadas:45,64

Panel

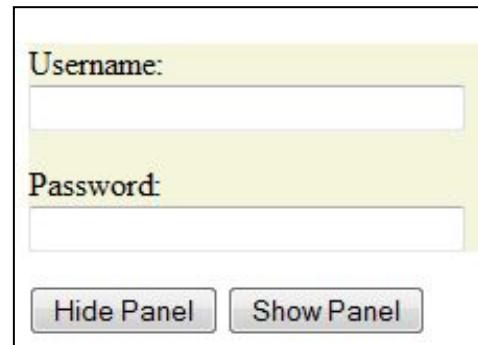
Página aspx

```
<form id="form1" runat="server">
<asp:Panel id="myPanel" BackColor="Beige" Width="220" runat="server">
<p>Username: <asp:TextBox id="usernameTextBox" Columns="30" runat="server" /></p>
<p>Password: <asp:TextBox id="TextBox1" TextMode="Password" Columns="30" runat="server" /></p>
</asp:Panel>
<asp:Button id="hideButton" Text="Hide Panel" OnClick="HidePanel" runat="server" />
<asp:Button id="showButton" Text="Show Panel" OnClick="ShowPanel" runat="server" />
</form>
```

Página aspx.cs

```
protected void HidePanel(object sender, EventArgs e)
{ myPanel.Visible = false; }

protected void ShowPanel(object sender, EventArgs e)
{ myPanel.Visible = true; }
```



Páginas maestras

4

Página maestra

- Una buena manera de desarrollar nuestras aplicaciones es implementar una **página maestra de la que se basen el resto de páginas web de nuestra aplicación**
- Las páginas maestras permiten crear un diseño coherente para las páginas de la aplicación
- Se puede definir el aspecto, el diseño y el comportamiento estándar que desea que tengan todas las páginas (o un grupo de páginas) de la aplicación en una sola página maestra
- **A continuación, se crean páginas de contenido individuales** que incluyan el contenido que desea mostrar
- Cuando los usuarios solicitan las páginas de contenido, **éstas son combinadas con la página maestra** con el fin de generar una salida que combine el diseño de la página maestra con el de la página de contenido

Principales ventajas

- Realización de cambios de diseño en una sola ubicación; los cambios se verán reflejados en todas las páginas que usan la página maestra.
- Reutilización de la interfaz de usuario
- Mejor experiencia del usuario final: páginas más coherentes

Cómo funciona

- Una página maestra es un archivo ASP.NET con extensión *.master*, que tiene un diseño predefinido, que puede incluir texto estático, elementos HTML y controles de servidor
- La página maestra se identifica mediante la directiva ***@ Master***, que reemplaza la directiva ***@ Page*** que se usa en las páginas *.aspx* ordinarias
- Además del texto estático y los controles que aparecerán en todas las páginas, la página maestra también incluye uno o varios controles ***ContentPlaceHolder***. Estos controles ***PlaceHolder*** definen las áreas que incluirán contenido reemplazable. A su vez, el contenido reemplazable se define en las páginas de contenido.

@Page directiva / @Master directiva

El `<%@ Page %>` directiva especifica los atributos específicos de página utilizados por el motor ASP.NET al analizar y compilar la página. Esto incluye su archivo de página maestra, la ubicación de su archivo de código y su título, entre otros datos.

```
<%@ Page Language="C#" MasterPageFile("~/Site.master" AutoEventWireup="true"  
CodeFile="Default.aspx.cs" Inherits="_Default" Title="Untitled Page" %>
```

```
<% @ Master Language="C#" CodeFile="MasterPageSample.master.cs" Inherits="MasterPageSample" %>
```

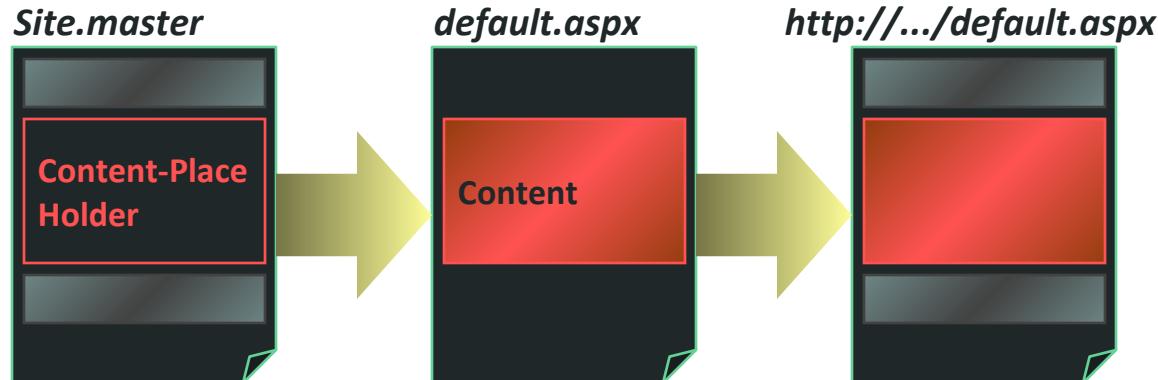
@Page directive / @Master directive



```
AltaUsuario.aspx  ✎ X  WebHada
1  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AltaUsuario.aspx.cs" Inherits="WebHada.AltaUsuario" %>
2
3  <!DOCTYPE html>
4
5  <html xmlns="http://www.w3.org/1999/xhtml">
6  <head runat="server">
7    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8    <title></title>
9  </head>
10 <body>
11   <form id="form1" runat="server">
12     <div>
13       </div>
14   </form>
15 </body>
16 </html>
17
```

Cómo funciona...

- Master pages definen el contenido común y los contenedores de contenido (content placeholders)
- Content pages hacen referencia a las páginas maestras y llenan a los contenedores con su contenido.



En tiempo de ejecución

- IIS controla las páginas maestras en la secuencia siguiente:
 - Los usuarios solicitan una página escribiendo la dirección URL de la página de contenido.
 - Cuando se captura la página, se lee la **directiva @ Page**. Si la directiva hace referencia a una página maestra, también se lee la página maestra. Si las páginas se solicitan por primera vez, se compilan las dos páginas.
 - La página maestra con el contenido actualizado se combina en el árbol de control de la página de contenido.
 - El contenido de los controles Content individuales se combina en el control **contentplaceholder** correspondiente de la página maestra.
 - La página combinada resultante se representa en el explorador.

Page.Master

- **Nueva propiedad (Master) de System.Web.UI.Page**
- Esta propiedad contiene una referencia a la página maestra de la página de contenido, por tanto provee a una página contenido de acceso programático a la página maestra
 - Determina si la pagina tiene asociada una maestra
 - Acceso a los controles definidos en la maestra pudiendo escribir código soporte en las páginas de contenido
 - Acceso a métodos públicos y propiedades definidas en la maestra
- Integración a nivel código de las páginas maestras y contenidos

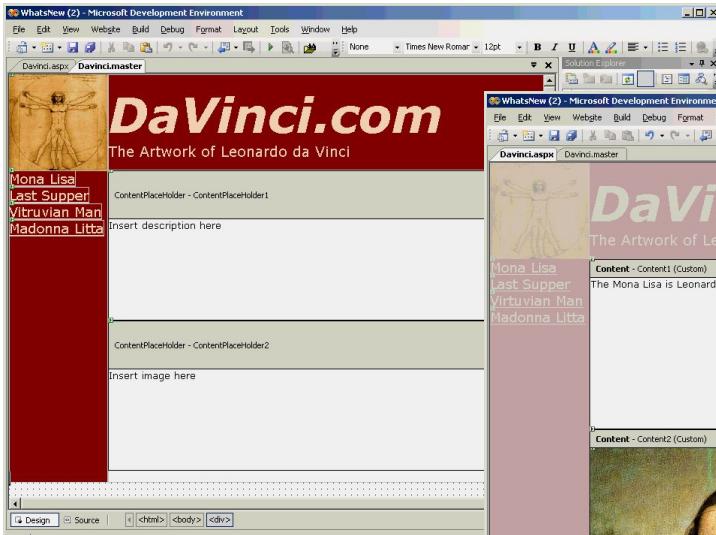
Desde el navegador..

- Desde la perspectiva del usuario, la combinación de las páginas maestras y las páginas de contenido da como resultado una única página. La dirección URL de esta página es la de la página de contenido.

En Visual Studio...

- Herencia visual

Master Page



Content Page



Aplicación web (Página maestra)

- Vamos a modificar nuestra aplicación web, para que se base en una página maestra.
- Modificaremos la página maestra **Site.Master**, que viene por defecto.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="WebHada.SiteMaster" %>

!DOCTYPE html

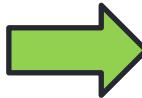
<html lang="en">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>%: Page.Title %% - Mi aplicación ASP.NET</title>
    <asp:PlaceHolder runat="server">
        <%%: Scripts.Render("~/bundles/modernizr") %%>
    </asp:PlaceHolder>
    <webp:bundlereference runat="server" path "~/Content/css" />
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
</head>
<body>
    <form runat="server">

        <div class="container">
            <h2>MI PRIMERA APPLICACIÓN WEB ASP.NET </h2>

            <div class="collapse navbar-collapse">
                <ul class="nav navbar-nav">
                    <li><a runat="server" href="/">Inicio</a></li>
                    <li><a runat="server" href="/AltaUsuarioHija">Alta de usuario</a></li>
                    <li><a runat="server" href="/Pedidos">Pedidos</a></li>
                    <li><a runat="server" href="/About">Acerca de</a></li>
                    <li><a runat="server" href="/Contact">Contacto</a></li>
                </ul>
            </div>
        </div>

        <div class="container body-content">
            <hr />
            <asp:ContentPlaceHolder ID="MainContent" runat="server"></asp:ContentPlaceHolder>
            <hr />

            <footer>
                <p>&copy; <%%: DateTime.Now.Year %%> - Mi primera aplicación ASP.NET</p>
            </footer>
        </div>
    </form>
</body>
</html>
```



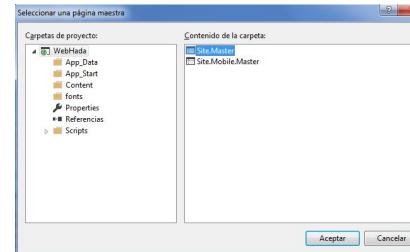
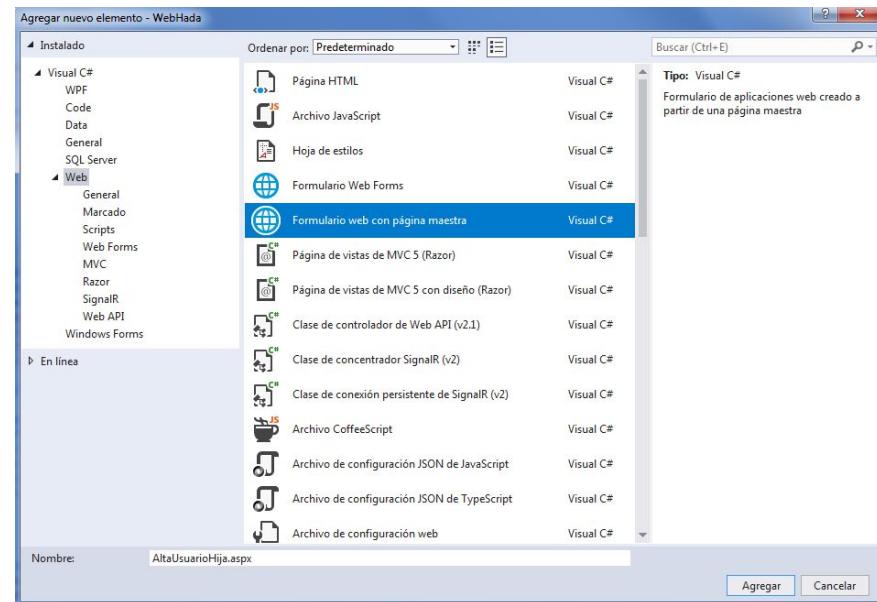
```
<%@ Master Language="C#"
    AutoEventWireup="true"
    CodeBehind="Site.master.cs"
    Inherits="WebHada.SiteMaster" %>
```

```
<asp:ContentPlaceHolder ID="MainContent"
    runat="server">
</asp:ContentPlaceHolder>
```

Aplicación web (Página maestra)

Creación de formulario web con página maestra (página hija)

- Creamos un nuevo formulario con página maestra, hay que acceder al menú *Agregar -> Nuevo elemento -> Formulario web con página maestra*.
- Le damos un nombre al formulario, en este caso *AltaUsuarioHija.aspx*
- Tenemos que elegir a partir de que página maestra lo creamos



Aplicación web (Página maestra)

Contenido del nuevo formulario web con página maestra

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"  
CodeBehind="AltaUsuarioHija.aspx.cs" Inherits="WebHada.AltaUsuarioHija" %>  
  
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">  
</asp:Content>
```

MI PRIMERA APPLICACIÓN WEB ASP.NET

[Inicio](#) [Alta de usuario](#) [Pedidos](#) [Acerca de](#) [Contacto](#)

© 2018 - Mi primera aplicación ASP.NET

Aplicación web (Página maestra)

Contenido del nuevo formulario web con página maestra

- El nuevo formulario *AltaUsuarioHija* ha incorporado todo lo necesario para invocar a la página maestra. Automáticamente la nueva página incorpora la directiva *MasterPageFile* e introducirá los *Content* definidos en la página maestra utilizando *ContentPlaceHolderID*.
- A las páginas hijas se ha añadido los siguientes elementos:
 - En la primera línea, el atributo *MasterPageFile* en el Tag Page.
MasterPageFile="~/Site.Master"
 - Hacer referencia al contenido mediante *Content*. La página hija, completa el contenido definido de la página maestra.

```
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" Runat="Server">
```

Aplicación web (Página maestra)

Modificamos el nuevo formulario web

- Insertamos el código que modifique el contenido, entre `<asp:Content>` y `</asp:Content>`
- Copiamos el código que teníamos de la página `AltaUsuario.aspx`, y lo incrustamos
- De esta manera, cuando se llame a la nueva página, se cargará la página maestra y a continuación incorporará los contenidos

The screenshot shows the Visual Studio IDE with two windows open. The top window is the code editor for the file `AltaUsuarioHija.aspx.cs`. It displays the following ASP.NET code:

```
1 <%@ Page Title="" Language="C#" MasterPageFile "~/Site.Master" AutoEventWireup="true" CodeBehind="AltaUsuarioHija.aspx.cs" %>
2 
3 <asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
4     <div style="height: 318px">
5         Nombre &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
6         <asp:TextBox ID="TBNombre" runat="server"></asp:TextBox>
7         <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TBNombre" ErrorMessage="El campo 'Nombre' es obligatorio."><br />
8     <br />
9         Apellidos &nbsp;&nbsp;&nbsp;
10        <asp:TextBox ID="TBApellidos" runat="server"></asp:TextBox>
11        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="TBApellidos" ErrorMessage="El campo 'Apellidos' es obligatorio."><br />
12        <br />
13        Sexo &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
14        <asp:RadioButton ID="RBHombrer" runat="server" Checked="True" Text="Hombre" GroupName="RBGrupo" />
15        <asp:RadioButton ID="RBmujer" runat="server" Text="Mujer" GroupName="GRGrupo" />
16        <br />
17        Dirección &nbsp;&nbsp;&nbsp; <asp:TextBox ID="TBDirreccion" runat="server" Width="253px"></asp:TextBox>
18        <br />
19        Cód. Postal <asp:TextBox ID="TBCodPostal" runat="server"></asp:TextBox>
20        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="TBCodPostal" Dis-
21        <br />
22        Correo <asp:TextBox ID="TBcorreo" runat="server"></asp:TextBox>
23        <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="TBcorreo" ErrorMessage="El campo 'Correo' es obligatorio."><br />
24    </div>
25 
```

The bottom window shows the browser preview titled "MI PRIMERA APPLICACIÓN WEB ASP.NET". It displays a form with five fields: Nombre, Apellidos, Sexo (radio buttons for Hombre and Mujer), Dirección, and Cód. Postal. Each field has a red validation message indicating it is required:

- Nombre: El campo 'Nombre' es obligatorio.
- Apellidos: El campo 'Apellidos' es obligatorio.
- Sexo: Hombre (radio button selected).
- Dirección: (empty input field).
- Cód. Postal: El código postal tiene que tener 5 números.
- Correo: El campo 'Correo' es obligatorio.

Aplicación web (Página maestra)

Ejecución

MI PRIMERA APPLICACIÓN WEB ASP.NET

[Inicio](#) [Alta de usuario](#) [Pedidos](#) [Acerca de](#) [Contacto](#)

Nombre

Apellidos

Sexo Hombre Mujer

Dirección

Cód. Postal

Correo

© 2018 - Mi primera aplicación ASP.NET

Eventos de los controles del servidor

5

Modelo de eventos

- En las páginas Web ASP.NET, los eventos asociados a los controles de servidor se originan en el cliente (explorador) pero los controla la página ASP.NET en el servidor Web.
- La información del evento se captura en el cliente y se transmite un mensaje de evento al servidor mediante un envío HTTP.
- La página debe interpretar el envío para determinar el evento ocurrido y, a continuación, llamar al método apropiado del código del servidor para controlar dicho evento.

Enlazar eventos a métodos

- Un evento es un mensaje que envía un objeto cuando ocurre una acción (Ej. "se ha hecho clic en un botón") . La acción puede estar causada por la interacción del usuario, como un clic, o por otra lógica del programa.
- En una aplicación, se debe traducir el mensaje en una llamada a un método del código.
- El enlace entre el mensaje del evento y un método específico (es decir, un controlador de evento) se lleva a cabo utilizando **un delegado de eventos**.

Eventos y delegados

- El objeto que provoca el evento se conoce como remitente del evento. El objeto que captura el evento y responde a él se denomina receptor del evento.
- En las comunicaciones de eventos, el remitente del evento no sabe qué objeto o método recibirá los eventos que provoca. **Se necesita un intermediario (o mecanismo de tipo puntero) entre el origen y el receptor.**
- .NET Framework define un tipo especial (**Delegate**) que proporciona la funcionalidad de un **puntero a función**.

Manejadores de eventos

- El prototipo de los métodos manejadores de eventos deben coincidir con el prototipo del *delegado EventHandler*.

```
delegate void EventHandler(object sender, EventArgs e);
```

- Por tanto, los manejadores de evento en ASPnet devuelven **void** y tienen **dos** parámetros:
 - **Objeto que lanza el evento:** objeto (*Object sender*)
 - **Argumentos del evento:** información específica del evento (*EventArgs* o tipo derivado)
- Por ejemplo, para un control ImageButton de servidor Web, el segundo argumento es de tipo ImageClickEventArgs, que incluye información sobre las coordenadas donde el usuario ha hecho clic..

Asociar el manejador al control

- Escribirlo manualmente

Archivo Default.aspx

```
<asp:Button ID="Button1" runat="server" onclick="EventoClick"  
Text="Button" />
```

Archivo Default.aspx.cs

```
protected void EventoClick(object sender, EventArgs e)  
{ }
```

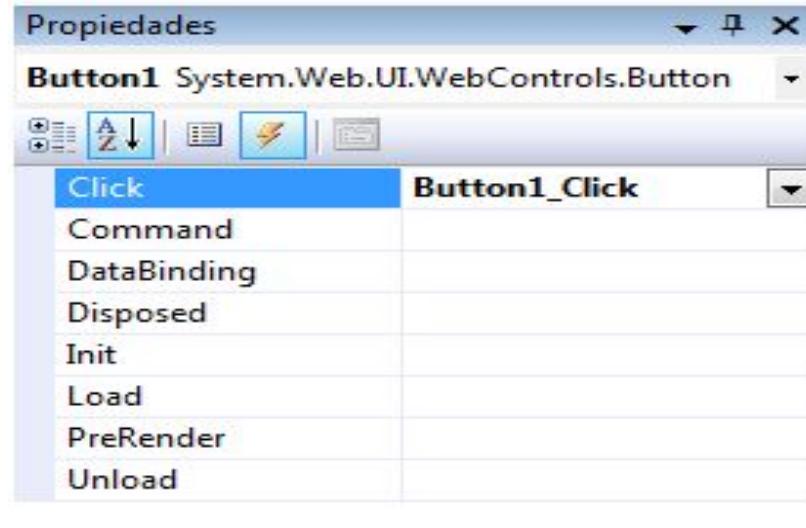
- Pulsar doble click sobre un control en vista diseño (evento por defecto)

```
protected void Button1_Click(object sender, EventArgs e)
```

Al compilar la página, ASP.NET busca un método denominado *EventoClick* y confirma que éste tiene la firma adecuada (acepta dos argumentos, uno de tipo **Object** y otro de tipo **EventArgs**). A continuación, ASP.NET enlaza automáticamente el evento al método

Asociar el manejador al control

- Seleccionar el evento de la ventana de propiedades del control (y asociar un nombre de método manejador del evento o doble click)



Eventos postback vs no-postback

- Una página se carga después de cada petición: fenómeno conocido como **PostBack (=envío)**.
- Eventos **Postback**:
 - causan que la información del formulario se envíe al servidor inmediatamente.
- Eventos **no-Postback (o cached)**:
 - la información se envía en el siguiente evento **postback**.
 - Los eventos se guardan en una cola en el cliente hasta que un evento **postback** ocurre.

Eventos postback vs no-postback

- Button, Link Button y Image Button causan eventos **postback**.
- TextBox, DropDownList, ListBox, RadioButton y CheckBox, proveen eventos **cached**.
 - Sin embargo podemos sobrecargar este comportamiento en los controles para poder realizar eventos postback, cambiando la propiedad **AutoPostBack** a true.

Eventos de página

- Las páginas ASP.NET provocan eventos de ciclos de vida como **Init**, **Load**, **PreRender** y otros.
- De manera predeterminada, los eventos de página se pueden enlazar a los métodos utilizando la convención de nomenclatura **Page_nombreDeEvento**.
 - Por ejemplo, con el fin de crear un controlador para el evento **Load** de la página, se puede crear un método denominado **Page_Load**.
 - En tiempo de compilación, ASP.NET buscará los métodos que se basen en esta convención de nomenclatura y realizará el enlace automáticamente entre el evento y el método.
- Se puede utilizar la convención **Page_nombreDeEvento** para cualquier evento expuesto por la clase **Page**.

Propiedad IsPostBack

- Una página se carga después de cada petición: fenómeno conocido como **PostBack** (=envío)
- El evento **Page_Load** se produce en cada petición
- **Page.IsPostBack** para ejecutar código condicional

```
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack) {
        Response.Write("<br>Page has been posted back.");
    }
}
```

Importante: Los métodos de control de eventos de página no requieren ningún argumento. Estos eventos (como Load) pueden aceptar los dos argumentos estándar, pero en estos argumentos no se pasa ningún valor.

Navegación entre formularios

6

Navegación tradicional

- En HTML:
 - Elemento `<a>` y atributo `href`

```
<a href="Login.aspx">Regístrate aquí </a>
```
 - La página `Login.aspx` debe estar en el mismo directorio que la página que contiene el enlace
 - Despues de pulsar sobre el enlace se muestra la página `Login.aspx`
- En ASPx:

```
<asp:HyperLink ID="HyperLink1" runat="server"  
NavigateUrl("~/Login.aspx")>Regístrate aquí</asp:HyperLink>
```

Diferentes formas de navegar

- **Control hipervínculo**
 - Navega a otra página
- Método **Response.Redirect**
 - Navega a otra página por medio del código. (Equivalente a navegar a través de un enlace)

Hipervínculos y Redirección

- Los hipervínculos responden a eventos click mostrando la página especificada en la propiedad *NavigateURL* del control.
- Si quieres capturar un click en el código, debes usar los eventos de un *LinkButton* o *ImageButton* y utilizar

```
Response.Redirect("SiguientePagina.aspx");
```
- También podemos pasar parámetros a la nueva página.

Paso de parámetros a otra página

- Paso de parámetros en la URL.
 - Almacena los datos en la colección QueryString
 - Múltiples parámetros separados por &

```
int valor = 22;  
int valor1 = 25;  
Response.Redirect("Default4.aspx?par1=" + valor);  
Response.Redirect("Default4.aspx?par1=" + valor + "&par2=" + valor1);
```

<http://localhost:49999/Default4.aspx?par1=22&par2=25>

Paso de parámetros a otra página

- Limitaciones con QueryString
 - Los caracteres utilizados deben ser caracteres permitidos en una URL
 - La información es visible a los usuarios en la barra del navegador
 - Los usuarios pueden modificar la información provocando errores inesperados
 - Muchos navegadores imponen un límite en la longitud de la URL

Paso de parámetros a otra página

- Caracteres especiales:
 - & (para separar múltiple query strings)
 - + (alternativa para representar un espacio)
 - # (especifica un marcador en una página web)
- Solución:
 - Utilizar los métodos de la clase *HttpServerUtility* para codificar los datos

```
Response.Redirect("WebForm2.aspx?par1="+ Server.UrlEncode(" &hola "));
```

Paso de parámetros a otra página

- El método *UrlEncode* reemplaza los caracteres especiales por secuencias de escape

`http://localhost:49999/WebForm2.aspx?par1=+%26hola+`

- La recuperación de datos codificados mediante el método *UrlEncode* con *QueryString* es automática en ASP.NET (no es necesario utilizar un método que decodifique los datos)

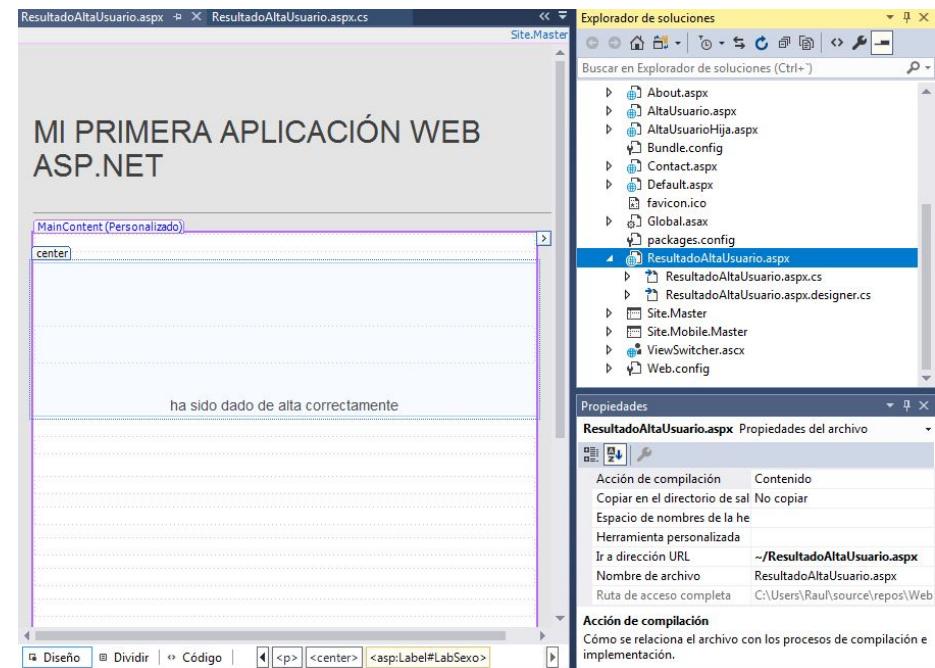
Objeto Request

- Proporciona información sobre la petición HTTP del cliente que ha provocado la carga de la página actual.
 - Información sobre el cliente (`Request.Browser`,
`Request.Browser.IsMobileDevice`, `Request.Browser.Id`)
 - Cookies (`Request.Browser.Cookies`)
 - Parámetros pasados a la página:

```
if (Request.QueryString["par1"] != "")  
    Label1.Text = Request.QueryString["par1"];
```

Aplicación web (Navegación)

- Vamos a modificar nuestra aplicación web, para que una vez introducidos los datos, y pulsemos en el botón *Enviar datos*, nos redirija a otra página de resultado al que le pasaremos el nombre, los apellidos y el sexo como parámetros.
- Creamos otro formulario web a partir de la página maestra *Site.master* con el nombre *ResultadoAltaUsuario.aspx*. Esta página recibirá los parámetros pasados por la página *AltaUsuarioHija.aspx*
- La página *AltaUsuarioHija.aspx*, mostrará según el sexo del usuario, la palabra “La usuaria” o “El usuario”.



Aplicación web (Navegación)

- Modificamos la página *AltaUsuarioHija.aspx*, para incorporar el evento *BtnEnviar_Click*, asociado al botón *Enviar datos*.

```
protected void BtnEnviar_Click(object sender, EventArgs e)
{
    String s;
    if(RBHomme.Checked)
        s=RBHomme.Text;
    else
        s=RBMujer.Text;
    Response.Redirect("ResultadoAltaUsuario.aspx?nombre="+ TBNombre.Text + "&apellidos=" + TBApellidos.Text &sexo="+s);
}
```

- Modificamos la página *ResultadoAltaUsuarioHija.aspx*, para incorporar en el evento *Page_Load*, el código para cargar los datos recibidos en una etiqueta.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.QueryString["sexo"] == "Hombre")
        LabSexo.Text = "El usuario";
    else
        LabSexo.Text = "La usuaria";

    LabNombreUsuario.Text = Request.QueryString["nombre"] + ' ' + Request.QueryString["apellidos"];
}
```

Aplicación web (Navegación)

Ejecución

MI PRIMERA APPLICACIÓN WEB ASP.NET

Inicio Alta de usuario Pedidos Acerca de Contacto

Nombre

Apellidos

Sexo Hombre Mujer

Dirección

Cód. Postal

Correo

© 2018 - Mi primera aplicación ASP.NET



MI PRIMERA APPLICACIÓN WEB ASP.NET

Inicio Alta de usuario Pedidos Acerca de Contacto

La usuaria
Sandra López

ha sido dado de alta correctamente

© 2018 - Mi primera aplicación ASP.NET

CSS

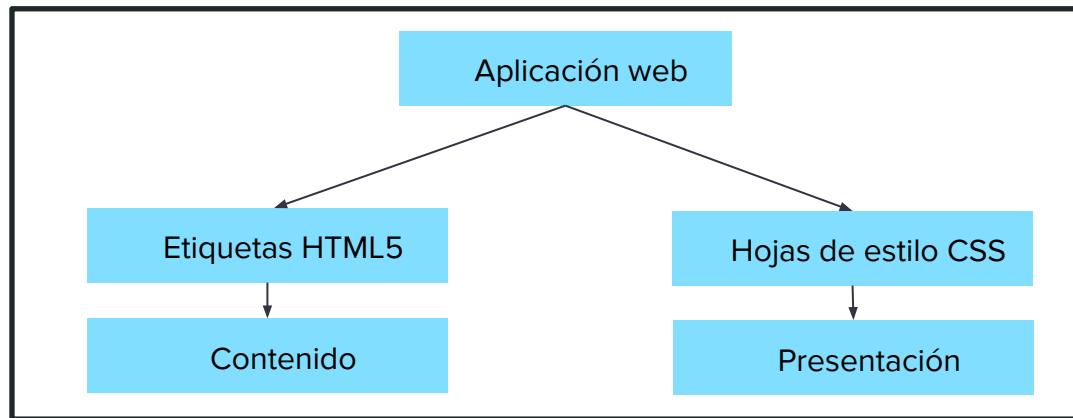
- HTML
 - ASP .NET
 - **CSS**
-

CSS

- CSS es una herramienta que permite definir la presentación de un documento
- Permite crear un conjunto de estilos en una única localización
- Las páginas a las que se aplica la misma hoja de estilos tendrán las mismas fuentes, colores y tamaño
- Proporcionan una estética homogénea en todas las páginas de un sitio web
- Las hojas de estilo son elementos agregados al lenguaje HTML que tomarán en cuenta la presentación del documento o de la aplicación web
- El mismo contenido podrá visualizarse según la hoja de estilos adoptada, de forma diferente en distintos dispositivos (PC, Tablet, móvil, etc.)

CSS

- Separación entre contenido y presentación



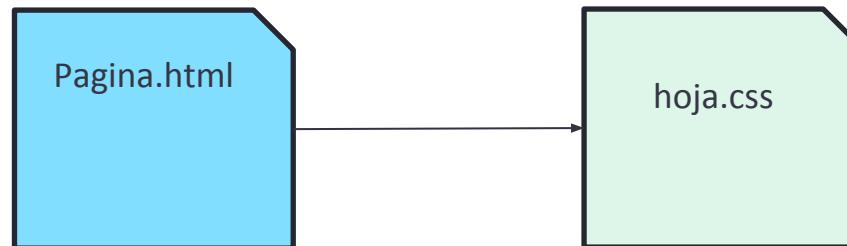
- Podemos incorporar el estilo de tres formas:
 - Una hoja de estilos externa, en un archivo distinto con extensión .css
 - Una hoja de estilos interna incrustada en el propio documento HTML
 - Una hoja de estilos en línea para aplicar a un elemento determinado

Tipos de hojas de estilo

Hoja de estilo externa

- Se emplazan todas las reglas de estilo en una única hoja de estilos externa
- Se enlaza esta hoja de estilos externa con todas las páginas que vayan a tener la misma apariencia
- Para hacer una referencia a una hoja de estilos externa en una web, hay que introducir en el elemento **HEAD**, la siguiente sentencia:

```
<link rel="stylesheet" type="text/css" href="~/hoja.css" />
```

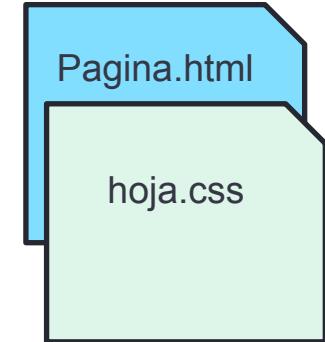


Tipos de hojas de estilo (2)

Hoja de estilo interna

- Hoja de estilo incrustada dentro de un documento.
- Se utiliza la etiqueta `<style>`, dentro del elemento **HEAD**
- Se debe reescribir en cada página (lo cual es un problema)
- Se definen las reglas de estilo entre las etiquetas:

```
<style type="text/css">
    a { background-color: #ff9;
        color: #00f; }
</style>
```



Tipos de hojas de estilo (3)

Hoja de estilo en línea

- Permiten asignar un estilo a un elemento determinado
- Para ello hay que utilizar el atributo style dentro del elemento al que se quiere aplicar el estilo

```
<a href="/" style="background-color: #ff9;  
color: #00f;"> Home</a>
```

```
<h1 style="color:blue; margin-left:60 px;"> Estilo  
en línea </h1>
```

Selector de estilos

- En caso de utilizar hojas externas o internas cada regla de estilo tiene un selector
- Un selector es el tipo de elemento al que se va a aplicar el estilo
- Ejemplo:

```
a {  
    background-color: #ff9;  
    color: #00f;  
}
```

- En ASP.NET hay dos tipos de selectores:
 - Tipos de elementos y clases

Selector de tipo elemento

- El estilo se aplica a todos los elementos del mismo tipo

```
h2 { color: #369;}
```



Cambia de color todas las cabeceras de segundo nivel

```
table {text-align: center;}
```



Alinear el texto de las celdas de todas las tablas

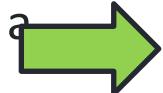
```
p{ font-size: 150%;}
```



Cambia el tamaño de la fuente para todo el párrafo

Selector de tipo clase

- Se utiliza cuando asignamos a cada elemento una clase determinada
- La página HTML indicará la referencia a la clase definida en el estilo



```
<p class="pageinfo">  
    Copyright 2019  
</p>
```

- La hoja de estilos contendrá para cada clase una serie de características
- Los selectores de clase van precedidos por un .



```
.pageinfo  
{  
    font-family: Arial;  
    font-size: x-small;  
}
```

Selector de tipo clase (2)

- Otra forma de definir una clase:

```
h1.textorojo  
{  
    color: red;  
}
```

- La invocación desde HTML será

```
<h1 class="textorojo"> Texto en rojo</h1>  
<h1> Texto por defecto</h1>
```



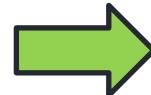
```
Texto en rojo  
Texto por defecto
```

Selector de identificador

- El selector id, permite aplicar una hoja de estilo, aunque sólo se podrá invocar una vez en el documento
- Identifica un elemento único en la página
- La invocación desde HTML será

```
<p id="textoazul"> Texto azul </p>
<div id="fondorojo"> Caja con fondo rojo </div>
```

```
#textoazul {color: blue;}
#fondorojo {
    background-color: red;
}
```



Texto azul
Caja con fondo rojo

Ejemplo de hoja de estilos css

```
html,body {  
    height:100%;  
}  
  
h1 {text-align: center;}  
  
header {  
    display:block;  
    background:#086af0;  
    padding:6px 10px;  
    color:white;  
}  
  
section {  
    width: 79%;  
    background: #ccc;  
    float: left;  
    overflow: auto;  
    padding-bottom: 60px;  
    padding-top:30px;  
}  
  
aside {  
    float: right;  
    border: 1px solid red;  
    width: 19%;  
    border: 1px solid red;  
}
```

```
footer {  
    position: relative;  
    margin-top: -50px;  
    height: 40px;  
    padding:5px 0px;  
    clear: both;  
    background: #286af0;  
    text-align: center;  
    color: white;  
}  
  
figure {  
    display: table; margin: 0 auto;  
}  
  
nav {  
    position : absolute;  
    width : 100%;  
    height: 50px;  
    background-color: #333;  
    overflow: hidden;  
    color: white;  
}  
  
nav ul {  
    margin : 0 auto;  
    width : 940px;  
    list-style : none;  
}
```

```
nav ul li {  
    float : left;  
}  
  
nav ul li a {  
    display : block;  
    margin-right : 20px;  
    width : 140px;  
    font-size : 18px;  
    line-height : 44px;  
    text-align : center;  
    text-decoration : none;  
    color : white;  
}  
  
nav ul li a:hover {  
    color : #fff;  
}  
nav ul li.selected a {  
    color : #fff;  
}
```

Propiedades de estilo

- Font: Tipo de fuente, tamaño, color, negrita...
- Background: color de fondo, imagen de fondo...
- Block: espacio entre párrafos, líneas, palabras...
- Box: personalizar tablas, colores, bordes...
- Border: dibuja bordes alrededor de diferentes elementos
- ...

CssClass

- Asociar selector de tipo clase en Formularios Web en ASP.NET:

```
<head runat="server">  
<title>Probando CSS</title>  
<link rel="stylesheet" type="text/css" href="hoja.css" />  
</head>
```

hoja.css

```
.textbox  
{ font-family: Arial;  
background-color: #0099FF;  
border: 1px solid  
}
```

```
<asp:TextBox ID="TextBox1" CssClass="textbox" runat="server" />
```

Maquetación con CSS

- ***maquetar una pagina web es pasar el diseño a código HTML***, poniendo cada cosa en su lugar (una cabecera, un menu, etc.).
- Hasta hace unos años la única manera de maquetar una pagina web era mediante **tablas HTML** (<table>), pero esto tiene muchas desventajas y limitaciones
 - el uso de las tablas está condicionado a la mera tabulación de datos,
 - Un diseño con tablas no es flexible, es decir, que no podemos cambiar la distribución de los elementos en la página, a menos que la volvamos a hacer.
 - Cada Explorador renderiza de manera distinta cada documento HTML y con estructuras con tablas el cambio es más notorio
 - Ocupa más espacio y más ancho de banda.
 - Google no indexa de igual manera las páginas con estructuras basadas en tablas.
- Por eso la técnica de maquetación fue evolucionando con los años hasta llegar al punto donde no se usan tablas, si no capas (**los famosos DIVs**) a **las que se le dan formato mediante CSS**.

Divs

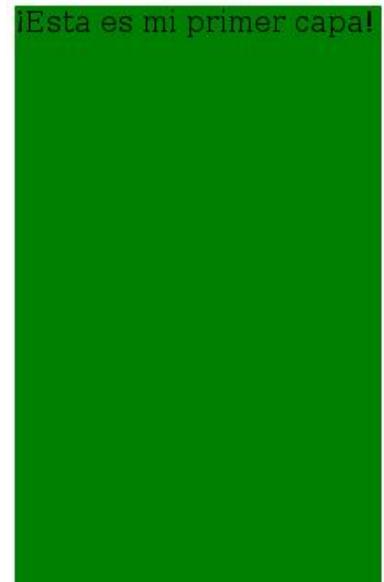
- Las capas, layouts o divs son la misma cosa con distinto nombre.
- Para tener un concepto mental de lo que son, podemos imaginarlos como ***contenedores o bloques donde podemos meter lo que queramos dentro (imágenes, texto, animaciones, otro bloque, o todo al mismo tiempo) a los que se le asigna un ancho, alto y posición***, de esta manera se van a ir posicionando consiguiendo la estructura que queremos.

Formato a un DIV

- Para darle formato a un DIV tenemos que identificarlo de alguna forma, para esto existe **el atributo ID**, en el pondremos el nombre del DIV para luego llamarlo desde la hoja de estilos, la forma de escribirlo es así:

```
<div id="capa1">¡Esta es mi primer capa!</div>
```

```
#capa1{  
    width:210px;  
    height:300px;  
    background-color:green;  
}
```



¡Esta es mi primer capa!

Class o id?

- La diferencia entre una clase (.) y un bloque (#) es que
 - El bloque es único e irrepetible en la pagina, es decir, si creamos un estilo de bloque "**#busqueda**" para mostrar el cuadro de búsqueda no podremos usarlo otra vez en la misma pagina,
 - En cambio si a "**#busqueda**" lo convertimos en una clase ".**busqueda**" podremos usarlo cuantas veces queramos.

Propiedades de maquetación

```
<div style = "display: table; margin-left: auto; margin-right: auto; width: 500px;">
    <div style = "height: 45px; width: 500px;"></div>
    <div style = "float: left; height: 75px; width: 150px;"></div>
    <div style = "float: right; height: 75px; width: 350px;"></div>
    <div style = "clear: both; height: 35px; width: 500px;"></div>
</div>
```

- Los elementos DIV pueden centrarse utilizando los atributos **margin-left: auto; margin-right: auto;**
- La propiedad **float** ajusta los elementos hacia el margen indicado (cuando tenemos capas adyacentes).
- A veces necesitarás tener una capa o bloque que no tenga capas a su/s lados, para eso esta la propiedad CSS **Clear**. (ej el pie de página).
Esta propiedad se utiliza en conjunto con *float* y sirve para evitar que una capa se posicione a **cualquiera de los lados**

Cuidado

- En las estructuras clásicas con tablas, podíamos utilizar tamaños en porcentaje. Aunque aquí, también podemos utilizar porcentajes, el dibujado de la página puede no verse como se esperaba.

Maquetar con CSS links

- <https://www.youtube.com/watch?v=sOyA84uh0CQ>
- <http://www.1keydata.com/css-tutorial/>
- https://www.youtube.com/watch?v=nTOAXkMbd_0
- <http://www.tutorialmonsters.com/creando-una-web-ejemplo-con-diy-y-css-primera-parte/>

Hada T5: Modelo de capas

Objetivos

- Conocer la **arquitectura de 3 capas** para el acceso a DDBB
- Comprender el concepto de **entidad de negocio** y cómo representarla.
- Comprender el concepto de **componente de acceso a datos** y cómo representarlo.

Arquitectura de capas

- **Arquitectura de capas**
 - Capa de presentación
 - Entidades de Negocio
 - Componentes de Acceso a Datos
 - De relacional a EN
 - Ejercicio
-

¿Por qué una arquitectura de capas?

- **Independencia, flexibilidad:** cada una de esas capas puede ser sustituida en cualquier momento sin afectar a las otras
 - Ej. diferentes presentaciones para la misma lógica de negocio
- Permite una mejor distribución del trabajo en un equipo de desarrolladores (diseñadores, programadores, etc.)

Arquitectura de Capas

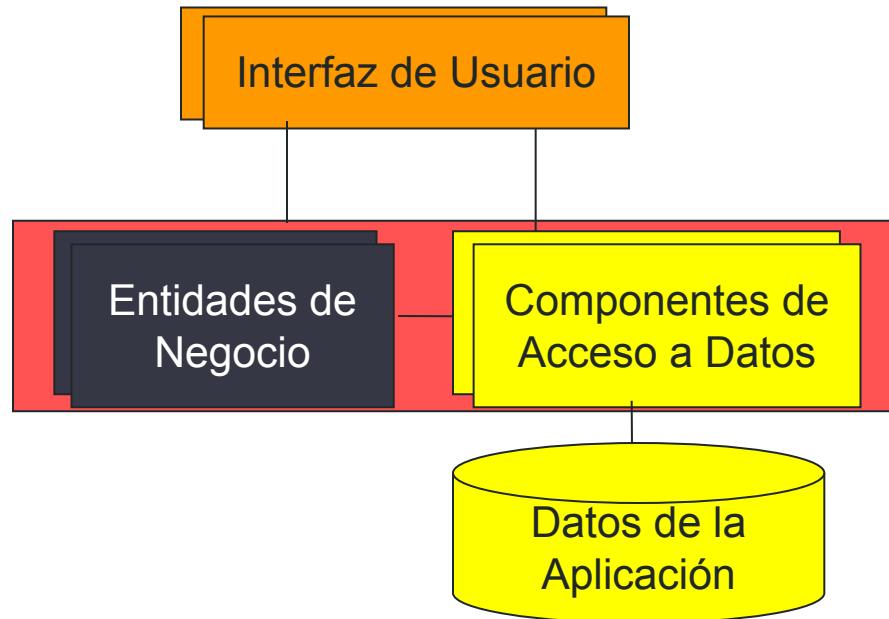
- Patrón de arquitectura [Buschmann] que establece la distribución de una aplicación en divisiones lógicas desarrolladas y mantenidas como módulos independientes, incluso en plataformas distintas.
- Una arquitectura de 3 capas esta divida en:
 - **Interfaz de usuario**, componentes que interactúan con el usuario final
 - **Lógica de negocio**, contienen las reglas de negocio de nuestra aplicación
 - **Persistencia**, contiene el acceso y almacenamiento de los datos

Arquitectura de una Aplicación

3 Capas Lógicas



Configuración de Componentes



Capa de Presentación

- Arquitectura de capas
 - **Capa de presentación**
 - Entidades de Negocio
 - Componentes de Acceso a Datos
 - De relacional a EN
 - Ejercicio
-

Capa de Presentación

- La **capa de presentación** maneja la entrada y salida básica del usuario. Es responsable de proporcionar la interfaz gráfica de usuario, registrar las teclas y rastrear el movimiento del ratón del usuario.
- En un restaurante de lujo, el camarero maneja esta "capa de presentación" al proporcionarle al cliente un menú --la interfaz-- y *anota la orden del cliente*.

Entidades de Negocio

- Arquitectura de capas
 - Capa de presentación
 - **Entidades de Negocio**
 - Componentes de Acceso a Datos
 - De relacional a EN
 - Ejercicio
-

Entidades de Negocio (EN)

- Componentes que representan entidades de negocio del mundo real, p.e. un producto, un pedido
- Contienen normalmente la información de una clase de dominio con sus atributos, operaciones y restricciones. Aunque pueden representar una composición de clases
- Tienen **asociado un CAD** (Componente de Acceso a Datos) que le proporciona el acceso y el mapeo a los datos
- Pueden ser representados de múltiples maneras, p.e. clases personalizadas, DataSets, XML, etc.

Representación de una EN

```
public class ENProducto
{
    // Campos privados para mantener el
    // estado de la Entidad Producto
    private int idProducto;
    private String nombre;
    private String cantidadPorUnidad;
    private decimal precioUnitario;
    private int unidadesStock;
    private int stockMinimo;
    // Propiedades públicas para exponer el
    // estado del producto
    public int IdProducto
    {
        get { return idProducto; }
        set { idProducto = value; }
    }
```

```
public String Nombre {
    get { return nombre; }
    set { nombre = value; }
}
public String CantidadPorUnidad
{
    get { return cantidadPorUnidad; }
    set { cantidadPorUnidad = value; }
}
public decimal PrecioUnitario
{
    get { return precioUnitario; }
    set { precioUnitario = value; }}
...
```

Representación de una EN (II)

```
// Métodos que realizan algún procesamiento
public void IncrementarPrecioUnidadPor (decimal
cantidad)
{
    precioUnitario += cantidad;
}

public short UnidadesSobreElNivelMinimo
{
    get { return (short)(unidadesStock - stockMinimo);
}
}
}//Fin de Clase
```

Capa de Persistencia

- Arquitectura de capas
 - Capa de presentación
 - Entidades de Negocio
 - **Componentes de Acceso a Datos**
 - De relacional a EN
 - Ejercicio
-

Componentes de Acceso a Datos

- Los Componentes de Acceso a Datos (CADs) **encapsulan la tecnología de acceso a datos** y la BD al resto de la aplicación
- Proporciona un interfaz sencillo que permite recuperar los datos de la BD y salvar una entidad de negocio en la BD
- Los CADs también contienen cualquier lógica de negocio necesaria para alcanzar las **operaciones relacionadas con los datos**

Operaciones de un CAD

- Un CAD debería proveer los métodos para realizar las siguientes tareas sobre la BD:
 - **Crear** registros en la BD
 - **Leer** registros en la BD y devolver las entidades de negocio al componente invocante
 - **Actualizar** registros en la BD, usando entidades de negocio proporcionadas por el componente invocante
 - **Borrar** registros de la BD
- Estos métodos son llamados **CRUD**, acrónimo de “Create, Read, Update and Delete”

Operaciones de un CAD (II)

- Los CAD pueden contener también métodos que realizan algún filtro. Por ejemplo, un CAD puede tener un método para encontrar el producto más vendido en un catalogo durante un mes
- Un CAD accede a una única BD y encapsula las operaciones relacionadas con una única tabla o un grupo de tablas vinculadas de la BD
- Por ejemplo, podréis definir un CAD que controle las tablas Pedidos y las LineasDePedidos

Ejemplo de CAD en .NET

CAD para la clase Cliente

```
public class ClienteCAD
{
    private String conexion;
    public ClienteCAD()
    {
        // Adquiere la cadena de conexión desde un único sitio
    }
    public ENCliente dameCliente (String id)
    {
        // Código para recuperar un tipo DataSet conteniendo los datos del
        // Cliente
    }
    public String Crear (String nombre, String direccion, String ciudad,
    String pais, int codPostal){
```

Ejemplo de CAD (II)

```
// Código para crear un cliente basado en los parametros escalares
// Devuelve el ID del cliente en este método.
}
public void Actualizar (ENCliente clienteActualizado)
{
//Código para actualizar la BD, basado en el los datos del cliente enviados como
un parámetro de tipo ClienteDataSet
}
public void Borrar (String id)
{
// Código para borrar el cliente con el ID especificado
}
public DataSet dameClientesPorCiudad (string ciudad)
{
// Código para recuperar clientes usando un criterio de búsqueda.
}}
```

Método CAD: BorrarCliente

```
// Método para recuperar el Nombre del Cliente
public void BorrarCliente( String clienteID )
{
    SqlConnection conn = null;
    // Encapsula todo el acceso a datos dentro del try
    String comando = "Delete from Cliente where id = "+ clienteID;
    try
    {
        conn = new SqlConnection(conexion);
        conn.Open();
        SqlCommand cmd = new SqlCommand(comando, conn );
```

Método CAD: BorrarCliente (II)

```
    cmd.ExecuteNonQuery();
}
catch (SqlException sqlex)
{
    // Envuelve la excepción actual en una excepción mas relevante
    throw new CADException ("Error borrando el cliente: " + clienteID, sqlex );
}
catch (Exception ex)
{
    // Captura la condición general y la reenvía.
    throw ex;
}
finally
{
    if(conn != null) conn.Close(); // Se asegura de cerrar la conexión.
}}
```

Método CAD: ObtenerClientesPorCiudad

```
// Método para recuperar los clientes de una determinada ciudad
public DataSet ObtenerClientesPorCiudad( String ciudad )
{
    SqlConnection conn = null;
    DataSet dsClientes = null;
    // Encapsula todo el acceso a datos dentro del try
    string comando = "Select * from Cliente where ciudad = "+ ciudad;
    try
    {
        conn = new SqlConnection(conexion);
        SqlDataAdapter sqlAdaptador = new SqlDataAdapter (comando, conn);
```

Método CAD: ObtenerClientesPorCiudad

```
dsClientes = new DataSet();
sqlAdaptador.Fill (dsClientes);
return dsClientes;
}
catch (SqlException sqlex)
{
    throw new CADException ("Error en la consulta de clientes por ciudad: " +
clienteID, sqlex );
}
catch (Exception ex)
{
// Captura la condición general y la reenvía.
throw ex;
}
finally
{
if(conn != null) conn.Close(); // Se asegura de cerrar la conexión.
}}
```

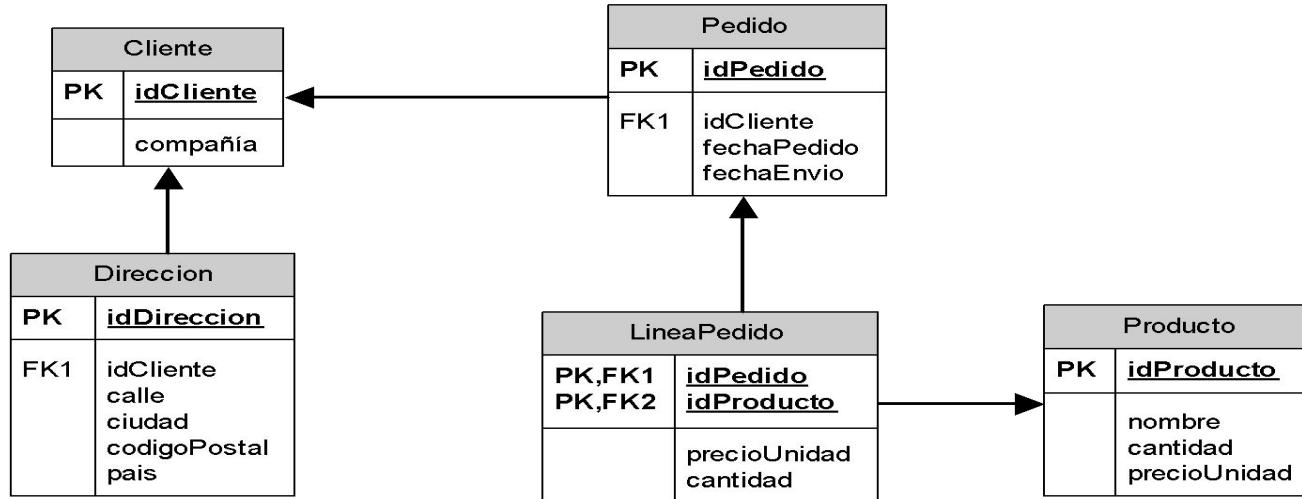
De relacional a EN

- Arquitectura de capas
 - Capa de presentación
 - Entidades de Negocio
 - Componentes de Acceso a Datos
 - **De relacional a EN**
 - Ejercicio
-

De Relacional a Entidad de Negocio

- Una BD contiene múltiples tablas con relaciones y debemos decidir como mapear las tablas en diferentes EN
- Cuando se define las EN se debe considerar “cómo” se usará la información en la aplicación
- Es mejor identificar el núcleo de EN que encapsulan la funcionalidad de la aplicación, antes que definir una EN por cada tabla

De Relacional a Entidad de Negocio



Base de Datos reducida de una aplicación de una Tienda de Venta al por menor

De Relacional a Entidad de Negocio

- Las requisitos funcionales mínimos de una tienda son:
 - Obtener información sobre el Cliente, incluyendo sus direcciones
 - Obtener la lista de pedidos para un cliente
 - Obtener la lista de artículos para un pedido en particular
 - Enviar un nuevo pedido
 - Obtener o actualizar la información de un producto o colección de productos

De Relacional a Entidad de Negocio

- Para completar estos requisitos, podemos hacerlo definiendo tres EN lógicas que controlan la aplicación:
 - Un Cliente que contendrá sus direcciones
 - Un Pedido que contendrá sus líneas de pedido
 - Y un Producto

De Relacional a Entidad de Negocio

- Para cada EN, definimos un CAD que será definido como sigue:
 - **ClienteCAD**: Esta clase provee los servicios para recuperar y modificar los datos de las tablas Cliente y Dirección
 - **PedidoCAD**: Esta clase provee los servicios para recuperar y modificar los datos de las tablas Pedido y LineaPedido
 - **ProductoCAD**: Esta clase provee los servicios para recuperar y modificar los datos de la tabla Producto

De Relacional a Entidad de Negocio: Recomendaciones

- Tómate tu tiempo para analizar y modelar las EN de tu aplicación, en lugar de definir una EN por cada tabla
- Básate en las composiciones y herencias UML para componer objetos complejos
- No definas EN separadas para representar tablas muchos-a-muchos. Estas relaciones pueden ser implementadas mediante colecciones en las EN implicadas

De Relacional a Entidad de Negocio: Recomendaciones

- Definir todos los métodos que devuelven un tipo concreto de Entidad de Negocio en un solo CAD
 - Por ejemplo, si se están recuperando todos los pedidos de un determinado cliente, implementar la función en PedidoCAD llamada **ObtenerPedidosPorCliente** que devuelva los pedidos filtrando por un idCliente
 - Contrariamente, si estás recuperando todos los clientes que han pedido un específico producto, implementa la función en ClienteCAD **ObtenerClientesPorProducto**

Otras tareas que puede realizar un CAD

- Los CADs pueden realizar otras tareas en su implementación:
 - Controlar la seguridad y autorización
 - Realizar la paginación de datos
 - Realizar Transacciones de entidades complejas
 - Invocar a procedimientos almacenados

Ejercicio

- Arquitectura de capas
 - Capa de presentación
 - Entidades de Negocio
 - Componentes de Acceso a Datos
 - De relacional a EN
 - **Ejercicio**
-

EJERCICIO 1:

Ejercicio

- Vamos a identificar las ENs y CADs necesarios para una aplicación web de un campus académico y vamos a definir una de las EN.
- La descripción de la aplicación Web está en la siguiente transparencia.

EJERCICIO 1: Campus Parte pública

- **Visualizar los cursos disponibles:** se podrá ver una breve descripción, nombre, categoría y precio.
- Iniciar sesión o registrarse en su defecto.
- **Búsqueda de cursos:** la búsqueda será un campo de texto el cual buscará sobre el nombre y la categoría.
- **Comentarios:** los comentarios sobre los cursos serán visibles. Estos comentarios aparecerán al seleccionar un curso. Estos comentarios se compondrán del texto, el usuario que lo ha escrito y una valoración optativa (puede ser positiva o negativa).
- **Datos de contacto:** cada vez que se visualice en particular un curso, se mostrarán los datos de contacto de los profesores relacionados (el nombre y email).
- **Avisos recientes:** existirán en la página principal, una lista de avisos de interés general (que serán unos específicos de la parte privada indicados por los profesores).
- **Filtro por categoría:** existirá una lista (se decidirá cómo mostrarla) con todas las categorías existentes. En el caso de seleccionar una, se mostrará una página equivalente a buscar con el buscador el nombre de la categoría).

Distribución capas en el proyecto

—

Dividiremos el código en tres capas o componentes:

- a. **Capa de interfaz de usuario.**
- b. **Capa de lógica de negocio** o *Entidad de Negocio (EN)*.
 - Se le asocia un **CAD** mediante el cual esta **EN** puede almacenarse/modificarse/recuperarse... en la bbdd con la que trabajemos.
- c. **Capa de persistencia** o *Componente de Acceso a Datos (CAD)*.
 - Los **CAD** implementan la lógica de comunicación con la bbdd, la cual es bidireccional entre las **EN** y la bbdd.
 - Las operaciones habituales que proporciona un **CAD** son las de *creación, lectura, actualización y borrado* de registros de la bbdd.

Hada T6: ADO.NET

*Herramientas Avanzadas para el Desarrollo de
Aplicaciones*

Objetivos

- ADO.net 2.0
- Creación de una BD SQL desde VStudio.net
- Acceso conectado a BD
- Creación cadena conexión: Web.config
- Propiedad DataDirectory

ADO.Net 2.0

1

Objetos de acceso a datos (*ActiveX Data Objects*)

- ADO.NET es la tecnología que las aplicaciones asp.net utilizan para comunicarse con la BD.
- Optimizada para aplicaciones distribuidas (como aplicaciones web).
- Basada en XML
- Modelo de objetos completamente nuevo.
- **Entorno conectado vs desconectado.**

Entorno conectado

- Un entorno conectado es aquel en que los usuarios están conectados continuamente a una fuente de datos
- **Ventajas:**
 - El entorno es más fácil de mantener
 - La concurrencia se controla más fácilmente
 - Es más probable que los datos estén más actualizados que en otros escenarios
- **Inconvenientes:**
 - Debe existir una conexión de red constante
 - Escalabilidad limitada

Entorno conectado (II)

CONEXIÓN ABIERTA



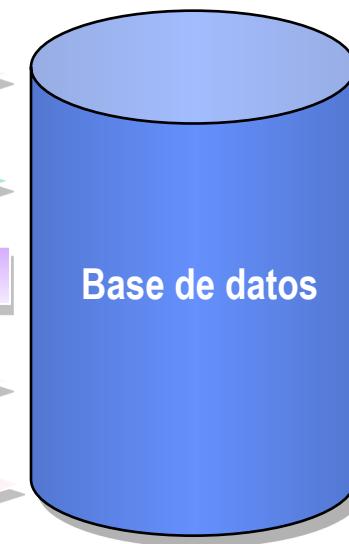
Conectar a una base de datos

Solicitar datos específicos

Devolver datos

Transmitir actualizaciones

Cerrar la conexión



SIN CONEXIÓN

Entorno desconectado

- Un entorno desconectado es aquel en el que los datos pueden modificarse de forma independiente y los cambios se escriben posteriormente en la base de datos.
- **Ventajas:**
 - Las conexiones se utilizan durante el menor tiempo posible, permitiendo que menos conexiones den servicio a más usuarios
 - Un entorno desconectado mejora la escalabilidad y el rendimiento de las aplicaciones
- **Inconvenientes:**
 - Los datos no siempre están actualizados
 - Pueden producirse conflictos de cambios que deben solucionarse

Entorno desconectado (II)

CONEXIÓN ABIERTA



Base de datos

SIN CONEXIÓN

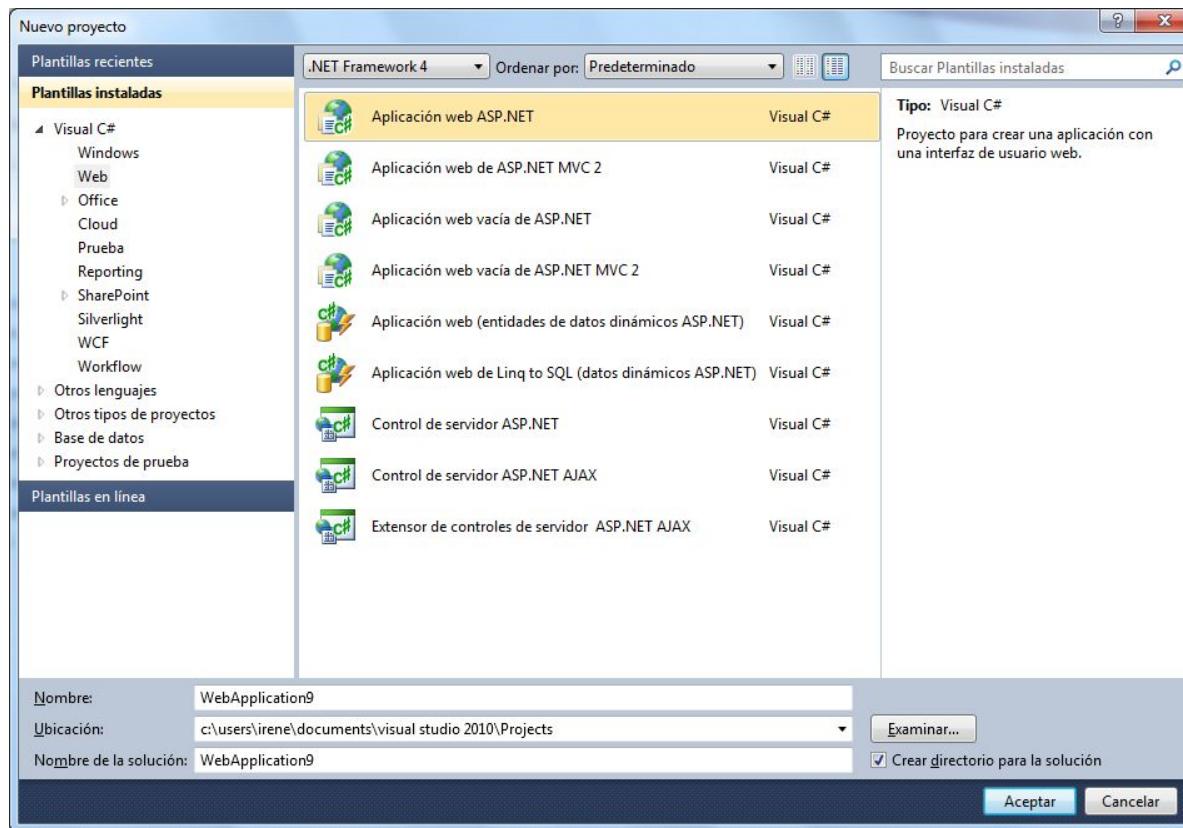
Ejemplos de entornos

- Un supermercado, donde en cada punto de venta (TPV) se almacenan las ventas realizadas. Cada cierto tiempo se las ventas se han de actualizar en la BD central.
- **Una fábrica que requiere una conexión en tiempo real para controlar la salida de producción y el almacén.**
- Una aplicación que mantiene datos de clientes en un equipo portátil de un representante.
- **Una aplicación que hace un seguimiento de lluvias y precipitaciones.**
- Un agente de bolsa que requiere una conexión constante a los valores del mercado.
- **Una aplicación que un granjero utiliza para contar el ganado. La aplicación está en el dispositivo basado en Microsoft Windows CE del granjero que ejecuta Microsoft SQL Server 2000 Windows CE Edition.**

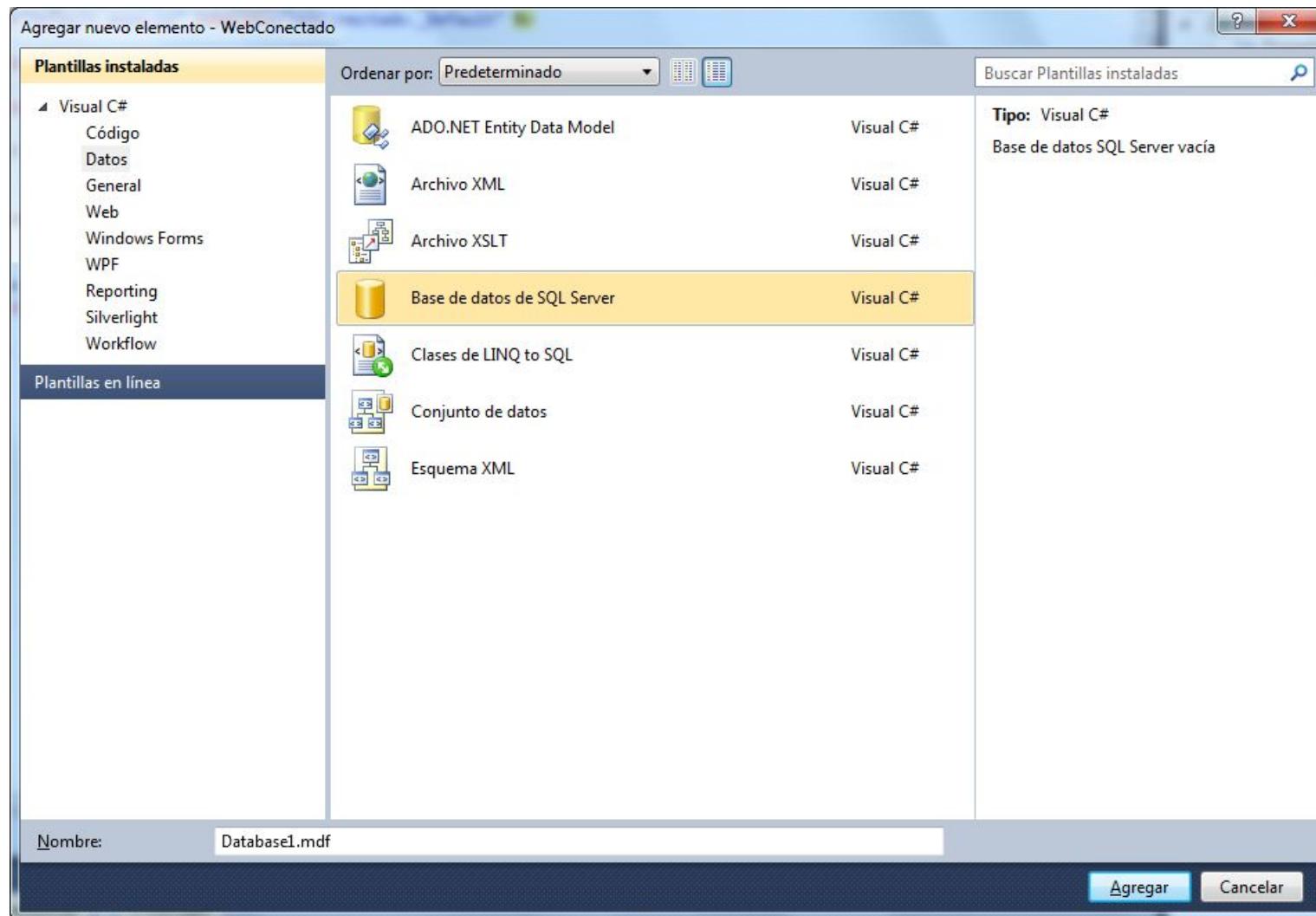
2

Creación BD
desde
Vstudio.net

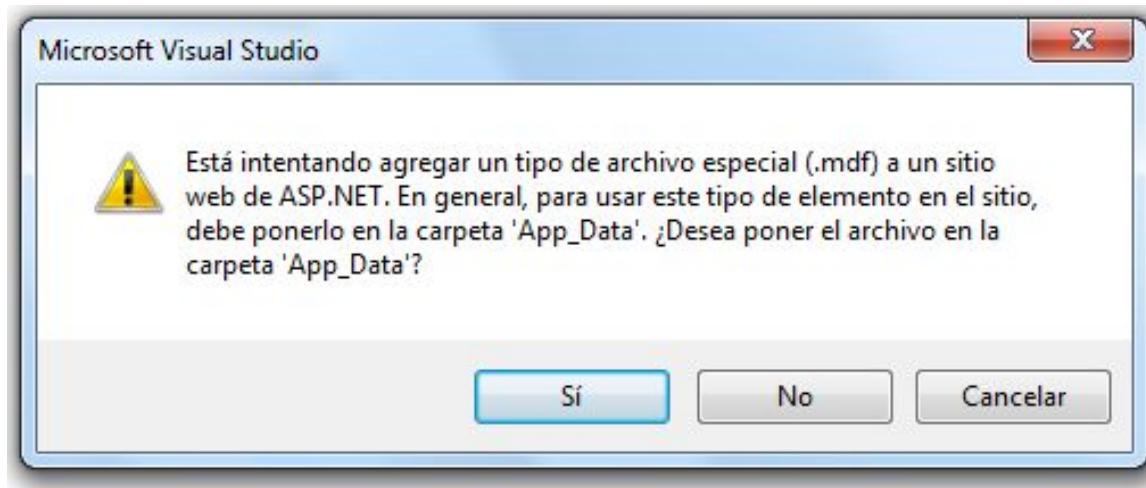
Nuevo proyecto: Aplicación Web C#



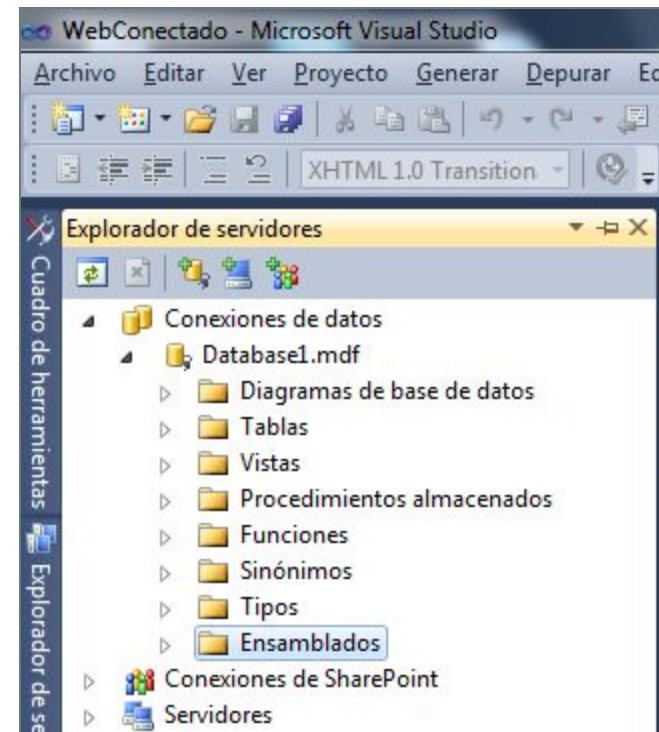
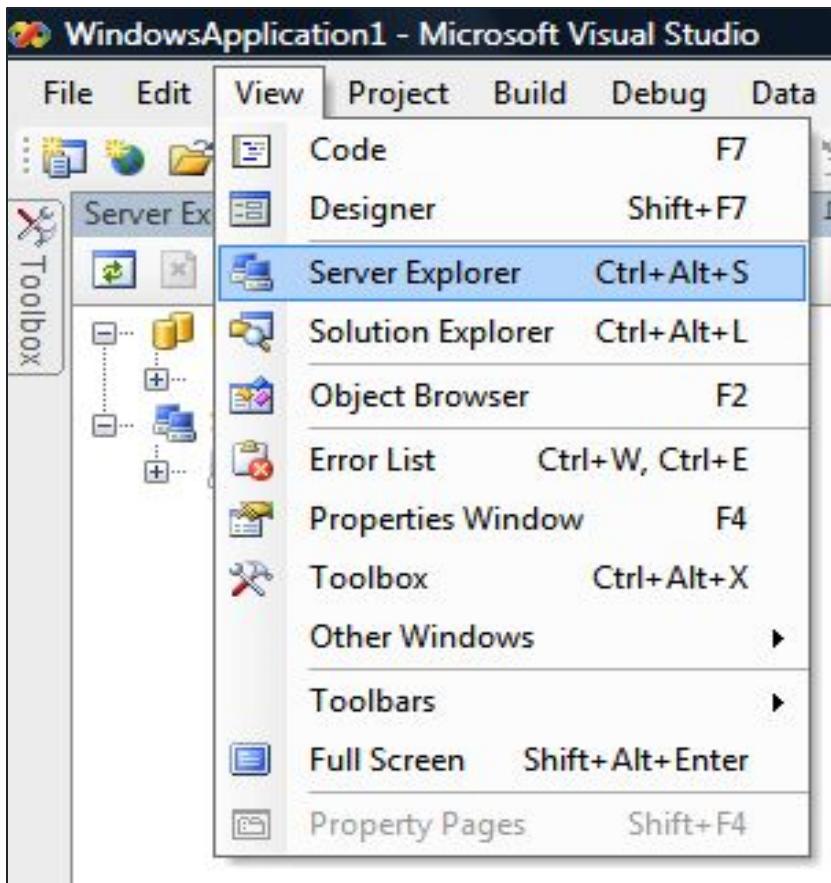
Agregar nuevo elemento: BD SQL server



Carpeta especial App_Data



Ver → explorador de servidores



Añadimos tablas, claves... y datos!

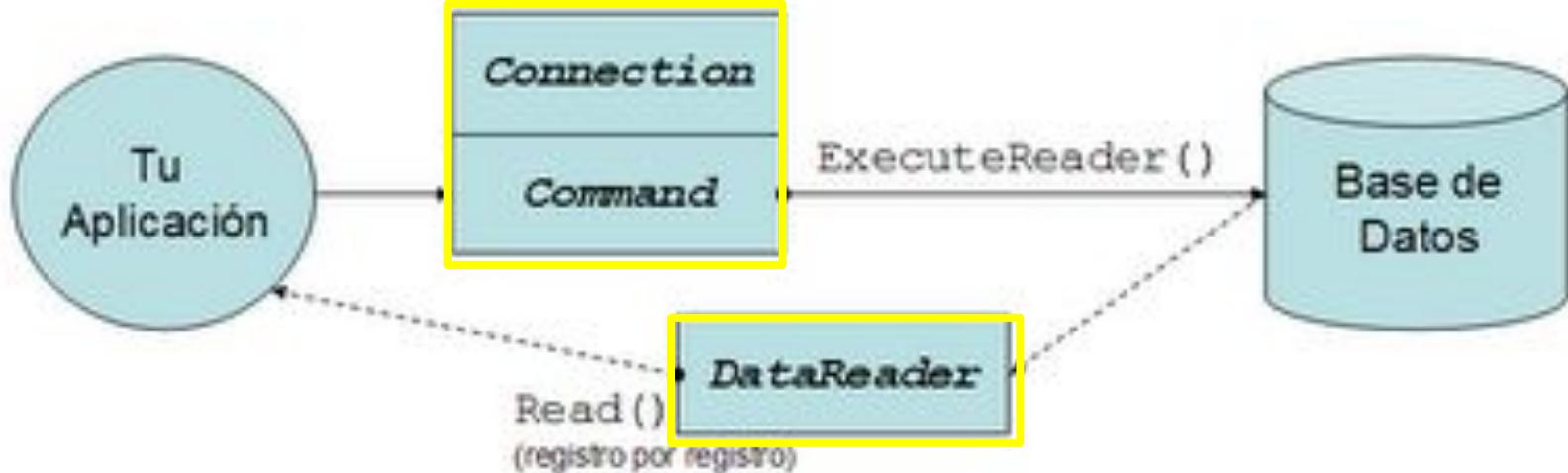
The screenshot shows the SQL Server Management Studio interface. On the left, the 'Explorador de servidores' (Server Explorer) tree view is open, showing 'Conexiones de datos' expanded, with 'Database1.mdf' selected. Under 'Database1.mdf', 'Diagramas de base de datos' is expanded, and 'Tablas' is selected and highlighted with a blue border. On the right, a table definition window titled 'dbo.Table1: Tabl...TA\DATABASE1.MDF)*' is displayed. The table has three columns: 'Nombre de columna' (Name), 'Tipo de datos' (Data type), and 'Permitir val...' (Allow nulls). The columns are listed as follows:

Nombre de columna	Tipo de datos	Permitir val...
usuario	nchar(10)	<input checked="" type="checkbox"/>
contraseña	nchar(10)	<input checked="" type="checkbox"/>
dni	nchar(10)	<input type="checkbox"/>
		<input type="checkbox"/>

3

Acceso
Conectado

Modo conectado



1. Objetos Connection y Command

- **Connection:** se utilizan para establecer las conexiones al proveedor de datos adecuado (método Open).
- **Command:** sirven para ejecutar sentencias SQL y procedimientos almacenados.

Objetos Connection y Command

- `System.Data.SqlClient`: clases responsables del acceso a datos desde fuentes SQL Server.
- Incluyen clases que al trabajar con SQL Server llevarán el prefijo `Sql`:
 - `SqlConnection`
 - `SqlCommand`

2. Objeto DataReader

- Proporcionan un flujo de datos firme.
- Proporcionan un cursor de sólo lectura que avanza por los registros sólo hacia delante.
 - Mantienen una **conexión viva** con el origen de datos, pero no permiten realizar ningún cambio.

Espacios de nombres de datos

- **System.Data**
- **System.Data.Common**
- **System.Data.SqlClient** → Ms SQL Server DB
- **System.Data.SqlTypes** → contiene clases para trabajar con tipos de datos nativos de SQL Server

EJEMPLO: Conexión a una BD en Sql Server

Importar namespaces

```
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.SqlTypes;
```

Crear la conexión

- `string s = "data source=(LocalDB)\MSSQLLocalDB;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf";`

```
SqlConnection c=new SqlConnection(s);
```

Abrir la conexión

```
c.Open();
```

Cadena de Conexión

Parámetro	Descripción
Connection TimeOut	Define el tiempo de espera máximo que debe esperar una conexión para intentar conectar con éxito con el servidor de base de datos. En caso de superar este tiempo se genera una excepción. El tiempo por defecto definido es de 15 segundos.
Data Source	Recibe el nombre del servidor SQL Server utilizado en la conexión.
Integrated Security	Configura nuestra conexión de un modo seguro o no. Recibe como valores True, False y SSPI, siendo True y SSPI el mismo modo de seguridad.
AttachDBFilename	Si se utiliza un nombre de archivo para conectar con la base de datos, se simplifica la implementación de la base de datos con la aplicación (especificamos el archivo mdf)

Propiedad
DataDirectory

Propiedad DataDirectory

- **DataDirectory** es una cadena de sustitución que indica la ruta de acceso de la base de datos.
- **DataDirectory** facilita el uso compartido de un proyecto y la impresión de una aplicación al eliminar la necesidad de definir la ruta de acceso completa. Por ejemplo, en vez de tener la siguiente cadena de conexión:
 - "AttachDbFilename= c:\program files\MyApp\Mydb.mdf"
- Al usar **|DataDirectory|**, puede tener la siguiente cadena de conexión:
 - "AttachDbFilename= **|DataDirectory|**\Mydb.mdf"
- **Para aplicaciones Web, se tendrá acceso a la carpeta App_Data.**

Dónde crear la cadena de conexión???

El archivo web.config

- **Archivo de configuración de la aplicación ASP.NET basado en XML**
- Incluye las opciones de seguridad personalizada, administración de estado, administración de memoria..etc

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.web>
        </system.web>
    <connectionStrings>
        </connectionStrings>
    </configuration>
```

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<!-- For more information on how to configure your ASP.NET application, please visit
<a href="http://go.microsoft.com/fwlink/?LinkId=301880">http://go.microsoft.com/fwlink/?LinkId=301880
-->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
      type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
      Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MvcMovie-20140425082247.mdf;Initial Catalog=aspnet-MvcMovie-20140425082247;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
```

ConnectionString en webconfig

```
<connectionStrings>
  <add name="miconexion"
    connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirector
y|\Database.mdf;Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

cadena de conexión llamada miconexion que hace referencia a una cadena de conexión que se conecta a una instancia de SQL Server.

Recuperar cadenas de conexión de archivos de configuración

- El espacio de nombres **System.Configuration** proporciona clases para trabajar con información de configuración almacenada en archivos de configuración.

C#

```
using System.Configuration;
```

```
string cadena;
```

```
cadena =
```

```
ConfigurationManager.ConnectionStrings["miconexion"].ToString()  
();
```

Se recupera la cadena de conexión del archivo de configuración pasando el nombre de la dicha cadena al ConfigurationManager

Definición de un comando Select

- Para recuperar los datos se necesita:
 - Una sentencia SQL que seleccione la información deseada
 - Un objeto Command que ejecute la sentencia SQL
 - Un objeto DataReader que capture los registros recuperados

Objeto Command

- Los objetos Command representan sentencias SQL. Para utilizar un objeto Command se define la sentencia SQL a utilizar y la conexión disponible y se ejecuta el comando:

```
SqlCommand com=  
new SqlCommand("Select * from clientes",c);
```

Objeto DataReader (I)

- Se utiliza el método ExecuteReader del objeto Command:
 - `SqlDataReader dr= com.ExecuteReader();`
- Se recupera una fila con el método Read
 - `dr.Read()`

Objeto DataReader (II)

- Podemos acceder a los valores de esa fila con el nombre del campo correspondiente:
 - `MyDataReader[“nombrecampo”];`
- Para leer la siguiente fila, volvemos a usar el método `Read`
 - devuelve true si se ha recuperado una fila de información correctamente
 - si devuelve false es porque hemos intentado leer después del final del conjunto de resultados

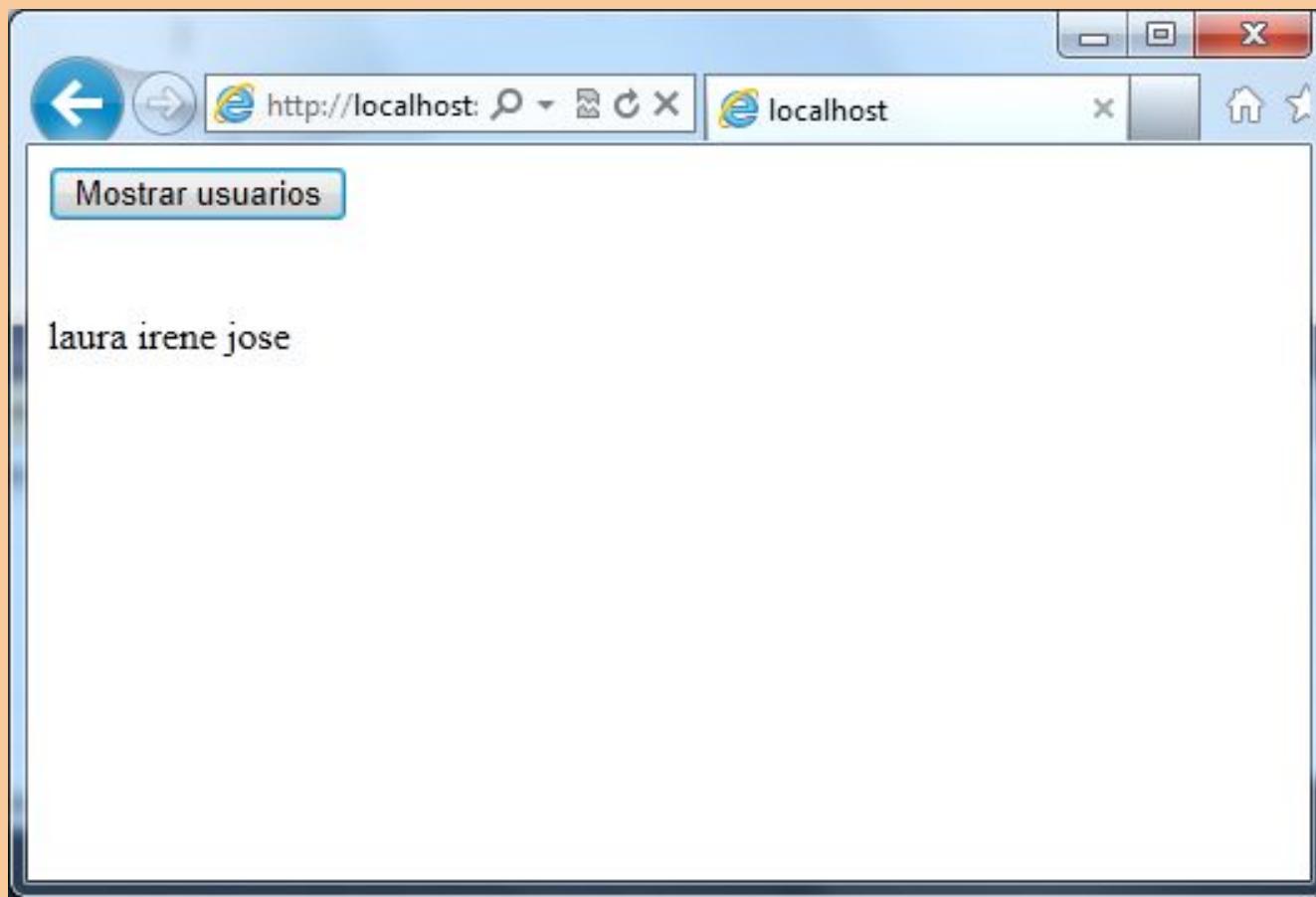
EJERCICIO 1

¿Cómo se haría en el ejemplo que estamos viendo??

- Escribir el código necesario para mostrar en una etiqueta el nombre de los clientes (tabla cliente, columna usuario)

	usuario	contraseña	dni
▶	laura	123	45698
	irene	123	45896
	jose	258	85964
*	NULL	NULL	NULL

Ejecución



En el ejemplo...

```
while ()  
{  
}  
}
```

Cerrar los objetos DataReader y Connection

- dr.Close();
- c.Close();

Importante

- Utilizar manejo de excepciones para conexión a BD:
 - Try/catch

Dónde añadimos try/catch?

```
string s = "data source=(LocalDB)\MSSQLLocalDB;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf";  
  
SqlConnection c=new SqlConnection(s);  
c.Open();  
SqlCommand com= new SqlCommand("Select * from cliente",c);  
SqlDataReader dr= com.ExecuteReader();  
  
while (dr.Read())  
{      this.label1.Text+=dr["usuario"].ToString();  
      label1.Text += " ";  
}  
  
dr.Close();  
c.Close();
```

```
string s = "data source=(LocalDB)\MSSQLLocalDB;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf";  
SqlConnection c=new SqlConnection(s);  
try  
{  
    c.Open();  
    SqlCommand com= new SqlCommand("Select * from cliente",c);  
    SqlDataReader dr= com.ExecuteReader();  
  
    while (dr.Read())  
    {  
        this.label1.Text+=dr["usuario"].ToString();  
        label1.Text += " ";  
    }  
  
    dr.Close();  
}  
catch (Exception ex) { label2.Text = ex.Message; }  
finally  
{  
    c.Close();}
```

Y cómo sería el código teniendo en cuenta las 3 capas?

CAPA INTERFAZ (I)

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Top Bar:** Shows tabs for "Web.config", "WebForm1.aspx.cs", "CADcliente.cs", "ENCliente.cs", and "WebForm1.aspx". The "WebForm1.aspx" tab is selected, and the status bar indicates "cliente: consulta...TA\DATABASE1.MDF".
- Code Editor:** Displays the ASPX page source code. It includes an HTML structure with a form containing a button and a label, and a script block with an onclick event.
- Design View:** Below the code editor, the design view shows the visual representation of the page. It features a single button labeled "Mostrar usuarios" and a label control below it with the text "[Label1]".
- Status Bar:** At the bottom, the status bar shows navigation icons: "Diseño" (Design), "Dividir" (Split), "Código" (Code), and "HTML" (HTML).

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebConectado.WebForm1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
</div>
<asp:Button ID="Button1" runat="server" Text="Mostrar usuarios" onclick="Button1_Click" />
<br />
<br />
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
</form>
</body>
</html>
```

CAPA INTERFAZ (II)

```
using System;
...
using System.Collections;
namespace WebConectado
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        ArrayList a = new ArrayList();

        protected void Button1_Click(object sender, EventArgs e)
        {
            ENCliente en = new ENCliente();
            a=en.listarClientes();

            foreach (string s in a)
                Label1.Text += s + " ";
        }
    }
}
```

CAPA EN

```
namespace WebConectado
{
    public class ENCliente
    {
        private string usuario;
        public string Usuario
        {   get { return usuario; }
            set { usuario = value; }
        }
        private string dni;
        public string Dni
        {   get { return dni; }
            set { dni = value; }
        }
        private string contraseña;
        public string Contraseña
        {   get { return contraseña; }
            set { contraseña = value; }
        }
    }
}
```

```
public ArrayList listarClientes()
{
    ArrayList a = new ArrayList();
    CADcliente c = new
    CADcliente();
    a=c.ListarClientes();

    return a;
}
}
```

SE PODRIA SIMPLIFICAR???

EJERCICIO 2:

CAPA CAD

Cómo sería poniendo la cadena de conexión en Web.config?? Indicar código de Web.config y de la capa CAD

```
public class CADcliente{  
    ArrayList lista = new ArrayList();  
    string s = "data source=(LocalDB)\MSSQLLocalDB;Integrated  
    Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf";  
    public ArrayList ListarClientes(){  
        SqlConnection c = new SqlConnection(s);  
        c.Open();  
        SqlCommand com = new SqlCommand("Select * from cliente", c);  
        SqlDataReader dr = com.ExecuteReader();  
        while (dr.Read())  
        {  
            lista.Add(dr["usuario"].ToString());  
        }  
        dr.Close();  
        c.Close();  
    return lista;  
    }  
}
```

Modificación de datos: Insert, Update, Delete

- En este caso no es necesario un objeto DataReader ya que no se recupera ningún resultado.
- Tampoco un comando Select de SQL, sino:
 - Update
 - Insert
 - Delete
- Necesitamos crear un objeto Command para ejecutar la sentencia SQL apropiada.
- Método ExecuteNonQuery: obtiene el número de registros afectados.

Ejecución de comandos

- ExecuteNonQuery
 - Ejecuta un comando y no devuelve ningún resultado (obtiene el número de registros afectados.)
- ExecuteReader
 - Ejecuta un comando y devuelve un comando que implementa DataReader (Permite iterar a partir de los registros recibidos)

EJERCICIO 3

Implementar en Visual Studio: Insertar usuarios



```
string s = "data source=(LocalDB)\MSSQLLocalDB;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf";  
  
SqlConnection c=new SqlConnection(s);  
try  
{  
c.Open();  
SqlCommand com= new SqlCommand("Select * from cliente",c);  
SqlDataReader dr= com.ExecuteReader();  
  
while (dr.Read())  
{  
    this.label1.Text+=dr["usuario"].ToString();  
    label1.Text += " ";  
}  
  
dr.Close();  
}  
catch (Exception ex) { label2.Text = ex.Message; }  
finally  
{  
    c.Close();  
}
```

```
string s = "data source=(LocalDB)\MSSQLLocalDB;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf"  
;  
  
SqlConnection c=new SqlConnection(s);  
try  
{  
c.Open();  
SqlCommand com = new SqlCommand("Insert Into Cliente  
(usuario,contraseña,dni) VALUES ('" + textBox1.Text + "','" +  
textBox3.Text + "','" + textBox2.Text + "')", c);  
  
com.ExecuteNonQuery();  
  
}  
catch (Exception ex) { label2.Text = ex.Message; }  
finally  
{  
    c.Close();  
}
```

Interfaz

```
protected void Button2_Click(object sender,  
EventArgs e)  
{  
    ENCliente en = new ENCliente();  
    en.Usuario = TextBox1.Text;  
    en.Contraseña = TextBox2.Text;  
    en.Dni = TextBox3.Text;  
  
    en.InsertarCliente();  
}
```

ENCliente

```
public void InsertarCliente()  
{  
    CADcliente c = new CADcliente();  
    c.InsertarCliente(this);  
}
```

CADCliente

```
public void InsertarCliente(ENCliente cli)
{
    SqlConnection c = new SqlConnection(s);
    c.Open();

    SqlCommand com = new SqlCommand("Insert Into Cliente
        (usuario,contraseña,dni) VALUES ('" + cli.Usuario + "','" +
        cli.Contraseña + "','" +
        cli.Dni + "')", c);

    com.ExecuteNonQuery();
    c.Close();
}
```

Hada T7:ADO.NET (2/2)

*Herramientas Avanzadas para el Desarrollo de
Aplicaciones*

1. Repaso
2. Acceso desconectado a BD
3. Controles de datos
4. Concurrency
5. Acceso conectado vs desconectado

OBJETIVOS

1

Repaso

Objetos de acceso a datos (*ActiveX Data Objects*)

- ADO.NET es la tecnología que las aplicaciones .net utilizan para comunicarse con la BD.
- Optimizada para aplicaciones distribuidas (como aplicaciones web).
- Basada en XML
- Modelo de objetos completamente nuevo.
- **Entorno conectado vs desconectado.**

Entorno conectado



SIN CONEXIÓN

Entorno conectado (II)

- Un entorno conectado es aquel en que los usuarios están conectados continuamente a una fuente de datos
- **Ventajas:**
 - El entorno es más fácil de mantener
 - La concurrencia se controla más fácilmente
 - Es más probable que los datos estén más actualizados que en otros escenarios
- **Inconvenientes:**
 - Debe existir una conexión de red constante
 - Escalabilidad limitada

Entorno desconectado



SIN CONEXIÓN

Entorno desconectado (II)

- Un entorno desconectado es aquel en el que los datos pueden modificarse de forma independiente y los cambios se escriben posteriormente en la base de datos.
- **Ventajas:**
 - Las conexiones se utilizan durante el menor tiempo posible, permitiendo que menos conexiones den servicio a más usuarios
 - Un entorno desconectado mejora la escalabilidad y el rendimiento de las aplicaciones
- **Inconvenientes:**
 - Los datos no siempre están actualizados
 - Pueden producirse conflictos de cambios que deben solucionarse

Tabla cliente...

WindowsApplication1 - Microsoft Visual Studio

File Edit View Project Build Debug Data Table Designer Tools Window Community Help

Server Explorer Form1.vb [Design] Start Page

Column Name Data Type Allow Nulls

Column Name	Data Type	Allow Nulls
usuario	nchar(10)	<input checked="" type="checkbox"/>
contraseña	nchar(10)	<input checked="" type="checkbox"/>
dni	nchar(10)	<input type="checkbox"/>

Toolbox

Server Explorer

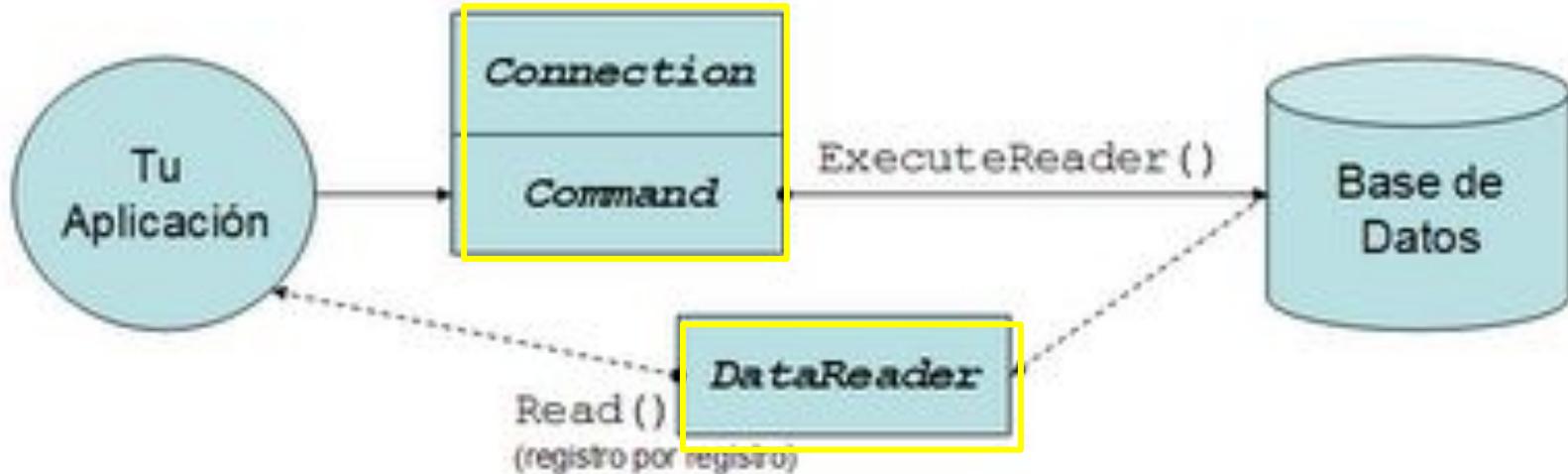
Data Connections

- irene1\sqlexpress.prueba.dbo
- irene1\sqlexpress.prueba1.dbo
 - Database Diagrams
 - Tables
 - cliente
 - usuario
 - contraseña
 - dni
 - Views
 - Stored Procedures
 - Functions
 - Synonyms
 - Types
 - Assemblies

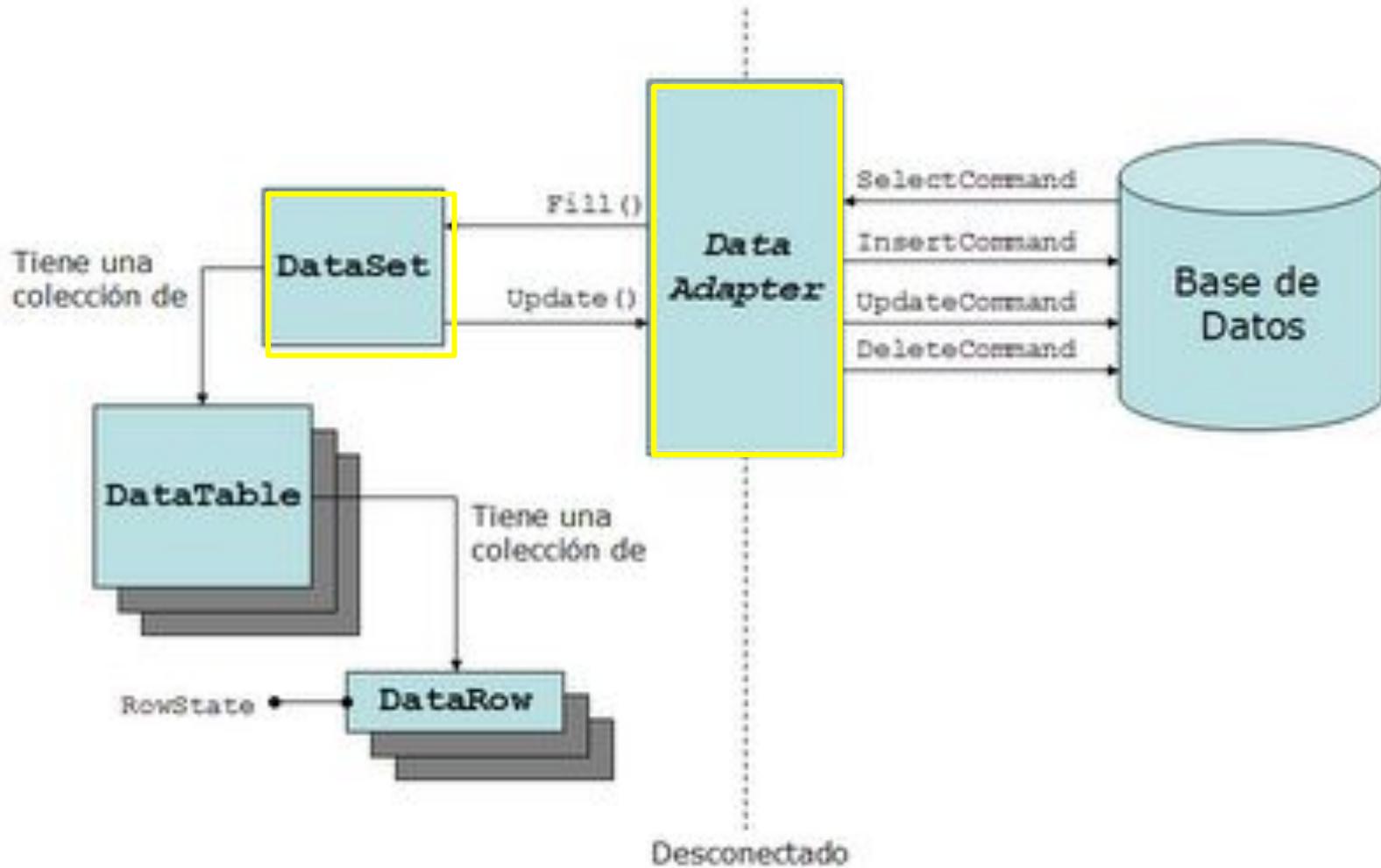
Servers

- irene1

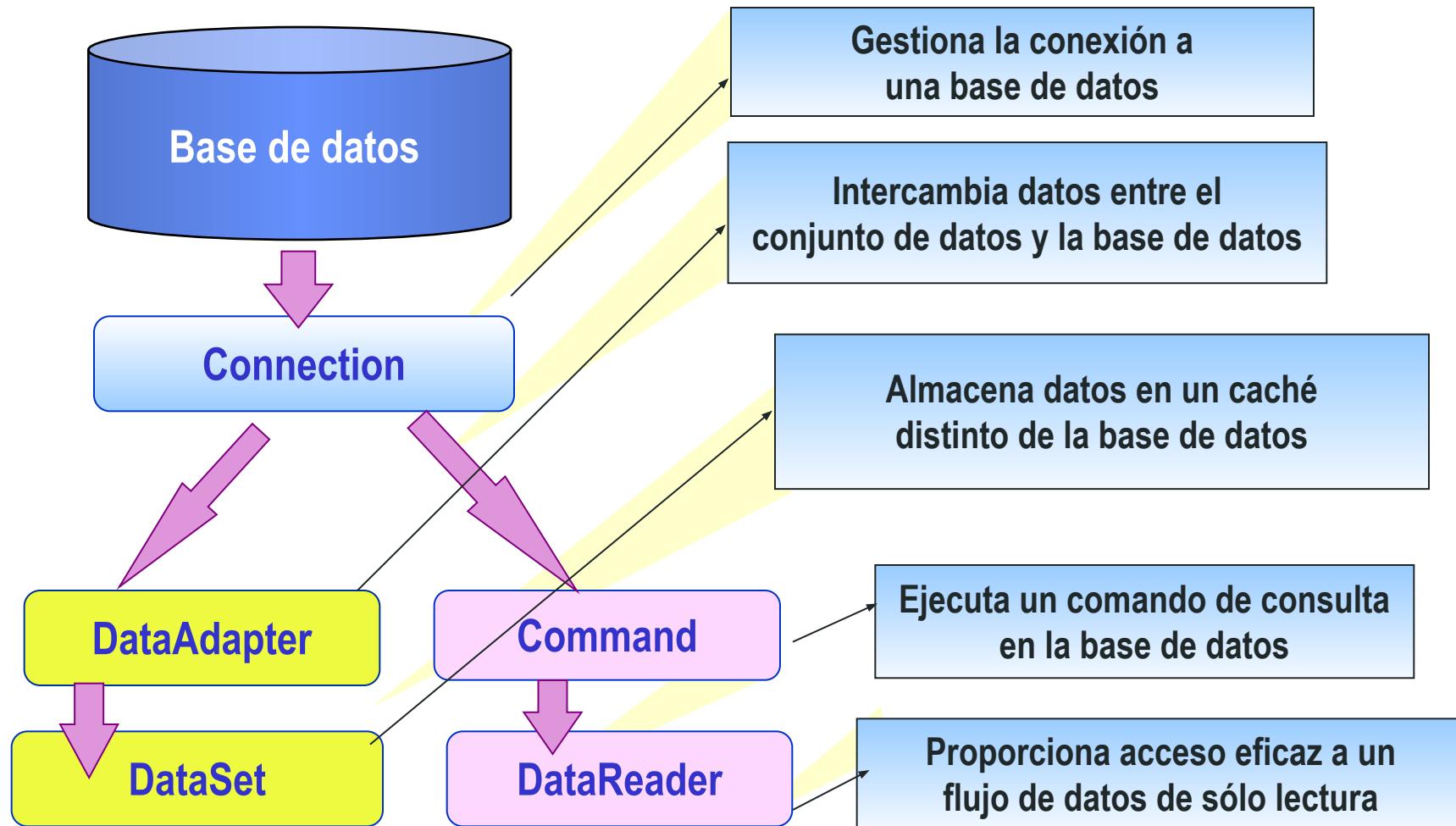
Modo conectado



Modo desconectado



Objetos de ADO.NET



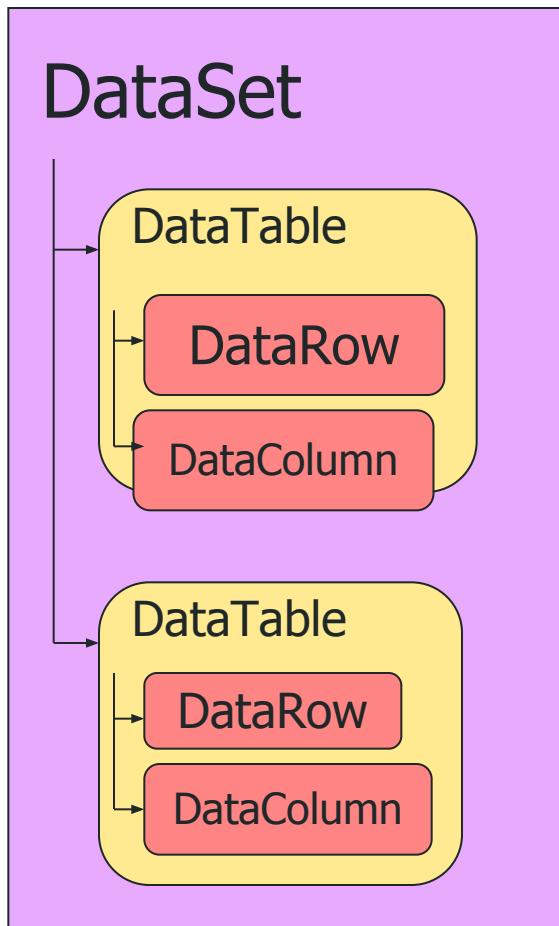
2

Acceso
desconectado

1. Objeto Connection

- **Connection:** se utiliza para establecer las conexiones al proveedor de datos adecuado (método Open).

2. Objeto DataSet



- Nuevo objeto **DataSet**: representación de una **base de datos** relacional **en memoria**:
 - No necesidad de conexión continua.

Objeto DataSet (II)

- Podemos trabajar con una BD que es copia de las partes con las que queremos trabajar de la BD real, liberando la conexión.
- Si queremos reflejar los cambios en la BD real, debemos **confirmar nuestro objeto DataSet**.

Objetos DataRow, DataColumn

- **DataRow**
 - Representa una única fila de información de la tabla.
 - Se puede acceder a los valores individuales usando el nombre de campo o un índice.
- **DataColumn**
 - No contienen ningún dato real.
 - Almacenan información sobre la columna (tipo de datos, valor predeterminado, restricciones..)

Objetos DataRelation, DataView

- DataRelation
 - Especifica una relación padre/hijo entre dos tablas diferentes de un objeto DataSet.
- DataView
 - Proporciona una vista sobre una DataTable.
 - Cada DataTable tiene al menos un DataView (a través de la propiedad DefaultView), que se usa para la vinculación de datos.
 - DataView muestra los datos de DataTable sin cambios o con opciones especiales de filtro u ordenación.

3. Objeto Adaptador

- El objeto Adaptador de datos se encarga de manejar la conexión por nosotros.
- Se utiliza para insertar datos en un objeto **DataSet**.
- El objeto **DataAdapter** utiliza comandos para actualizar el origen de datos después de hacer modificaciones en el objeto **DataSet**.

DataAdapter (II)

- Una ventaja al usar un DataAdapter es que **no tengo que preocuparme por abrir y cerrar la conexión**. Estos métodos lo hacen automáticamente

4. Objeto CommandBuilder

- Este objeto (opcional) lo utiliza el **DataAdapter** para crear los comandos SQL necesarios.
- También se pueden proporcionar las sentencias SQL explícitamente o por medio de procedimientos almacenados.

Recordad...

- `System.Data.SqlClient`: clases responsables del acceso a datos desde fuentes SQL Server.
- Incluyen clases que al trabajar con SQL llevarán el prefijo `Sql`:
 - `SqlConnection`
 - `SqlDataAdapter`
 - `SqlCommandBuilder`
 - **PERO NO CON DataSet (ni dataRow, dataColumn...)**

EJEMPLO: inserción de datos

Usuario	<input type="text" value="luis"/>
Contraseña	<input type="text" value="1256"/>
DNI	<input type="text" value="55555"/>
<input type="button" value="Insertar usuario"/>	

Conexión a una BD en Sql Server

Importar namespaces

```
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.SqlTypes;
```

Crear la conexión

```
string s =  
"Data Source=(LocalDB)\MSSQLLocalDB;  
AttachDbFilename=|DataDirectory|\Database.mdf;  
Integrated Security=True"  
  
SqlConnection c=new SqlConnection(s);
```

Definición de un comando Select

- Para recuperar los datos se necesita:
 - Una sentencia SQL que seleccione la información deseada
 - Un objeto Command que ejecute la sentencia SQL
 - Un objeto DataReader que capture los registros recuperados

ATENCIÓN: En modo desconectado siempre necesitamos recuperar los datos (SELECT) para poder trabajar con ellos (INSERT, UPDATE, DELETE)

Definición de un comando Select

- Para recuperar los datos se necesita:
 - Una sentencia SQL que seleccione la información deseada
 - Un objeto Command que ejecute la sentencia SQL
 - Un objeto DataReader que capture los registros recuperados
- Un objeto **DataAdapter** que ejecute la sentencia SQL
- Un objeto **DATASET** donde guardar el resultado de la sentencia SQL

Objetos DataSet y DataAdapter

- Creamos una BD virtual, mediante un objeto DataSet

```
DataSet bdvirtual = new DataSet();
```

- La llenamos con las tablas que estamos interesados en trabajar:
 - Objeto DataAdapter
 - Método Fill()

...

```
SqlDataAdapter da = new  
SqlDataAdapter("select * from Cliente", c);  
  
da.Fill(bdvirtual,"cliente");
```

Ahora trabajamos en local

- Ahora en “bdvirtual” tenemos nuestra base de datos local.

Para trabajar en local

- Lo hacemos modificando filas y columnas de las tablas almacenadas en local.
- En dataset tenemos almacenada la bd virtual, copiamos a un datatable la tabla a modificar.

```
DataTable t = new DataTable();
t = bdvirtual.Tables["cliente"];
```

Operaciones

- Obtener una tabla:
 - `DataTable t = new DataTable();`
 - `t = bdvirtual.Tables["cliente"];`
- Acceder a los elementos filas de esa tabla (podemos usar un bucle):
 - `DataRow fila = t.Rows[0];`
 - `fila[0] = "Andrés";`
 - **IGUAL QUE**
 - `t.Rows[0][0] = "Andrés";`

Primera fila segunda columna:

- `t.Rows[0][1]`
- Información de las columnas: (nombre, tipo)
 - `t.Columns[0].ColumnName`
 - `t.Columns[0].DataType`

Queremos insertar un nuevo cliente

- Esto equivale a insertar una fila en nuestra tabla local...

```
DataRow nuevafila = t.NewRow();
nuevafila[0] = textBox1.Text;
nuevafila[1] = textBox2.Text;
nuevafila[2] = textBox3.Text;
t.Rows.Add(nuevafila);
```

Validar los cambios

- Objeto Adaptador:
 - nos ha servido para llenar la tabla, también para actualizar los datos en la BD real.
- Método Update
 - El *DataAdapter* analizará los cambios que se han hecho en el *DataSet* y ejecutará los comandos apropiados para *insertar*, *actualizar* o *borrar* en la BD real.

Constructor de comandos

- Objeto CommandBuilder:
 - Constructor de comandos
 - Le pasamos como argumento el adaptador
 - Construye los comandos SQL que nos hagan falta para actuar sobre la BD

```
• SqlCommandBuilder cbuilder = new  
SqlCommandBuilder(da);  
• da.Update(bdvirtual, "cliente");  
• label4.Text = "Cambiado";
```

Ahora faltaría actualizar los cambios en la BD real

- Objeto CommandBuilder
- Objeto DataAdapter → Método UPDATE

```
SqlCommandBuilder cbuilder = new  
SqlCommandBuilder(da);  
  
da.Update(bdvirtual, "cliente");
```

```
string s = "Data Source=(LocalDB)\MSSQLLocalDB;  
AttachDbFilename=|DataDirectory|\Database.mdf;Integrated  
Security=True"  
SqlConnection c=new SqlConnection(s);  
DataSet bdvirtual = new DataSet();  
SqlDataAdapter da = new SqlDataAdapter("select * from Cliente", c);  
da.Fill(bdvirtual, "cliente");
```

```
DataTable t = new DataTable();  
t = bdvirtual.Tables["cliente"];  
DataRow nuevafila = t.NewRow();  
nuevafila[0] = textBox1.Text;  
nuevafila[1] = textBox2.Text;  
nuevafila[2] = textBox3.Text;  
t.Rows.Add(nuevafila);
```

¿Qué faltaría en el código?

```
SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);  
da.Update(bdvirtual, "cliente");  
label4.Text = "CAMBIADO";
```

En las 3 capas, modificamos la función CAD

```
public bool InsertarCliente(ENCliente cli)      {  
    bool cambiado;  
    DataSet bdvirtual = new DataSet() ;  
    SqlConnection c = new SqlConnection(s);  
    try      {  
        SqlDataAdapter da = new SqlDataAdapter("select * from Cliente",c);  
        da.Fill(bdvirtual,"cliente");  
        DataTable t = new DataTable();  
        t = bdvirtual.Tables["cliente"];  
        DataRow nuevafila = t.NewRow();  
        nuevafila[0] = cli.Usuario;  
        nuevafila[1] = cli.Contraseña;  
        nuevafila[2] = cli.Dni;  
        t.Rows.Add(nuevafila);  
        SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);  
        da.Update(bdvirtual, "cliente");  
        cambiado = true;  
    }  
    catch (Exception ex) {    cambiado = false;      }  
    finally    {      c.Close();          }  
    return cambiado;  }
```

Ejecución

Mostrar usuarios

Usuario	<input type="text" value="usuario1"/>
Contraseña	<input type="text" value="34343"/>
DNI	<input type="text" value="44444444"/>

CAMBIADO

cliente: consulta...TA\DATABASE1.MDF) X Web.config

	usuario	contraseña	dni
▶	irenee	345	4343
	usuario1	34343	44444444
	laura	123	45698
	irene	123	45896
	luis	1256	55555
	jose	258	85964
*	ii	343	rr
	NULL	NULL	NULL

3

Control GridView

Control GridView

- Control para presentación de datos en forma de tabla (filas y columnas)
- Propiedades:
 - Selección
 - Paginación
 - Ordenación
 - Edición
 - Extensible mediante plantillas
- [http://msdn.microsoft.com/es-es/library/cc295223\(v=expression.40\).aspx](http://msdn.microsoft.com/es-es/library/cc295223(v=expression.40).aspx)

GridView

- Datos
 - Puntero
 - GridView
 - DataList
 - DetailsView
 - FormView
 - ListView
 - Repeater
 - DataPager
 - SqlDataSource
 - AccessDataSource
 - LinqDataSource
 - ObjectDataSource
 - XmlDataSource
 - SitemapDataSource

asp:gridview#GridView1

Column0	Column1	Column2
abc	abc	abc

Tareas de GridView

- Formato automático...
- Elegir origen de datos: (Ninguno)
- Editar columnas...
- Agregar nueva columna...
- Editar plantillas

asp:sqldatasource#SqlDataSource1

SqlDataSource - SqlDataSource1

Tareas de SqlDataSource

- Configurar origen de datos...

Objetivo:

Vamos a mostrar en un gridview los datos de la tabla cliente.

- *Con el asistente*
- *Con código*

GridView → DataSource

- Añadir un objeto GridView y elegir como origen de datos la bd.

Control GridView

- Podemos asignarle una tabla de la BD como origen de datos, o los registros obtenidos como resultado de una sentencia SQL.

Elegir BD

Asistente para la configuración de orígenes de datos

Elija un tipo de origen de datos

¿De dónde obtendrá la aplicación los datos?

Archivo XML **Base de datos** Base de datos de Access Entity LINQ Mapa del sitio Objeto

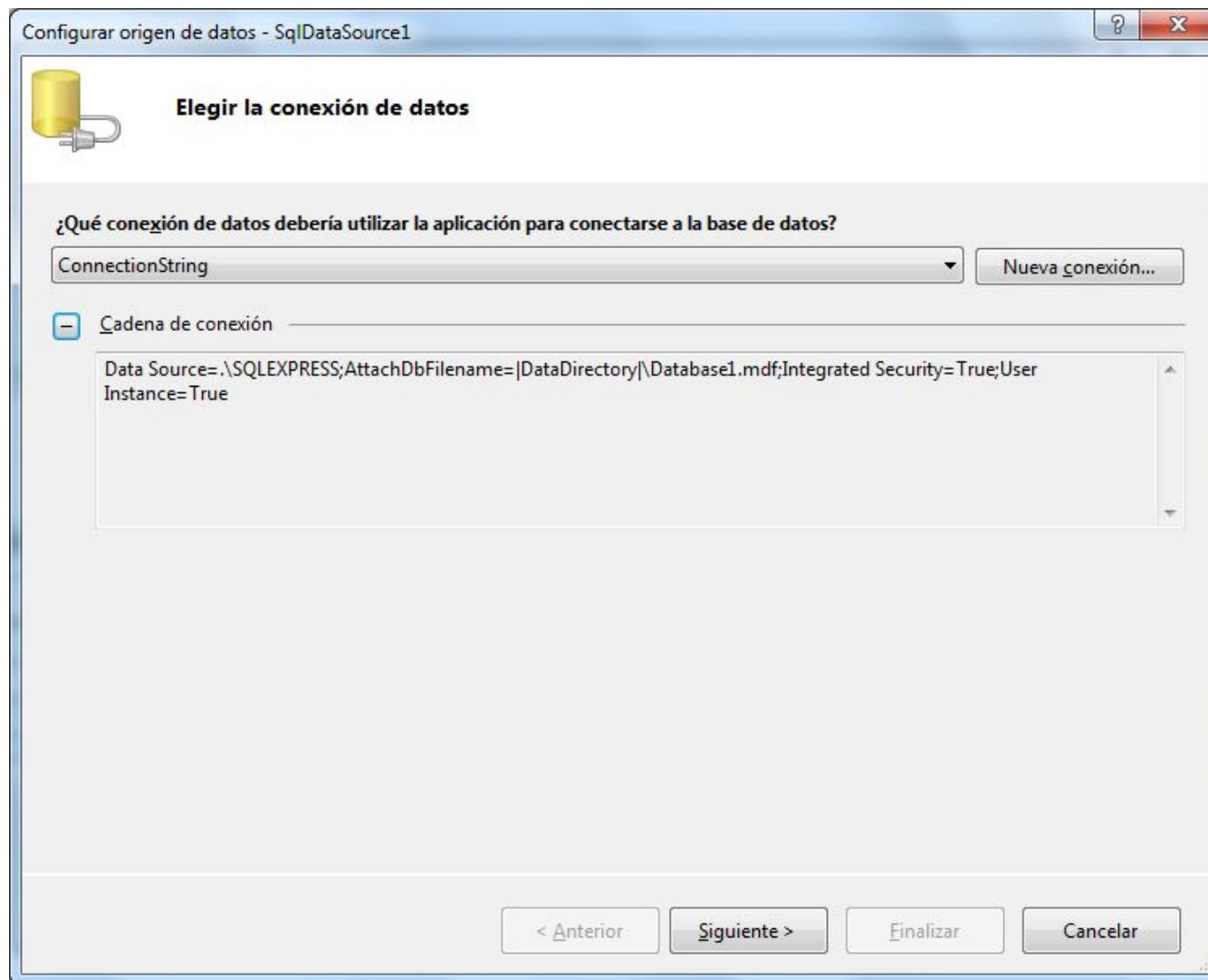
Conectar con cualquier base de datos SQL compatible con ADO.NET, como Microsoft SQL Server, Oracle u OLEDB.

Especifique un identificador para el origen de datos:

SqlDataSource1

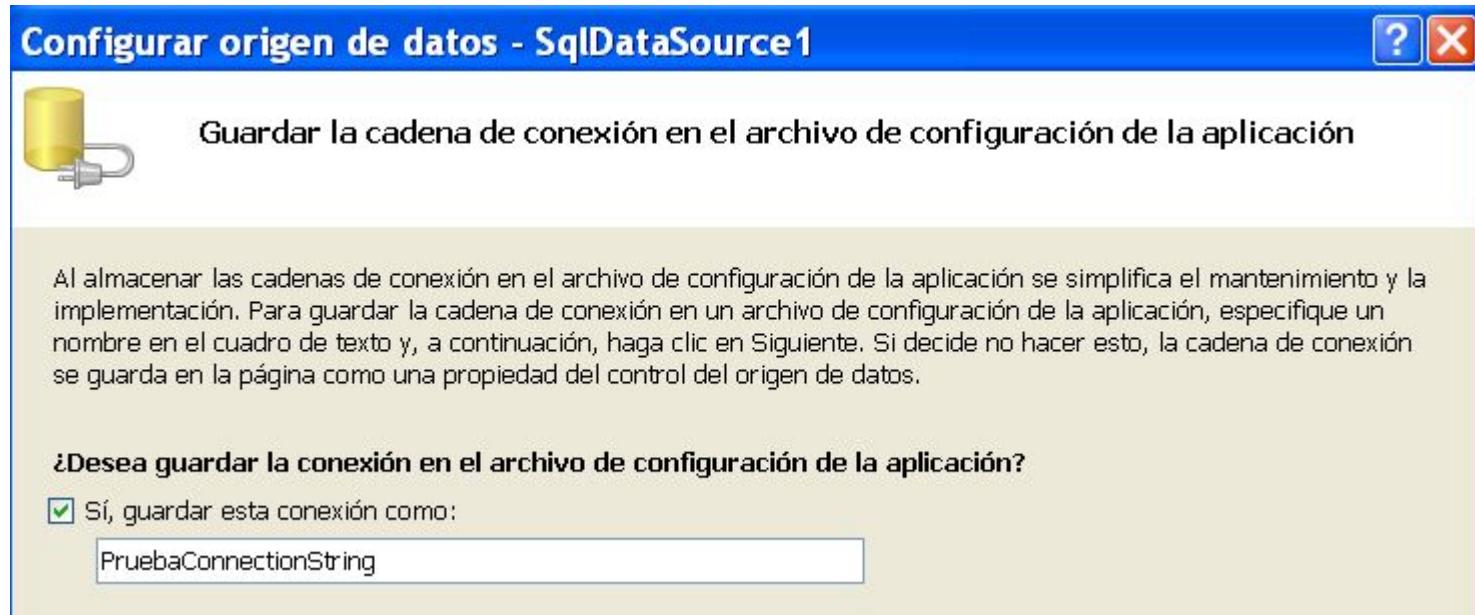
Aceptar Cancelar

Elegir conexión de datos



GridView

- Guardamos la cadena de conexión en el archivo Web.config



Configurar origen de datos - SqlDataSource1

 **Configurar la instrucción Select**

¿Cómo desea recuperar los datos de la base de datos?

Especificar una instrucción SQL o un procedimiento almacenado personalizado
 Especificar columnas de una tabla o vista

Nombre:
cliente

Columnas:

*
 usuario
 contraseña
 dni

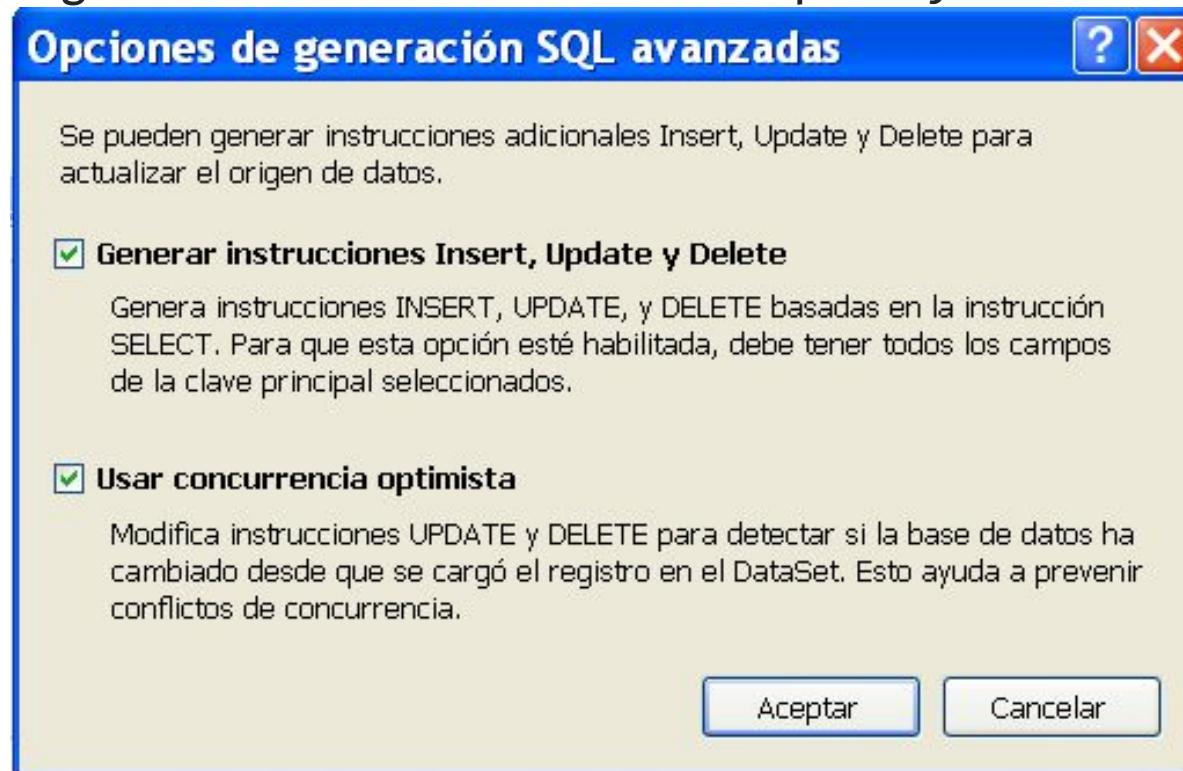
Devolver sólo filas únicas
WHERE...
ORDER BY...
Avanzadas...

Instrucción SELECT:
SELECT * FROM [cliente]

< Anterior Siguiente > Finalizar Cancelar

GridView (asistente)

- Avanzadas
 - Podemos generar las instrucciones Insert, Update y Delete



Ejecución...

The screenshot shows a Microsoft Internet Explorer window with the URL `http://localhost:49221/WebForr`. The page displays a form for inserting a new user and a table showing existing users.

Form Fields:

- Mostrar usuarios** (button)
- Usuario**: Text input field
- Contraseña**: Text input field
- DNI**: Text input field
- Insertar usuario** (button)

User Table:

usuario	contraseña	dni
irenee	345	4343
usuario1	34343	44444444
laura	123	45698
irene	123	45896
luis	1256	55555
jose	258	85964
ii	343	rr

Código para mostrar datos de un dataset

```
ENCliente enl = new ENCliente();
DataSet d = new DataSet();

    protected void Page_Load(object sender,
EventArgs e)
{
    if (!Page.IsPostBack)
    {
        d = enl.listarClientesD();

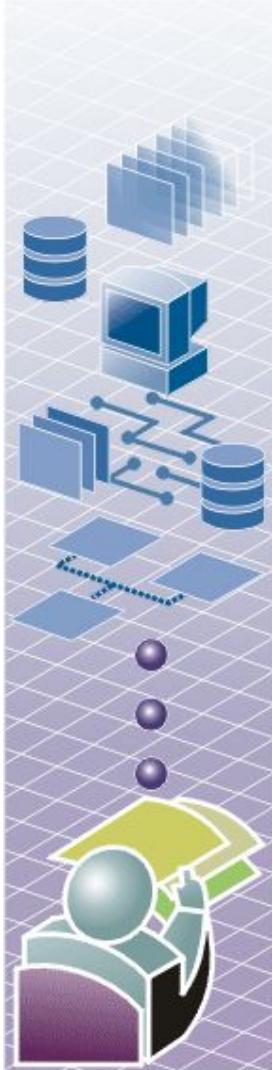
        GridView1.DataSource = d;
        GridView1.DataBind();
    }
}
```

Donde en el CAD...

```
public DataSet ListarClientesD()
{
    DataSet bdvirtual = new DataSet();

    SqlConnection c = new SqlConnection(s);
    SqlDataAdapter da = new SqlDataAdapter("select *
from Cliente", c);
    da.Fill(bdvirtual, "cliente");
    return bdvirtual;
}
```

Ejercicio



Ejercicio

- Modificar el formulario anterior para editar los datos de una persona en la BD.

5 min



AutoGenerateSelectButton=true

Usuario	<input type="text"/>		
Contraseña	<input type="text"/>		
DNI	<input type="text"/>		
Insertar usuario	Editar usuario		
	<u>usuario</u>	<u>contraseña</u>	<u>dni</u>
Seleccionar	laura	1237	44444444
Seleccionar	laura2	1239	4569
Seleccionar	irene	123	45896
Seleccionar	luis	1256	55555
Seleccionar	jose	258	85964
1 2			

Evento SelectedIndexChanged

```
protected void GridView2_SelectedIndexChanged(object sender,  
EventArgs e)  
{  
    TextBox1.Text = GridView2.SelectedRow.Cells[1].Text;  
    TextBox2.Text = GridView2.SelectedRow.Cells[2].Text;  
    TextBox3.Text = GridView2.SelectedRow.Cells[3].Text;  
    TextBox3.Enabled = false;  
}
```

Botón editar (click)

```
{  
  
    ENCliente en = new ENCliente();  
    en.Usuario = TextBox1.Text;  
    en.Contraseña = TextBox2.Text;  
    d = en.ModificarCliente(GridView2.SelectedIndex);  
  
    GridView2.DataSource = d;  
    GridView2.DataBind();  
}
```

EN

```
public DataSet ModificarCliente(int i)
{
    CADcliente c = new CADcliente();
    DataSet a = c.ModificarCliente(this,i);
    return a;
}
```

CAD (simplificado)

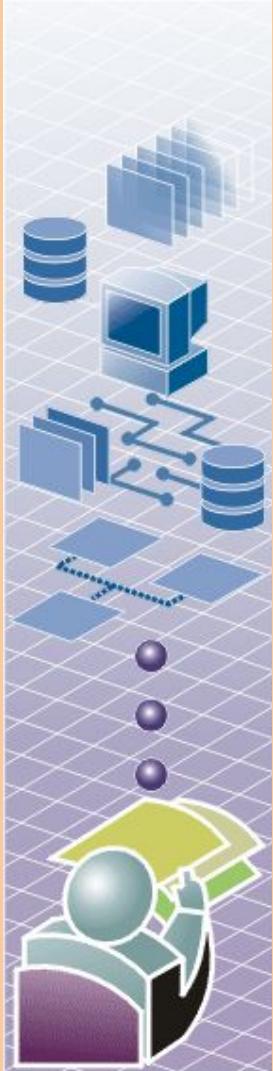
```
public DataSet ModificarCliente(ENCliente cli, int i)
{
    DataSet bdvirtual = new DataSet();
    SqlConnection c = new SqlConnection(s); SqlDataAdapter da = new
    SqlDataAdapter("select * from Cliente", c);
    da.Fill(bdvirtual, "cliente");
    DataTable t = new DataTable();
    t = bdvirtual.Tables["cliente"];

    t.Rows[i]["usuario"] = cli.Usuario;
    t.Rows[i]["contraseña"] = cli.Contraseña;

    SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
    da.Update(bdvirtual, "cliente");

    return bdvirtual;
}
```

Ejercicio



Ejercicio

- Modificar el formulario anterior para eliminar los datos de una persona en la BD.

5 min



AutoGenerateDeleteButton=true

```
protected void GridView2_RowDeleting(object sender,  
GridViewDeleteEventArgs e)  
{  
  
}  
}
```

EN

```
public DataSet BorrarCliente(int i)
```

```
{
```

```
}
```

CAD

```
public DataSet BorrarCliente(ENCliente cli, int i)
{
    DataSet bdvirtual = new DataSet();

    ...
    SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
    da.Update(bdvirtual, "cliente");

    return bdvirtual;
}
```

Paginación en GridView

- Propiedades
 - AllowPaging = true
 - PageSize = 5 (numero de elementos por página)
- Al mostrar los datos con asistente no hay que escribir código, pero al enlazarlo nosotros necesitamos escribir el código para el evento

```
protected void GridView2_PageIndexChanging(object sender,  
    GridViewEventArgs e)  
{  
    d = enl.listarClientesD();  
    GridView2PageIndex = e.NewPageIndex;  
    GridView2.DataSource = d;  
    GridView2.DataBind();  
}
```

GridView: editar columnas

Campos

Campos disponibles:

- (Todos los campos)
- BoundField
 - DNI
 - Nombre
 - Apellidos
- Campo CheckBox
- Campo HyperLink
- ImageField

Agregar

Campos seleccionados:

- Seleccionar
- Editar, Actualizar, Cancelar
- Eliminar
- DNI
- Nombre
- Apellidos

Propiedades de BoundField:

Accesibilidad	
AccessibleHeaderText	
Apariencia	
FooterText	
HeaderImageUrl	
HeaderText	Nombre
Comportamiento	
ApplyFormatInEditMode	False
ConvertEmptyStringToNull	True
HtmlEncode	True
HtmlEncodeFormatString	True
InsertVisible	True
NullDisplayText	
ReadOnly	False

GridView

- Tipos de columnas:
 - **BoundField**: Muestra el texto de un campo de la BBDD
 - **ButtonField**: Muestra un botón para cada ítem
 - **CheckBoxField**: Muestra un checkbox para cada ítem
 - **CommandField**: Proporciona funciones de selección, edición y borrado
 - **HyperLinkField**: Muestra el texto de un campo de la BBDD como un hipervínculo
 - **ImageField**: Muestra una imagen
 - **TemplateField**: Permite especificar múltiples campos y controles personalizados

GridView

- Modificamos el aspecto del GridView

The screenshot shows the Visual Studio IDE interface. On the left, there is a GridView control with five rows of data. The columns are labeled "DNI", "Nombre", and "Apellidos". Each row contains three buttons: "Seleccionar", "Editar", and "Eliminar". The first row is highlighted with a yellow background. At the bottom of the grid, there is a page navigation bar with the numbers "1 2". To the right of the grid, a context menu titled "Tareas de GridView" is open. This menu includes options like "Formato automático...", "Elegir origen de datos: SqlDataSource1", "Configurar origen de datos...", "Actualizar esquema", "Editar columnas...", "Agregar nueva columna...", and several checkboxes for enabling features: "Habilitar paginación" (checked), "Habilitar ordenación" (checked), "Habilitar edición" (checked), "Habilitar eliminación" (checked), "Habilitar selección" (checked), and "Editar plantillas". Two specific items in the menu are highlighted with black boxes and arrows pointing to them from the left: "Formato automático..." and "Editar columnas...".

GridView

- Resultado final

		<u>DNI</u>	<u>Nombre</u>	<u>Apellidos</u>
		Actualizar	Cancelar	
Seleccionar	Editar	Eliminar	22222222	Alberto Lopez
Seleccionar	Editar	Eliminar	44444444	Juan Perez
Seleccionar	Editar	Eliminar	55555555	Sara Jover
Seleccionar	Editar	Eliminar	66666666	Berta Belda

1 2

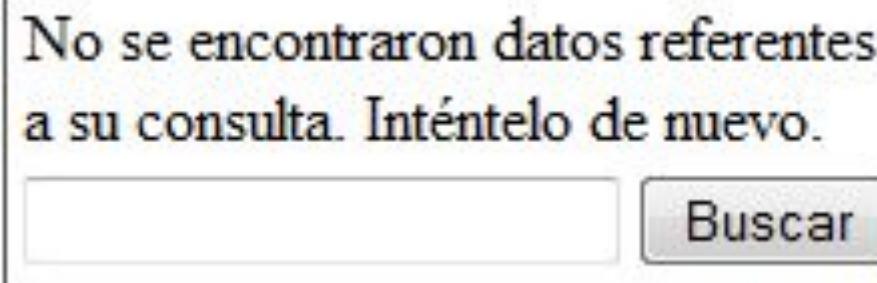
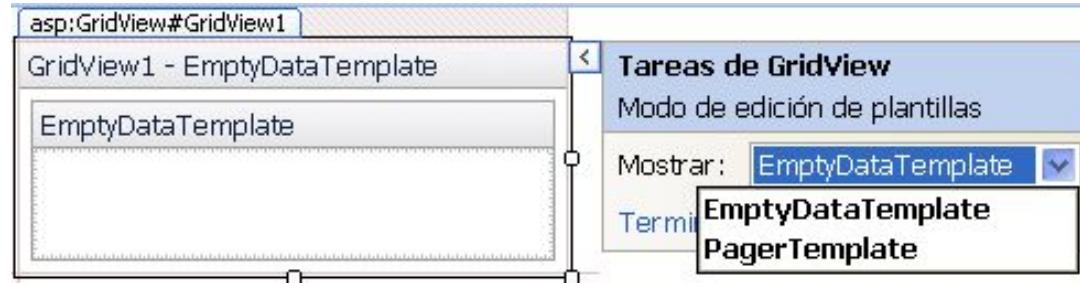
Editar plantillas

- **EmptyDataText**

- Se utiliza para mostrar un mensaje cuando no existen datos que mostrar en el GridView

- **EmptyDataTemplate**

- Podemos personalizar el mensaje mostrado cuando el GridView está vacío



4

Concurrencia

Entorno desconectado: conflictos

- En un entorno desconectado, varios usuarios pueden modificar los datos de los mismos registros al mismo tiempo.
- **Formas de gestión del conflicto:**
 - Conurrencia pesimista.
 - Conurrencia positiva.
 - Last Win.
 - Escribir código para gestionar el conflicto.

Concurrencia

- **Concurrencia pesimista:** Cuando una fila es leída, esta queda bloqueada para su lectura para cualquier otro que la demande hasta que aquel que la posee la libere.

Concurrencia (I)

- **Concurrencia positiva:** Las filas están disponibles para su lectura en todo momento, estas pueden ser leídas por distintos usuarios al mismo tiempo.
- Cuando alguno intenta modificar una fila ya modificada se produce un error y no se modifica.

Concurrencia (II)

- “Last win”: esta técnica implica que no existe control. El último cambio en escribirse es el que permanece.

ADO.NET: Concurrencia positiva

- El objeto DataSet mantiene dos versiones de las filas que leímos:
 - Versión original, idéntica a la leída en la BD
 - Versión actualizada, representa los cambios del usuario
- Cuando se actualiza la fila, se comparan los valores originales con la fila real de la BD, para ver si ha sido modificada.
 - Si ha sido modificada, hay que capturar una excepción
 - Sino, la actualización es efectuada

Evento RowUpdated

- Escribir código en la aplicación que permita a los usuarios determinar qué cambios deberían conservarse. Las soluciones específicas pueden variar dependiendo de los requerimientos de negocio de una determinada aplicación.
- Evento RowUpdated:
 - Al actualizar una fila: después de cada operación pero antes de lanzar cualquier excepción.
 - Podemos examinar los resultados e impedir que se lance una excepción.

5

Conectado vs
Desconectado

Conectado vs Desconectado

- Acceso conectado a datos (conexión viva)
 - **DataReader**
 - Podemos recuperar rápidamente todos los resultados.
 - Utiliza una conexión viva. Más ligero y veloz que DataSet
 - Acceso a los resultados sólo hacia delante de sólo lectura.
 - Mejor rendimiento que DataSet, por lo que es mejor elección para acceso a datos simple.
- Acceso desconectado a datos
 - **DataSet**

Conectado vs Desconectado

- Para realizar consultas de sólo lectura, que únicamente serán necesarias realizarlas una vez (no tendremos que volver a acceder a filas anteriores) el objeto recomendado es **DataReader**.
- *Por ejemplo, para comprobar si un artículo se encuentra entre una tabla que guarda la lista de artículos del inventario de un almacén, basta con realizar una única consulta de sólo lectura.*
- Sin embargo, si vamos a realizar un acceso a datos más complicado, como puede ser la consulta de todos los artículos de diferentes tipos que pertenecen a un proveedor, la elección correcta sería utilizar **DataSet**.

Conectado vs Desconectado

- **Acceso a datos.**
 - Como hemos dicho, si tenemos previsto recibir y almacenar datos, optamos por *DataSet*, ya que *DataReader* sólo permite lecturas.
- **Trabajar con más de una tabla o más de una base de datos.**
 - Si la función que estamos desarrollando requiere información situada en varias tablas de una misma base de datos o de varias, utilizaremos el objeto *DataSet*. Con *DataReader* sólo podemos construir consultas SQL que accedan a una base de datos.

Tema 8. ASP.NET II

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Indice

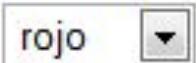
1. Controles de lista sencillos
2. Controles de Navegación
3. Controles de Validación
4. Objetos Session y Application
5. Eventos de Aplicación: Global.asax
6. Cookies
7. Envío de email

1

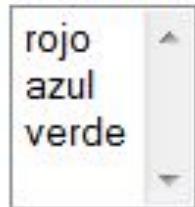
Controles de lista
sencillos

Controles de Lista Sencillos

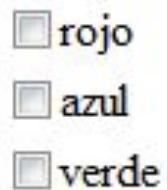
DROPODOWNLIST



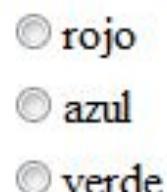
LISTBOX



CHECKBOXLIST



RADIOBUTTON LIST



- Lista desplegable
- Lista desplegada
- Lista de botones de verificación
- Lista de botones de radio

Controles de Lista Sencillos (listBox, dropDownList, radioButtonList, checkBoxList)

- Lista desplegable (dropDownList)



- ElementoLista1 (ListItem) FILA 0
- ElementoLista2 **Seleccionado** (ListItem) FILA 1
- ElementoLista3 (ListItem) FILA 2

Añadir elementos de forma declarativa

```
<asp:DropDownList ID="DropDownList1" runat="server"  
onselectedindexchanged="DropDownList1_SelectedIndexChanged"  
Width="90px" AutoPostBack="False">  
    <asp:ListItem Value="rojo1">rojo</asp:ListItem>  
    <asp:ListItem>azul</asp:ListItem>  
    <asp:ListItem>verde</asp:ListItem>  
</asp:DropDownList>
```

RepeatDirection

- RadioButtonList: lista de botones de radio

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server"  
    onselectedindexchanged="RadioButtonList1_SelectedIndexChanged">  
    <asp:ListItem>rojo</asp:ListItem>  
    <asp:ListItem>azul</asp:ListItem>  
    <asp:ListItem>verde</asp:ListItem>  
</asp:RadioButtonList>
```

- CheckBoxList: lista de opciones múltiple

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server"  
    onselectedindexchanged="CheckBoxList1_SelectedIndexChanged"  
    RepeatDirection="Horizontal">  
    <asp:ListItem>rojo</asp:ListItem>  
    <asp:ListItem>azul</asp:ListItem>  
    <asp:ListItem>verde</asp:ListItem>  
</asp:CheckBoxList>
```

Propiedad Items: colección de ListItems

Objeto ListItem: propiedades

The screenshot shows the Visual Studio IDE interface. On the left, there is a 'Miembros:' (Members) list containing three items: '0 rojo', '1 azul', and '2 verde'. The item '0 rojo' is currently selected, highlighted with a blue background. To the right of this list are two buttons: an upward arrow and a downward arrow. On the far right, the 'Propiedades de rojo:' (Properties of rojo) window is open. It displays a list of properties for the selected item:

Nombre	Valor
Enabled	True
Selected	False
Text	rojo
Value	rojo1

- Text:
 - contenido visualizado
- Value:
 - valor oculto del código HTML
- Selected:
 - true o false (seleccionado o no)

Propiedades de los Controles de Lista Sencillos

- Propiedad SelectedIndex
 - Indica la fila seleccionada como un índice que empieza en cero
- Propiedad SelectedItem
 - Permite que el código recupere un objeto **ListItem** que representa el elemento seleccionado

```
Label1.Text = DropDownList1.SelectedIndex.ToString();
```

```
Label1.Text = DropDownList1.SelectedItem.Text.ToString();
```

```
Label1.Text = DropDownList1.SelectedItem.Value.ToString();
```

```
Label1.Text=DropDownList1.SelectedValue;
```

Controles de lista con selección múltiple

- **ListBox:**
 - Pueden seleccionarse varios elementos: propiedad **SelectionMode=Multiple**
- **CheckBoxList**
 - Siempre pueden seleccionarse varios elementos
- **Hacer:** Para encontrar todos los elementos seleccionados necesitamos
 - recorrer la colección Items del control lista
 - comprobar la propiedad ListItem.Selected de cada elemento

```
Foreach ListItem i in DropDownList1.Items  
{ if (i.Selected==true)  
... }
```

Añadir elementos dinámicamente a una lista (código)

- Método *Add* del objeto *Items* del control

EJEMPLO:

```
DropDownList1.Items.Add("rojo");
```

```
DropDownList1.Items.Add(new ListItem("rojo", "Red"));
```



```
    text → "rojo"  
    value → "Red"
```

2

Controles de
navegación:
Menu

Control menu

- Se puede utilizar para crear un menú que podemos colocar en la **página maestra**.
- Permite añadir un **menú principal con submenús** y también nos permite definir menús dinámicos.
- Los **elementos del Menu** pueden añadirse directamente en el control o enlazarlos con una fuente de datos.
- En las propiedades podemos especificar la **apariencia, orientación y contenido del menú**.



Static Display / Dynamic Display

- El control tiene dos modos de “Display”:
 - **Estático (static)**: El control Menu está **expandido completamente todo el tiempo**. Toda la estructura es visible y el usuario puede hacer click en cualquier parte.
 - **Dinámico (dynamic)**: En este caso solo son estáticas las porciones especificadas, mientras que **los elementos hijos se muestran cuando el usuario mantiene el puntero del ratón sobre el nodo padre**.

Propiedades

- El Control **Menu** es una colección de **MenuItem**s
- Propiedades del control → Colección **Items** (colección de objetos **MenuItem**)
- Añadir objetos individuales **MenuItem** (de forma declarativa o programática) .

Ejemplo

```
<asp:menu id="NavigationMenu" orientation="Vertical" runat="server">
    <items>
        <asp:menuitem navigateurl="Home.aspx"
            text="Home"
            tooltip="Home">
            <asp:menuitem navigateurl="Music.aspx"
                text="Music"
                tooltip="Music">
                <asp:menuitem navigateurl="Classical.aspx"
                    text="Classical"
                    tooltip="Classical"/>
                <asp:menuitem navigateurl="Rock.aspx"
                    text="Rock"
                    tooltip="Rock"/>
                <asp:menuitem navigateurl="Jazz.aspx"
                    text="Jazz"
                    tooltip="Jazz"/>
            </asp:menuitem>
        </asp:menuitem>
    </items>
</asp:menu>
```

Otra forma, navegar desde el código C#

```
<asp:Menu ID="Menu1" runat="server" Orientation="Vertical">
  <Items>
    <asp:MenuItem Text="File" Value="File">
      <asp:MenuItem Text="New" Value="New"></asp:MenuItem>
      <asp:MenuItem Text="Open" Value="Open"></asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem Text="Edit" Value="Edit">
      <asp:MenuItem Text="Copy" Value="Copy"></asp:MenuItem>
      <asp:MenuItem Text="Paste" Value="Paste"></asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem Text="View" Value="View">
      <asp:MenuItem Text="Normal" Value="Normal"></asp:MenuItem>
      <asp:MenuItem Text="Preview" Value="Preview"></asp:MenuItem>
    </asp:MenuItem>
  </Items>
</asp:Menu>
```

Code-Behind C#

```
protected void Menu1_MenuItemClick(object sender,
    System.Web.UI.WebControls.MenuEventArgs e)
{
    switch(e.Item.Value)
    {
        case "File":
            ...
            return;
        case "Open":
            ...
            return;
    }
}
```

[http://msdn.microsoft.com/en-us/library/16yk5dby\(v=vs.80\).aspx](http://msdn.microsoft.com/en-us/library/16yk5dby(v=vs.80).aspx)

http://www.obout.com/em/doc_server.aspx

3

Controles de Validación

Validación de datos

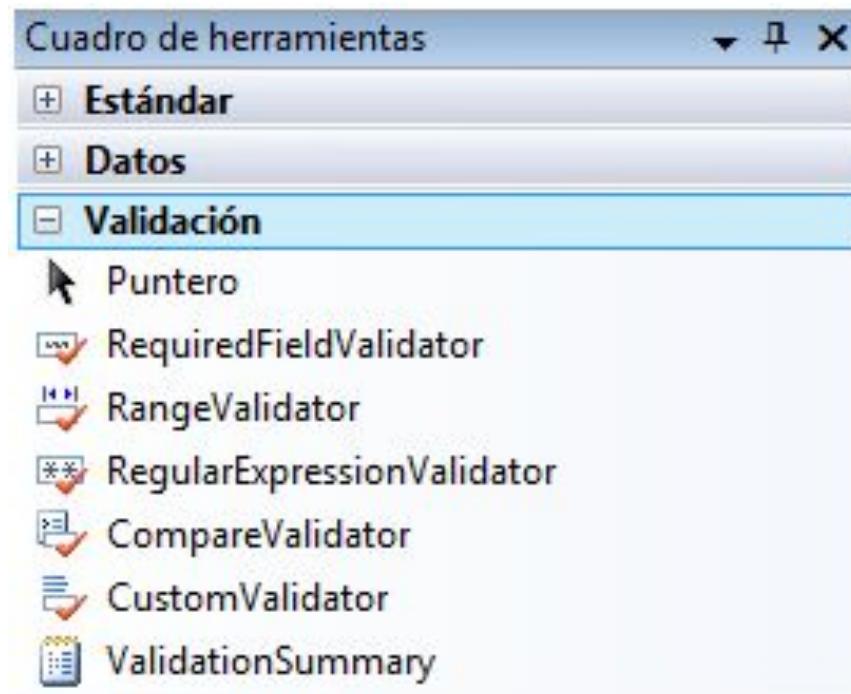
- Debemos asegurar que los usuarios introducen datos correctamente
 - Dirección de email
 - Número de teléfono
- ASP.Net proporciona un conjunto de controles de validación predefinidos
- Dos tipos de validación
 - Validación del lado del cliente
 - Validación del lado del servidor

Validación de datos

- **Lado del cliente:**
 - Utilización de código **JavaScript** que valida los datos introducidos por el usuario, directamente en el navegador
- **Lado del servidor:**
 - Utilización de código (**C#**) para validar los datos de formularios una vez han sido enviados al servidor

Controles de validación

- ASP.Net detecta si el navegador soporta validación del lado del cliente:
 - Generan el código JavaScript necesario para validar los datos
 - En otro caso, los datos del formulario se validan sólo en el servidor



▲ Validación
Puntero
CompareValidator
CustomValidator
RangeValidator
RegularExpressionValidator
RequiredFieldValidator
ValidationSummary

Controles de validación

RequiredFieldValidator	Dato de entrada requerido para el usuario
CompareValidator	Compara los datos proporcionados por el usuario con un valor constante, o con el valor de otro control (mediante un operador de comparación como menor que, igual que o mayor que) para un tipo de datos específico.
RangeValidator	Comprueba que una entrada de usuario está entre los límites superior e inferior especificados. Se pueden comprobar los intervalos entre pares de números, caracteres alfabéticos y fechas
RegularExpressionValidator	Comprueba que la entrada del usuario coincide con un modelo definido por una expresión regular. Este tipo de validación permite comprobar secuencias de caracteres predecibles, como los que aparecen en las direcciones de correo electrónico, números de teléfono, códigos postales, etc.
CustomValidator	Comprueba la entrada de usuario utilizando la validación lógica que ha escrito. Este tipo de validación permite comprobar valores derivados en tiempo de ejecución

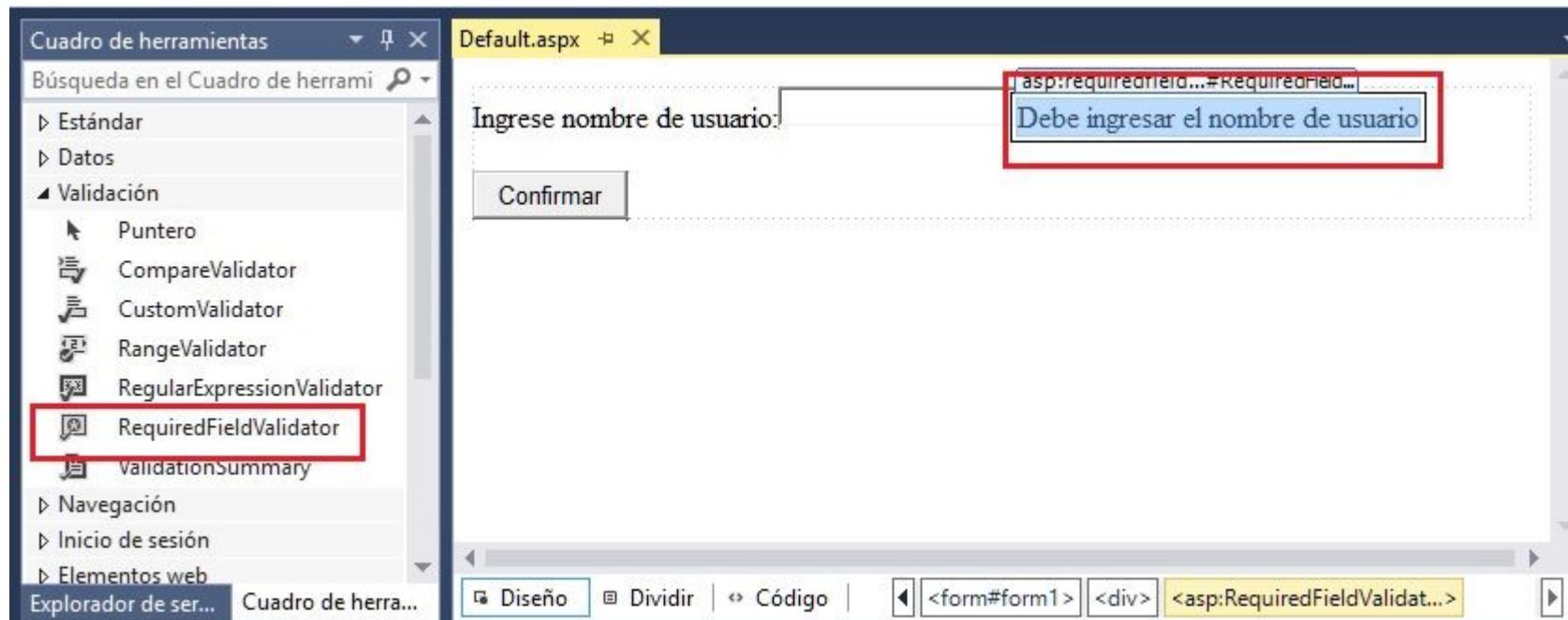
Controles de Validación

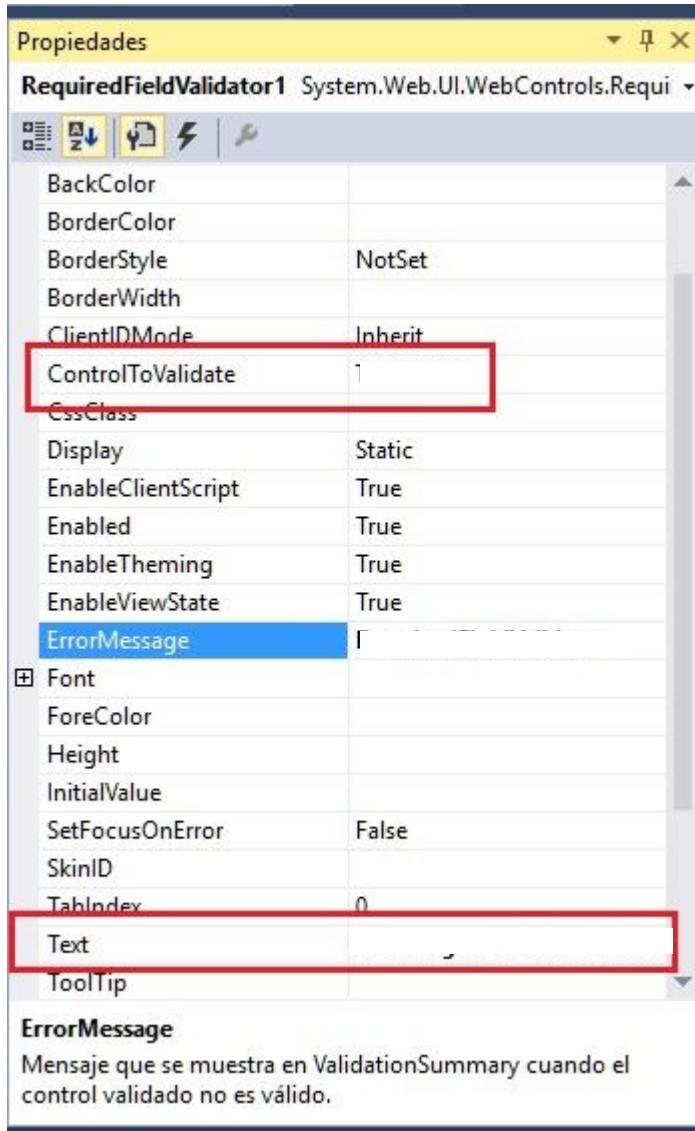
- **Propiedades comunes a todos los controles de validación:**
 - Para mostrar el mensaje de error **ErrorMessage**
 - Para indicar el control a validar **ControlToValidate**
 - Para mostrar texto inicial **Text**
 - **IsValid:** si ha pasado positivamente la validación
- **Propiedad de la página IsValid**

 Será true si se pasan las validaciones de la página positivamente

Entrada requerida

- **Entrada requerida:** RequiredFieldValidator: La validación es OK cuando el control de entrada no contiene una cadena vacía.



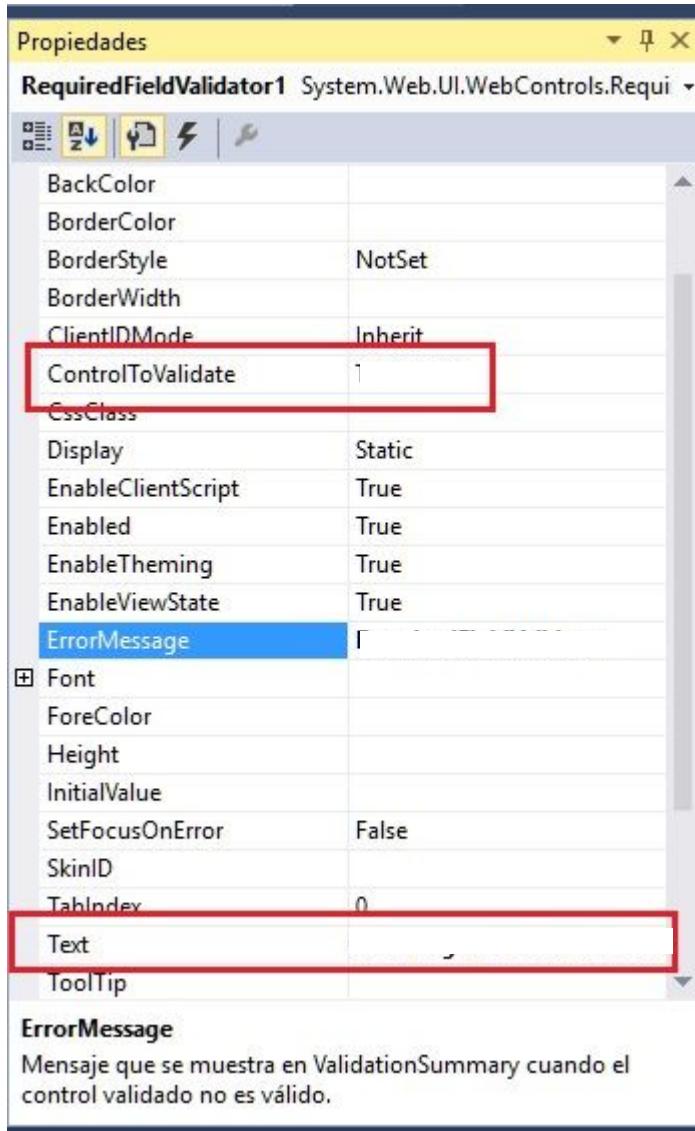


¿Qué ponemos en las propiedades?

ControlToValidate

ErrorMessage

Text



¿Qué ponemos en las propiedades?

ControlToValidate=
TextBox1

ErrorMessage= Debe introducir nombre usuario

Text= Debe introducir nombre usuario

Código Button1_Click

Si queremos que al presionar el botón se redireccione a la página Default2.aspx en caso de haber ingresado un nombre de usuario

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
        Response.Redirect("Default2.aspx");
}
```

Controles de validación

```
<form id="form1" runat="server">
  <div>
    Usuario: <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="UserNameReq" runat="server"
      ControlToValidate="TextBox1" ErrorMessage="Introduce el usuario!!">
    </asp:RequiredFieldValidator>
    Password: <asp:TextBox ID="TextBox2" runat="server"> </asp:TextBox>
    <asp:RequiredFieldValidator ID="PasswordReq" runat="server"
      ControlToValidate="TextBox2" ErrorMessage="Introduce el
      password!!"></asp:RequiredFieldValidator>
    <asp:Button ID="Button1" runat="server" Text="Enviar" />
  </div>
</form>
```

Usuario: Introduce el usuario!!

Password: Introduce el password!!

Comprobación de intervalo

- **Comprobación de intervalo:** RangeValidator: La validación es OK cuando el control de entrada contiene un valor dentro de un intervalo numérico, alfabético o temporal especificado.
 - **MaximumValue**
 - **MinimumValue**
 - **Type**
- Definir un control RangeValidator que compruebe si un número está en el intervalo numérico 10-100

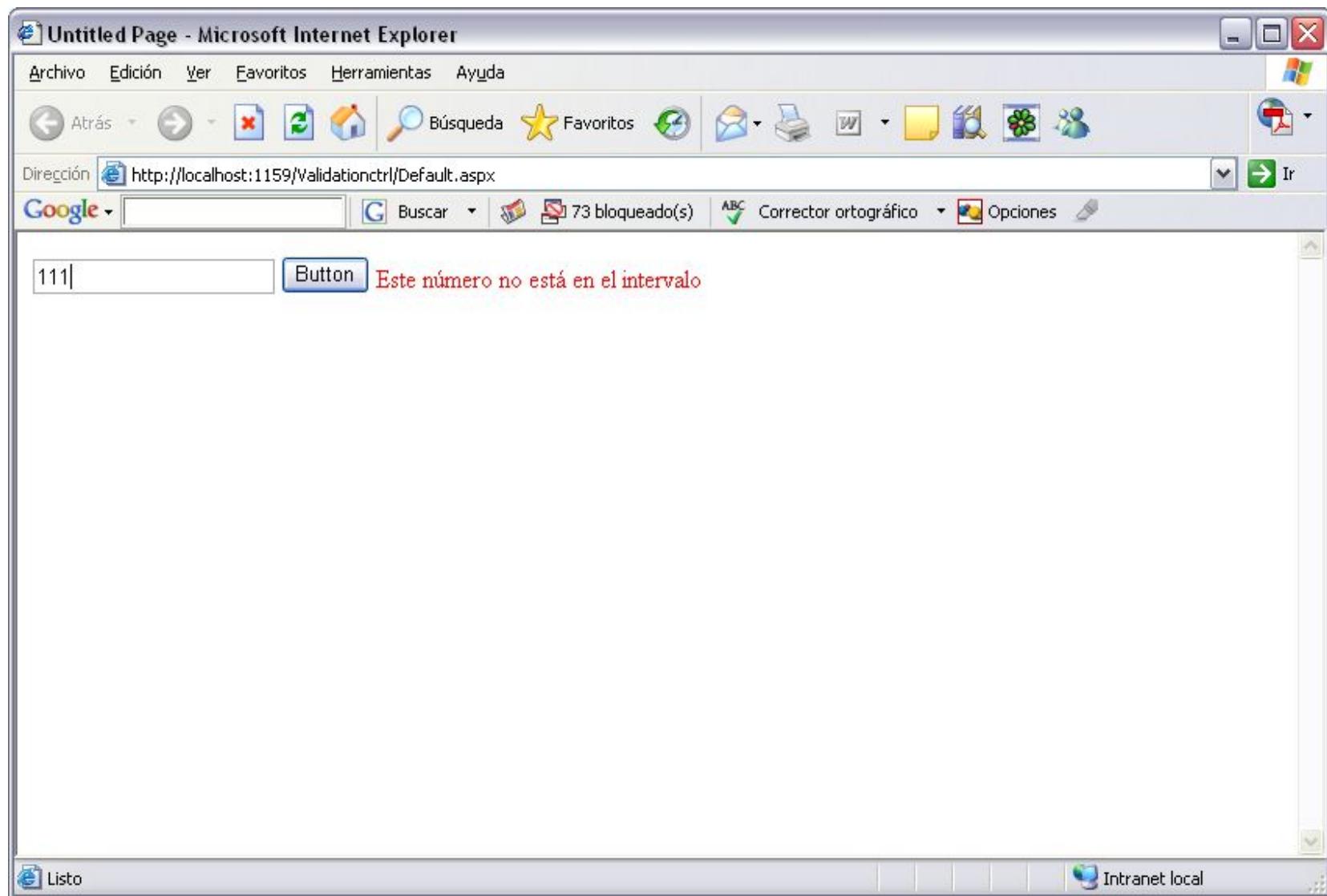
Properties

RangeValidator1 System.Web.UI.WebControls.

(Expressions)	
(ID)	RangeValidator1
AccessKey	
BackColor	<input type="color"/>
BorderColor	<input type="color"/>
BorderStyle	NotSet
BorderWidth	
ControlToValidate	TextBox1
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	Este número no está
+ Font	
ForeColor	<input style="background-color: red; color: red; width: 15px; height: 15px;" type="color"/> Red
Height	
MaximumValue	100
MinimumValue	10
SetFocusOnError	True
SkinID	
TabIndex	0
Text	
ToolTip	
Type	Integer

```
<asp:RangeValidator ID="RangeValidator1"
    runat="server"
    ControlToValidate="TextBox1"
    ErrorMessage="Este número no está en
    el intervalo" MaximumValue="100"
    MinimumValue="10" Type="Integer">
</asp:RangeValidator>
```

Vista ejecución (range validator)



Comparación

- **Comparación con un valor/control:** CompareValidator: La validación es OK si el control contiene un valor que se corresponde con el valor de otro control especificado.
 - ControlToCompare / ValueToCompare
 - ControlToValidate
 - Type
 - Operator

Ejercicio

- ControlToCompare=
- ValueToCompare=
- ControlToValidate=
- Type=
- Operator=

Default3.aspx*

Nombre de usuario:

Clave

Repita clave

asp:comparevalidator#CompareValida...

Las claves ingresadas son distintas

Confirmar

The screenshot shows a web form titled "Default3.aspx*". It contains three text input fields: "Nombre de usuario:", "Clave", and "Repita clave". Below the "Repita clave" field is a CompareValidator control, indicated by the text "asp:comparevalidator#CompareValida...". A tooltip or validation message box is displayed over the validator, stating "Las claves ingresadas son distintas" (The entered keys are different). At the bottom of the form is a "Confirmar" button.

Solución

- ControlToCompare= TextBox2
- ValueToCompare=
- ControlToValidate= TextBox3
- Type= String
- Operator= Equal

The screenshot shows a web form titled "Default3.aspx". The form contains three text input fields: "Nombre de usuario", "Clave", and "Repita clave". Below the "Repita clave" field is a validation control, specifically an "asp:CompareValidator". A tooltip for this control displays the message "Las claves ingresadas son distintas" (The entered keys are different). A "Confirmar" button is located at the bottom of the form.

Coincidencia de modelos

- **Coincidencia de modelos:** RegularExpressionValidator: La validación es OK si el valor de un control de entrada se corresponde con una expresión regular especificada.
 - ValidationExpression

El control **RegularExpressionValidator** permite validar el valor de un campo de un formulario **con un patrón específico**, por ejemplo un **código postal, un número telefónico, una dirección de mail, una URL** etc.

Expresiones regulares

- **Dirección de correo electrónico**
 - Comprobar que existe una @, un punto y sólo permite caracteres que no sean espacios.
- **Contraseña**
 - Entre 4 y 10 caracteres y el primer carácter debe ser una letra.
- **Número de cuenta**
 - Secuencia de 4, 4, 2, y 10 dígitos, cada grupo separado por un guión.
- **Campo de longitud limitada**
 - Entre 4 y 10 caracteres incluyendo caracteres especiales (*, &...)

Sintaxis Expresiones Regulares

- * cero o más ocurrencias del carácter o subexpresión anterior.
- + una o más ocurrencias del carácter o subexpresión anterior
- () agrupa una subexpresión que se trata como un único elemento
- | Cualquiera de las dos partes (OR)
- [] se corresponde con un carácter en un intervalo de caracteres válidos [a-c]
- {n} exactamente n de los caracteres o subexpresiones anteriores
- . cualquier carácter excepto el salto de línea
- ? el carácter anterior o la subexpresión anterior es opcional
- ^ comienzo de una cadena
- \$ fin de una cadena

...

- \s carácter de espacio en blanco (ej. tab o espacio)
- \S cualquier carácter no espacio
- \d cualquier carácter numérico
- \D cualquier carácter no dígito
- \w cualquier carácter alfanumérico (letra, número o carácter de subrayado)

Solución...

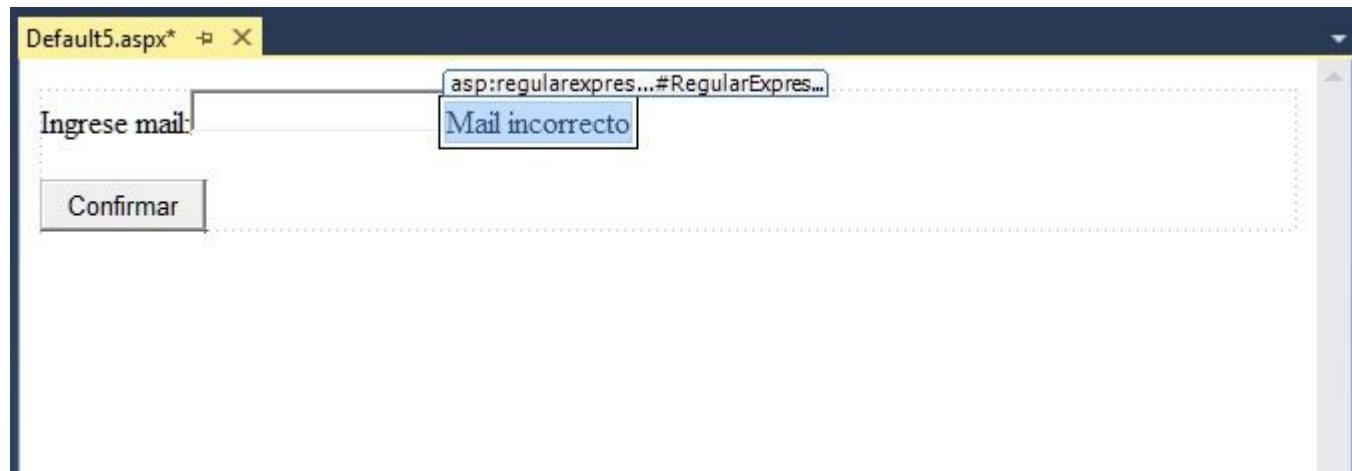
- Correo electrónico
 - `\S+@\S+\.\S+`
- Contraseña
 - `[a-zA-Z]\w{3,9}`
- Número de cuenta
 - `\d{4}-\d{4}-\d{2}-\d{10}`
- Campo de longitud limitada
 - `\S{4,10}`

Propiedades

ControlToValidate=

ErrorMessage=

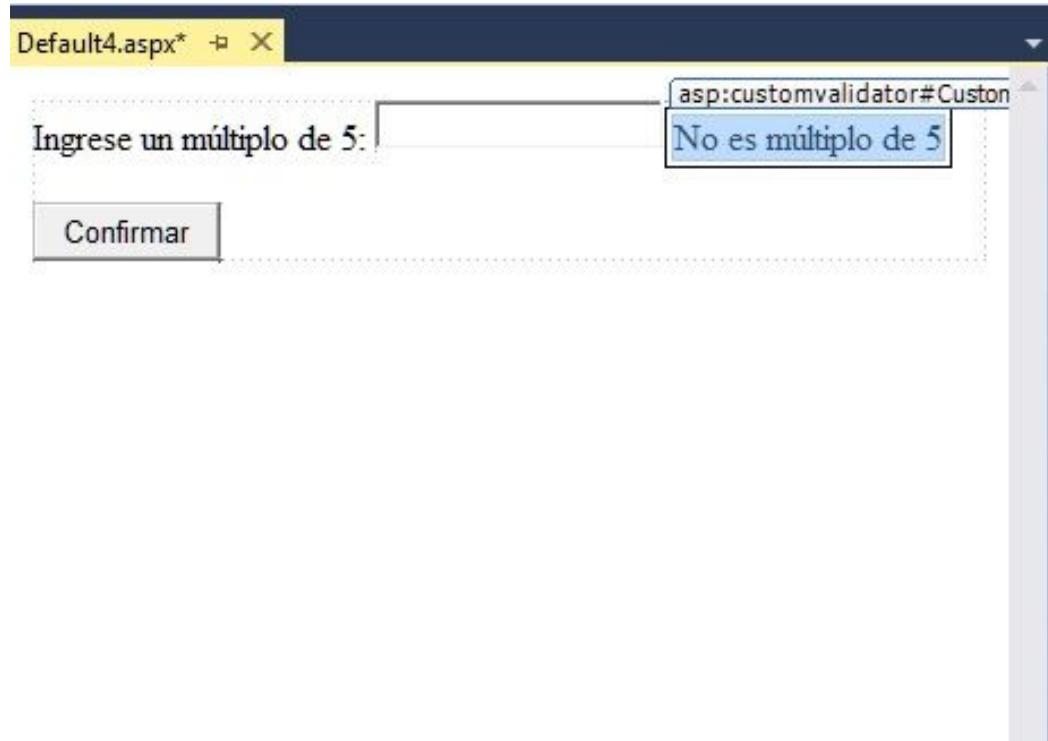
ValidationExpression=



Controles de Validación

- **CustomValidator:** La validación la realiza una función definida por el usuario.
 - ClientValidationFunction
 - OnServerValidate
- El control **CustomValidator** permite validar el campo de un formulario con una función de validación propia.
- Debemos asociar nuestro control **CustomValidator** con un evento propio.

Ejemplo

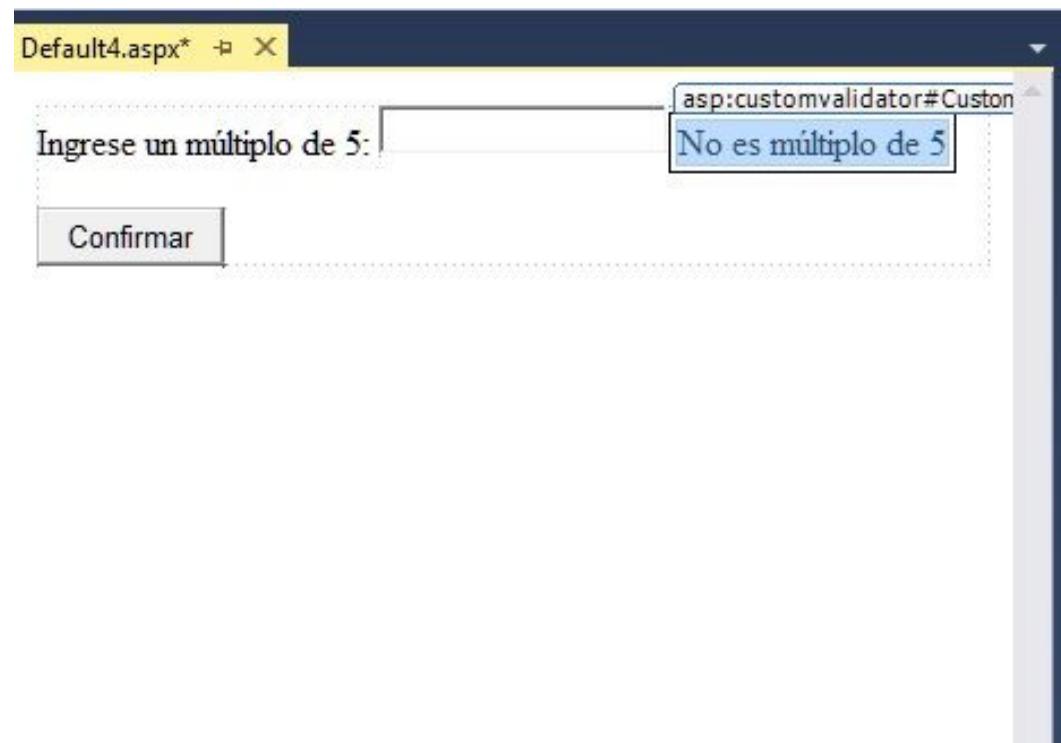


página que solicite el ingreso de un número múltiplo de 5, en caso de ingresar un valor incorrecto mostraremos un mensaje de error.

Propiedades

ControlToValidate=

ErrorMessage=



C#

```
protected void CustomValidator1_ServerValidate(object source,  
ServerValidateEventArgs args)  
{  
    int valor;  
    valor = int.Parse(TextBox1.Text);  
    if (valor % 5 == 0)  
        args.IsValid = true;  
    else  
        args.IsValid = false;  
}
```

mediante el operador %
(resto de una división)
verificamos si es cero

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        Response.Redirect("Default5.aspx");
    }
}
```

Controles de Validación

- **ValidationSummary**: Este control muestra un resumen con todos los mensajes de error de cada control de validación.
 - ShowSummary

Ejemplo

La propiedad **Text** de los objetos **RequiredFieldValidator** las inicializamos con un (*) asterisco y las propiedades **ErrorMessage** con las cadenas: *Debe ingresar el nombre de usuario* y *Debe ingresar la clave* respectivamente

Default6.aspx

(body)

Usuario: *

Clave: *

Confirmar

- Mensaje de error 1.
- Mensaje de error 2.

localhost:65278/Default6.aspx

localhost:65278/Default6.aspx

Usuario: *

Clave: *

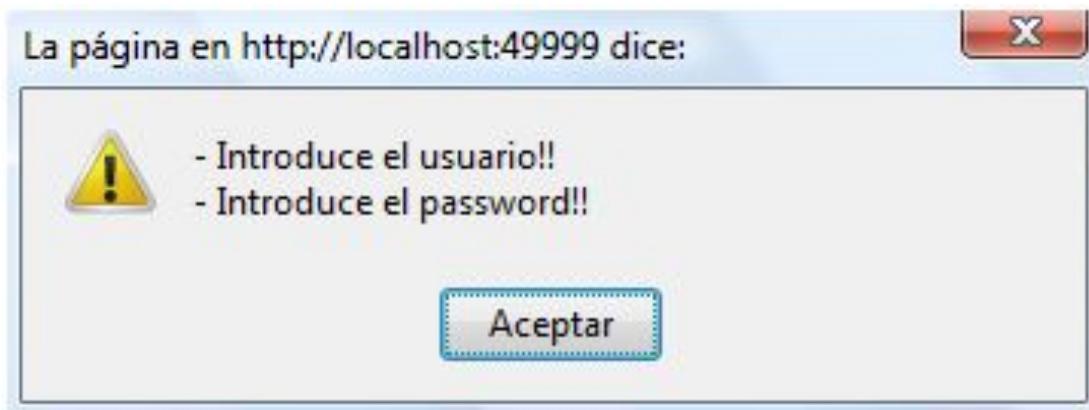
Confirmar

- Debe ingresar el nombre de usuario
- Debe ingresar la clave

ValidationSummary propiedades

- ShowMessageBox:

muestra un cuadro
de diálogo resumen



El proceso de validación (servidor)

1. El usuario recibe una página y comienza a llenar los valores de entrada. Al final el usuario pulsa un botón para enviar la página.
2. Cada control Button tiene una propiedad **Causes Validation**.
 - Si esta propiedad es **False**, ASP.NET ignora los controles de validación.
 - Si está a **True** (Valor Predeterminado), ASP.NET valida automáticamente la página cuando el usuario pulsa el botón. Se realiza la validación de cada control de la página.
 - Cada control de validación expone su propiedad **IsValid**, La página también expone una propiedad **IsValid** que resume el estado **IsValid** de todos los controles de validación de la página.

Añadir a Web.config

```
<appSettings>

    <add
        key="ValidationSettings:UnobtrusiveValidationMode"
        value="None" />

</appSettings>
```

4

Mantenimiento de
estado:
Objetos Session y
Application

Objetos Session y Application

- Los objetos **Session** están asociados a un usuario particular y sirven como manera de transportar y mantener los datos del usuario en páginas web, como foros o sitios de comercio electrónico.
- Los objetos **Application** son compartidos por todos los usuarios y permiten almacenar información compartida por toda la aplicación web.
- En ASP.NET los objetos **Session** y **Application** están implementados como colecciones o **conjuntos de pares nombre–valor**.

Qué es una sesión?

- Una **sesión** es el período de tiempo en el que un usuario particular interactúa con una aplicación web.
- Durante una sesión la identidad única de un usuario se mantiene internamente.
- Los datos se almacenan temporalmente en el servidor.
- Una sesión finaliza si hay un *timeout* o si tú finalizas la sesión del visitante en el código.

Cuál es el uso de una sesión?

- Las sesiones ayudan a preservar los datos entre accesos sucesivos. Esto puede hacerse gracias a los objetos de sesión.
- **Los objetos de Sesión** nos permiten preservar las preferencias del usuario y otra información del usuario al navegar por la aplicación web.
- Ejemplo
- Website de comercio electrónico donde el visitante navega a través de muchas páginas y quiere seguir qué productos ha adquirido.

Objeto Session

- **Session:** sirve para almacenar datos pertenecientes a un único usuario (en el ámbito de una sesión).

```
//Borra todos los valores de estado de la sesión  
Session.Clear();  
Session.Add("nombre","Homer");  
Response.Write(Session["nombre"]);
```

En ASP.NET

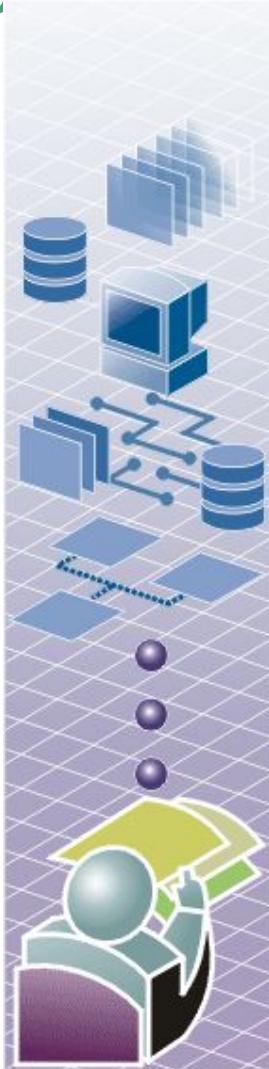
- Las sesiones son tablas Hash en memoria con un timeout especificado.
- Session[“username”] = “Jose Martínez”; Session[“color”] = “Blue”;
- Asignamos los valores a las variables de sesión “username” y “color”, respectivamente. Si necesito saber el “username” o “color” en páginas siguientes puedo usar Session[“username”], Session[“color”].
- Las sesiones en ASP.NET están identificadas usando enteros 32-bit long conocidos como Session IDs. El motor ASP genera estos session ID's de tal forma que se garantice que son únicos

Objeto Session

Session Type	Qué hace	Ejemplo
Session.Abandon	Abandona (cancela) la sesión actual	
Session.Remove	Borra un elemento de la colección de estado de la sesión.	Session[“username”] = “Jose Martínez”; (Inicializa una variable de sesión) Session.Remove[“username”]; (Borra la variable de sesión “username”)
Session.RemoveAll	Borra todos los elementos de estado de la sesión.	

Session Type	Qué hace	Ejemplo
Session.Timeout	Establece el timeout (<i>en minutos</i>) para una sesión	Session.Timeout=30 (Si un usuario NO pide una página en la aplicación ASP.NET en 30 minutos la sesión expira.)
Session.SessionID	Recupera el ID de la sesión (propiedad de sólo lectura de una sesión) para la sesión.	
Session.IsNewSession	Es para comprobar que la sesión del usuario se creó con la petición actual p.ej. el usuario acaba de entrar al sitio web. La propiedad IsNewSession es cierta en la primera página de la aplicación.	

Ejercicio



Ejercicio

- Crear una aplicación web en la cual pidamos introducir un nombre de usuario y un botón enviar.
- Al pinchar en el botón que nos redireccione a un segundo formulario en el cual pondremos “hola “ seguido del login introducido:
 - Metiendo el login en una variable de sesión

Código

Default.aspx.cs

```
protected void Button1_Click (object sender, EventArgs e)
{
}
```

session2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
}
```

Botón con SESSION

Archivo Default.aspx.cs

```
protected void Button1_Click (object sender, EventArgs e)
{
    Session["login"] = TextBox1.Text;
    Response.Redirect("session2.aspx");
}
```

Archivo session2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    Label3.Text = Session["Login"].ToString();
}
```

Importante

Al crear la parte privada de la Web utilizaremos variables de sesión para controlar si el usuario ha entrado logueándose o ha entrado poniendo directamente la URL, en cuyo caso la variable de sesión estará vacía y no deberíamos permitir el acceso

Variables de aplicación

- Y si queremos inicializar variables que estén disponibles en una sesión y sean las mismas para todos los usuarios??
- Esto supone que un cambio en el valor de una **variable de aplicación** se refleja en las sesiones actuales de todos los usuarios.

Objeto Application

- Por ejemplo:
 - Se puede dar un valor a una variable de aplicación llamada SiteName

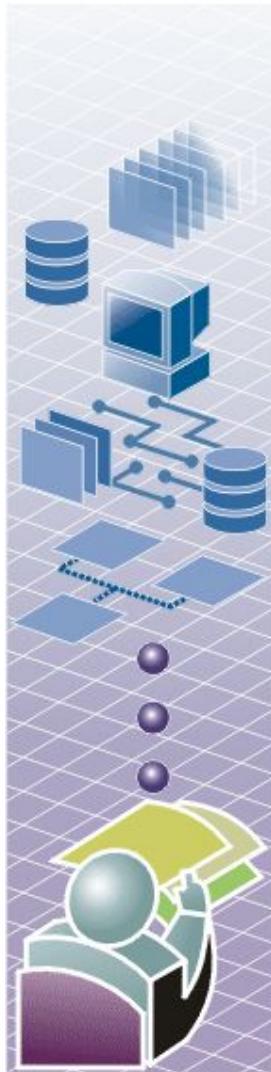
```
Application["SiteName"] = "Mi aplicación";
```
 - Cualquier página de la aplicación puede leer esa cadena:

```
string appName = (string)Application["SiteName"];
```
- Para eliminar cualquier variable del objeto Application:

```
Application.Remove("SiteName");
```
- Para eliminar todas las variables:

```
Application.RemoveAll();
```

Variables de aplicación



Ejercicio

- Crear una aplicación web que cuente el número de visitas que recibe
 - Utilizar variables de aplicación
 - Cuando llegue a 10 visitas el contador se debe reiniciar

Variables de aplicación

- Primer paso:
 - En la página Default.aspx incluimos una etiqueta
`<asp:Label ID="LabelCont" runat="server"></asp:Label>`
- Segundo paso:
 - En el archivo Default.aspx.cs (code behind) utilizar una variable Application para controlar el número de visitas

```
protected void Page_Load(object sender, EventArgs e) {
    if(Application["PageCounter"] != null && (int)Application["PageCounter"] >= 10)
    { Application.Remove("PageCounter"); }
    if(Application["PageCounter"] == null)
    { Application["PageCounter"] = 1; }
    else
    { Application["PageCounter"] =
        (int)Application["PageCounter"] + 1; }
    LabelCont.Text = Application["PageCounter"].ToString();
}
```

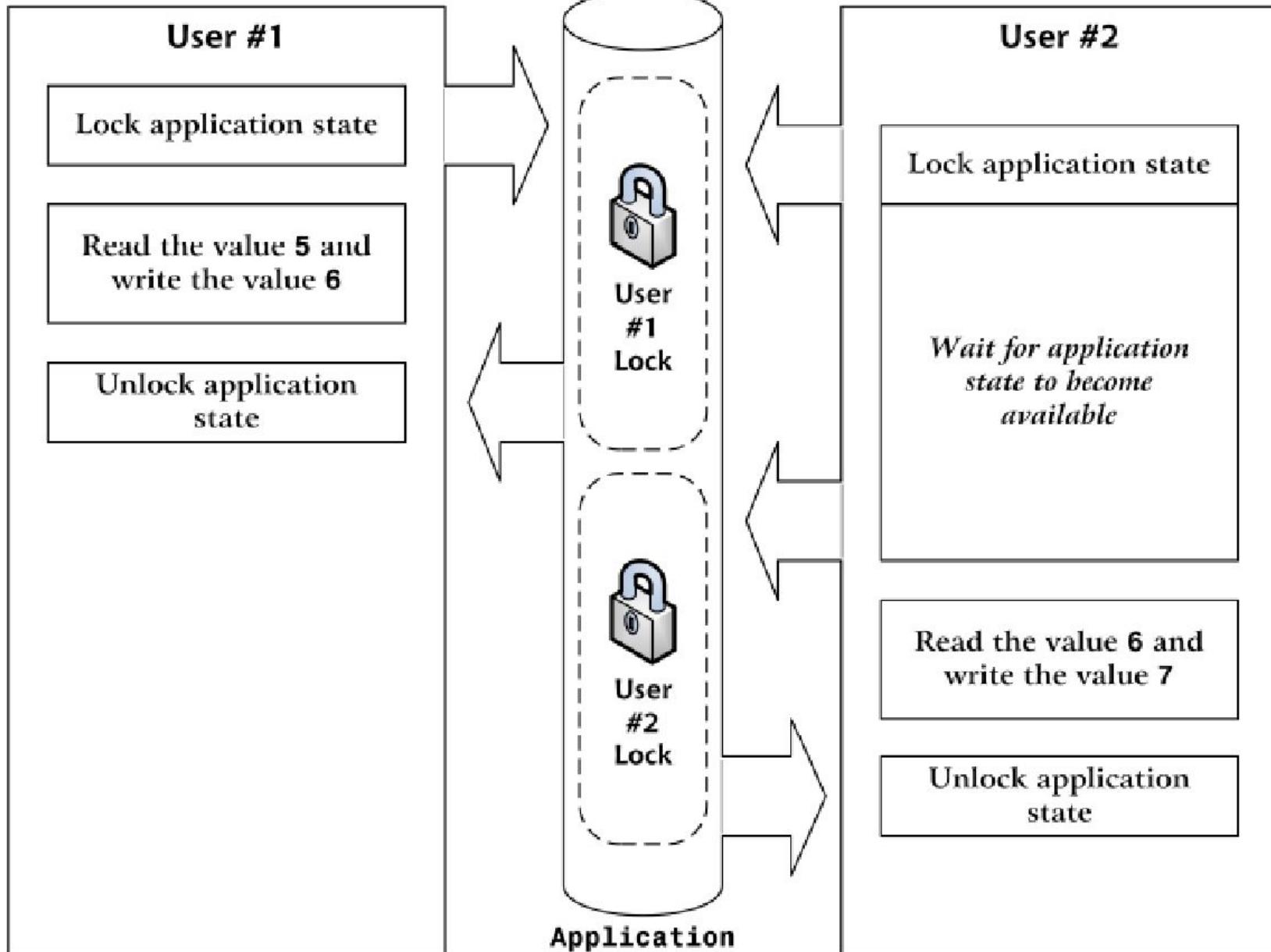
Variables de aplicación

- Problema:
 - Dos personas cargan simultáneamente la página:
 - El contador podría incrementarse sólo 1 unidad
$$\text{Application}["\text{PageCounter}"] = (\text{int})\text{Application}["\text{PageCounter}"] + 1$$
 - La expresión de la derecha se evalúa primero
 - El usuario1 lee el valor de PageCounter almacenado en la aplicación
 - El usuario2 lee el valor de PageCounter almacenado en la aplicación
 - Ambos le suman 1 unidad pero el incremento final en lugar de ser 2 unidades es sólo 1

Actualizar una variable de aplicación

- Solución:
 - En un instante concreto, varias sesiones pueden estar intentando cambiar el valor, aunque sólo una sesión estará autorizada a cambiarlo.
 - ASP.NET tiene exclusión mutua para este tipo de problemas:
 - `Application.Lock()` – Bloquea las variables de aplicación
 - `Application.Unlock()` – Desbloquea las variables de aplicación
 - Una vez que las variables están bloqueadas las sesiones que intentan cambiarlas tienen que esperar.

Variables de aplicación



Objeto Application

- **Application:** proporciona una manera sencilla de almacenar en el servidor datos comunes a todos los visitantes de nuestro sitio web.

```
Application.Lock();
Application.Add("edad",22);
int valor=(int)Application["edad"];
valor++;
Application["edad"]=valor;
Application.UnLock();
Response.Write(Application["edad"]);
```

Problema

- Dónde inicializo una variable de aplicación para que no se reinicie cada vez (ej. Contador de visitas)????

5

Global.asax

El archivo global.asax

- Permite escribir código de aplicación global.
- No contiene etiquetas HTML ni ASP.NET
- Se utiliza para definir variables globales y reaccionar a eventos globales.
- Contiene código de tratamiento de eventos que reacciona a los eventos de aplicación o sesión.

El archivo global.asax

- Se añade a la aplicación web como un nuevo elemento Clase de aplicación global
 - Global.asax y Global.asax.cs

```
protected void Application_Start(object sender, EventArgs e)
{ Application["SiteName"] = "Mi aplicación"; }
```

```
protected void Session_Start(object sender, EventArgs e)
{
    Session.Timeout = 15;
    Response.Write("Servida el " + DateTime.Now.ToString());
}
```

- Importante:
 - Cualquier cambio en el archivo global.asax reiniciará la aplicación

Ejemplo, uso de objetos de sesión

- Se quiere modificar el timeout por defecto (20min)
- Se puede hacer en cualquier lugar del código pero lo más recomendable es hacerlo en el archivo Global.asax

Archivo Global.asax

```
protected void Session_Start(object sender, EventArgs e)
{
    Session.Timeout = 15;
}
```

El archivo global.asax

- Sólo puede haber un archivo global.asax
- Debe residir en el directorio raíz de la aplicación

Global.asax.cs

```
protected void Application_Error(object sender, EventArgs e)
{
    Response.Write("<b>");
    Response.Write("OOps! Ha ocurrido un error! </b>");
    Response.Write(Server.GetLastError().Message.ToString());
    Response.Write(Server.GetLastError().ToString());
    Server.ClearError(); }
```

Default.aspx.cs

```
int j = 1;
int x = 0;
int k=j / x;
```

OOps! Ha ocurrido un error! System.Web.HttpUnhandledException: Se produjo una excepción de tipo 'System.Web.HttpUnhandledException' ---> System.DivideByZeroException:

CONTADOR DE VISITAS

Añadir nuevo elemento...

- Clase de aplicación Global → Global.asax

```
void Application_Start(object sender, EventArgs e)
```

```
{
```

```
// Código que se ejecuta al iniciarse la aplicación
```

```
Application.Add("contador", 0);
```

```
}
```

6

Cookies

Cookies

- Para almacenar datos relativos a un usuario se puede utilizar el objeto Session
- Problema:
 - Los datos se borran cuando el usuario cierra la ventana del navegador
- Solución:
 - Para almacenar datos y que éstos se preserven es necesario utilizar las cookies

Cookies

- Las cookies son extractos de datos que una aplicación ASP.NET puede almacenar en el navegador del cliente para su posterior recuperación
- Las cookies no se pierden cuando se cierra el navegador (a no ser que el usuario las borre)

Cookies en ASP.NET

- Una cookie se representa por la clase HttpCookie
- Las cookies del usuario se leen a través de la propiedad Cookies del objeto Request
- Las cookies del usuario se modifican a través de la propiedad Cookies del objeto Response
- Por defecto las cookies exirpan cuando se cierra el navegador
 - Se pueden alterar los puntos de expiración (poner una fecha determinada de expiración)

Cookies en ASP.NET

Página default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie userCookie;
    userCookie = Request.Cookies["UserID"];
    if (userCookie == null)
    {
        Label1.Text ="No existe la cookie, creando cookie ahora";
        userCookie = new HttpCookie("UserID", "Ana López");
        userCookie.Expires = DateTime.Now.AddMonths(1);
        Response.Cookies.Add(userCookie);
    }
    else
    {
        Label1.Text = "Bienvenida otra vez, " + userCookie.Value;
    }
}
```

Cookies en ASP.NET

- La variable userCookie se inicializa como una instancia de la clase HttpCookie y se le asigna el valor de la cookie llamada UserID
- Se comprueba la existencia de la cookie
 - Caso de no existir se muestra un mensaje
 - Se le da el valor “Ana López”
 - Se le asigna una fecha de expiración
- La cookie se transfiere al navegador usando el método Response.Cookies.Add
- Si la cookie existe se muestra un mensaje de bienvenida

Cookies en ASP.NET

- La primera vez que se carga la página se mostrará el mensaje
 - No existe la cookie, creando cookie ahora
- Si se recarga de nuevo la página, la cookie ya existirá y se mostrará el mensaje
 - Bienvenida otra vez, Ana López
- Cuidado!
 - Ten en cuenta que los usuarios pueden rechazar las cookies
 - No puedes dejar que recaigan en ellas aspectos importantes de la aplicación

Cookies en ASP.NET

- Borrar cookies
 - La única forma es reemplazarla por una cookie con una fecha de expiración que ya ha pasado

```
HttpCookie userCookie= new HttpCookie("UserID");
userCookie.Expires=DateTime.Now.AddDays(-1);
Response.Cookies.Add(userCookie);
```

7

Email

Email

- Supongamos que tenemos una tienda online y queremos enviar un email de confirmación de pedido a cada cliente
- En lugar de escribir manualmente cada email ASP.NET permite automatizar este proceso

Email

- System.Net.Mail
 - **SmtpClient**
 - **MailMessage**
 - **Attachment**
 - **AttachmentCollection**
 - **MailAddress**
 - **MailAddressCollection**

Email

- **MailMessage**
 - From
 - To
 - CC
 - Bcc
 - Attachments
 - Subject
 - Body
 - IsBodyHTML

Email

```
SmtpClient smtpClient = new SmtpClient("smtp.gmail.com",587);
MailMessage message = new MailMessage();
try
{
    MailAddress fromAddress = new MailAddress("irene@dlsi.ua.es", "Alias
    remitente");
    MailAddress toAddress = new MailAddress("correo@gmail.com", "Alias
    destinatario");
    message.Attachments.Add(new Attachment("C:\\\\imagen1.gif"));
    message.Attachments.Add(new Attachment("C:\\\\imagen2.jpg"));
    message.From = fromAddress;
    message.To.Add(toAddress);
    message.Subject = "Probando el envío!";
    message.Body = "Este es el cuerpo del mensaje";
    smtpClient.EnableSsl = true;
    smtpClient.Credentials = new System.Net.NetworkCredential("usuario",
    "password");
    smtpClient.Send(message);
    Label1.Text = "Mensaje enviado.";
}
catch (Exception ex)
{
    Label1.Text = "No se pudo enviar el mensaje!";
}
```

Hada T9: Bibliotecas

Herramientas Avanzadas para el Desarrollo de
Aplicaciones

Objetivos del tema

- Saber qué es una biblioteca. Aprender a crearlas y usarlas.
- Saber qué son las referencias en un proyecto C#.
- Conocer y saber usar los proyectos de tipo *Librería* en C#+VisualStudio.

¿Qué es una biblioteca?

- De manera muy resumida podemos decir que una *biblioteca* -A lo largo del tema emplearemos el término *biblioteca* en lugar de *librería* por ser el primero más apropiado.- es un compendio de recursos (normalmente binarios): *subprogramas, clases, datos, iconos, etc...*
- **Cuando distribuimos estos recursos dentro de una biblioteca estamos favoreciendo su uso y reutilización.**
- ¿Motivo?: En el caso de *código fuente* no es necesario recompilar ya que éste se distribuye dentro de la biblioteca en forma binaria, ya compilado; hasta ahora solo sabíamos redistribuirlo en forma de código fuente.
- **Para emplear una biblioteca hemos de enlazar nuestro código con dicha biblioteca,** de esta forma tenemos acceso a su contenido.

¿Por qué distribuir algo en formato binario?

- Si para usarlo debe estar en formato binario, le evitamos al usuario del mismo tener que generar este formato binario a partir de ‘sus fuentes’.
- En ocasiones el proceso de compilación y obtención de una biblioteca es costoso y puede que no sea sencillo.
- En el caso de las **bibliotecas de enlace dinámico** tenemos la ventaja de poder cambiarlas para solucionar problemas sin necesidad de recompilar.
- Aunque estas bibliotecas en ocasiones son fuente de numerosos problemas, echa un vistazo al concepto de [DLL Hell](#).

Bibliotecas estáticas vs dinámicas

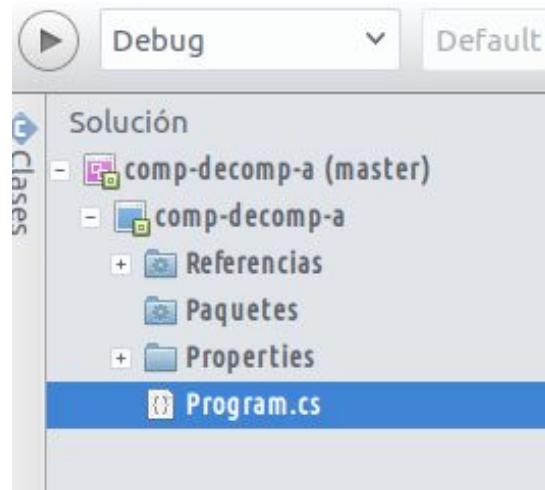
- **Estáticas:** (*unix extensión .a, win extensión .lib*)
 - El código de la librería se adjunta al ejecutable, que incrementa de tamaño. (*compile time*)
 - Ventajas:
 - *No hay problemas de dependencia.*
 - *Mejoras en el rendimiento*
 - *Simple distribución e instalación.*
- **Dinámicas:** (*unix extension .so, win extension .dll*)
 - Se enlaza al poner el ejecutable en marcha. (*runtime*)

Ejemplo de creación y uso de bibliotecas

- Partimos de un código **monolítico** (todo-en-uno) donde el programa principal y las funciones que éste emplea están en un único archivo. Llamamos a esta solución: `comp-decomp-a`.
- Se trata de una aplicación que implementa un sencillo algoritmo de compresión de cadenas. Puedes tratar de implementar la descompresión (de ahí el “*decomp*” del nombre).
- Invocado de esta forma: `comp-decomp -c ccccaassssssssaaaaaa` produce esta salida: Compresion de “ccccaaaaaaa”(21) es “4caa8s7a”(8).
- Posteriormente crearemos una versión en la que el código del método que comprime la cadena y la clase a la cual pertenece, se encuentra en otro proyecto, el cual creará una biblioteca con la que enlazar el programa principal. Llamamos a esta solución: `comp-decomp-b`.

Comp-decomp-a I

- Se trata de una solución que consta de un solo proyecto.



- El archivo `Program.cs` contiene todo el código de la aplicación, de forma resumida:

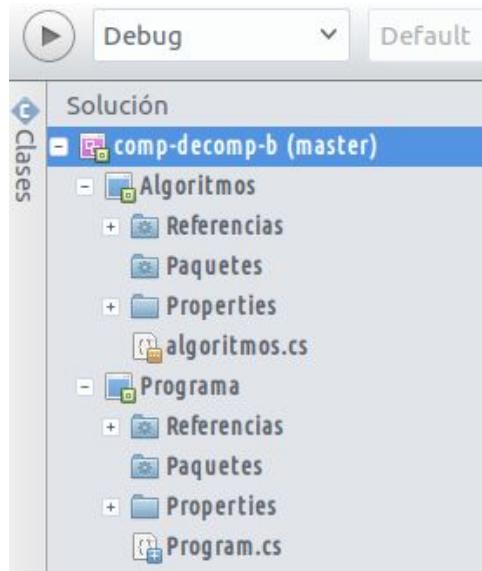
Comp-decomp-a II

- **Program.cs:**

```
namespace CompDecomp {
    class Algoritmos {
        private static int caracteresIguales (string s) {...}
        public static void comprime (string s, ref string cs) {...}
    }
    class MainClass {
        public static void Main (string[] args) {
            Console.WriteLine ("Hello compressed World!");
            string s="cccccaaaaaaaaaaaaaaa", cs="";
            Algoritmos.comprime (s, ref cs);
            Console.WriteLine ("{{0}}{{1}} compressed is [{2}{{3}}]", 
                s, s.Length, cs, cs.Length);
        }
    }
}
```

Comp-decomp-b I

- Se trata de una solución que consta de dos proyectos: Algoritmos y Programa.



- De manera resumida, el contenido de algoritmos.cs y Program.cs es:

Comp-decomp-b II

- **Program.cs:** fíjate que pertenece al proyecto “Programa”

```
namespace CompDecomp {
    class MainClass {
        public static void Main (string[] args) {
            Console.WriteLine ("Hello compressed World!");
            string s="cccccaaaaaaaaaaaaaa", cs="";
            Algoritmos.comprime (s, ref cs);
            Console.WriteLine ("{0}({1}) compressed is [{2}({3})]",
                               s, s.Length, cs, cs.Length);
        }
    }
}
```

Comp-decomp-b III

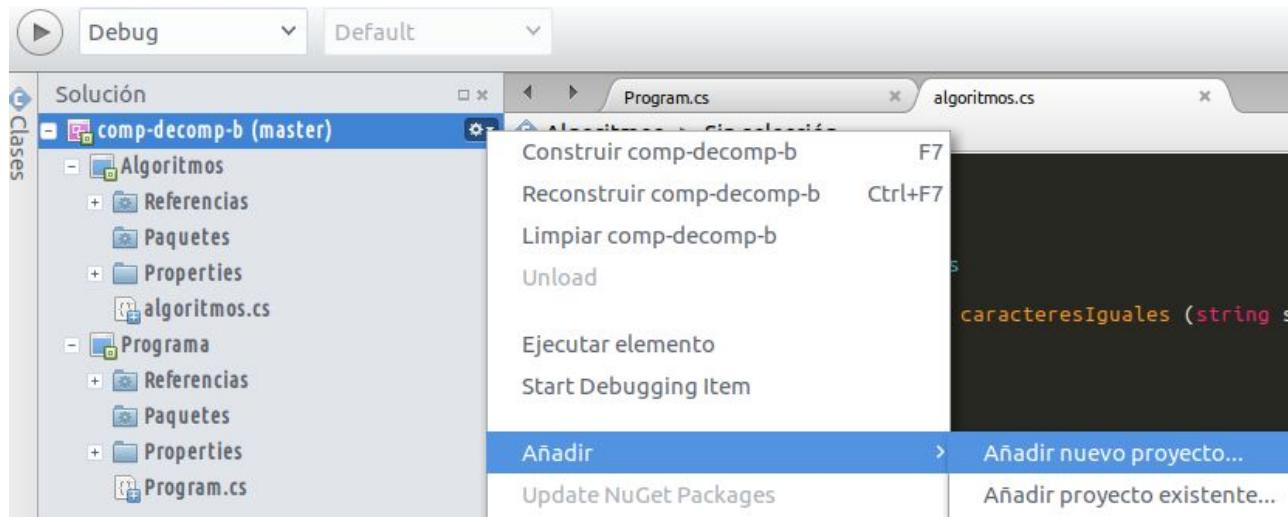
- **algoritmos.cs**: fíjate que pertenece al proyecto “Algoritmos”

```
namespace CompDecomp {  
    public class Algoritmos {  
        private static int caracteresIguales (string s) {...}  
        public static void comprime (string s, ref string cs) {...}  
    }  
}
```

- Por cierto...comprueba qué ocurre si quitamos el atributo **public** de la clase **Algoritmos**.

Comp-decomp-b IV

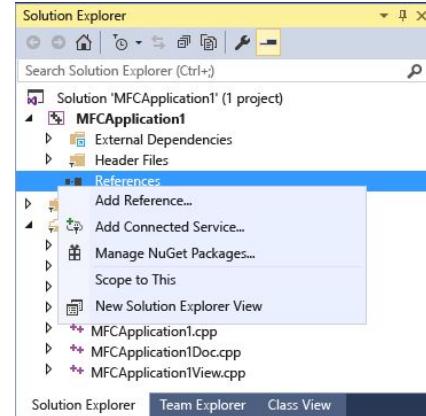
- ¿Cómo se añade un nuevo proyecto a una solución?



- Este botón de configuración, p.e. también nos permite renombrar un proyecto, un archivo, etc...
- También podemos acceder a estas operaciones pulsando el botón derecho del ratón sobre el elemento correspondiente.

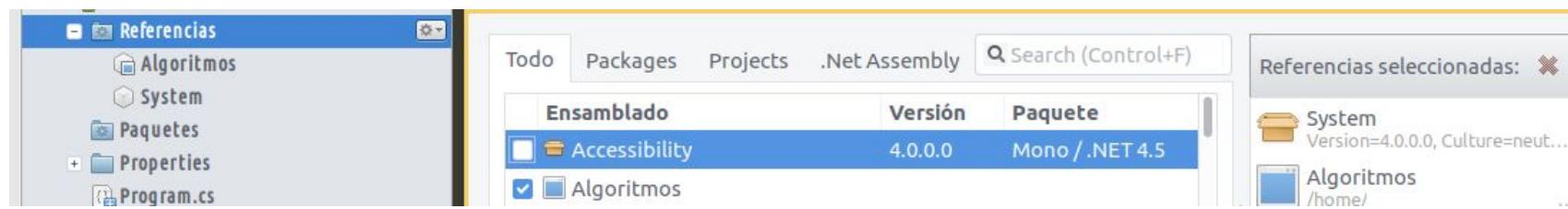
Comp-decomp-b V

- El proyecto Algoritmos **es de tipo librería y no aplicación de consola.**
- Su finalidad es producir una **DLL** y no un ejecutable.
- Además de añadir el proyecto a la solución, hemos de añadir una **referencia** a este proyecto (Algoritmos) en el resto de proyectos de la solución que lo empleen, en este ejemplo sólo en el proyecto Programa.
- Seguro que ya te imaginas cómo se hace...



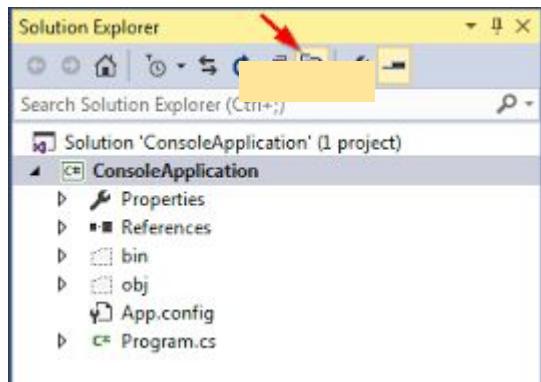
- Lo cual nos mostrará un diálogo como éste donde podremos seleccionarla:

Comp-decomp-b VI

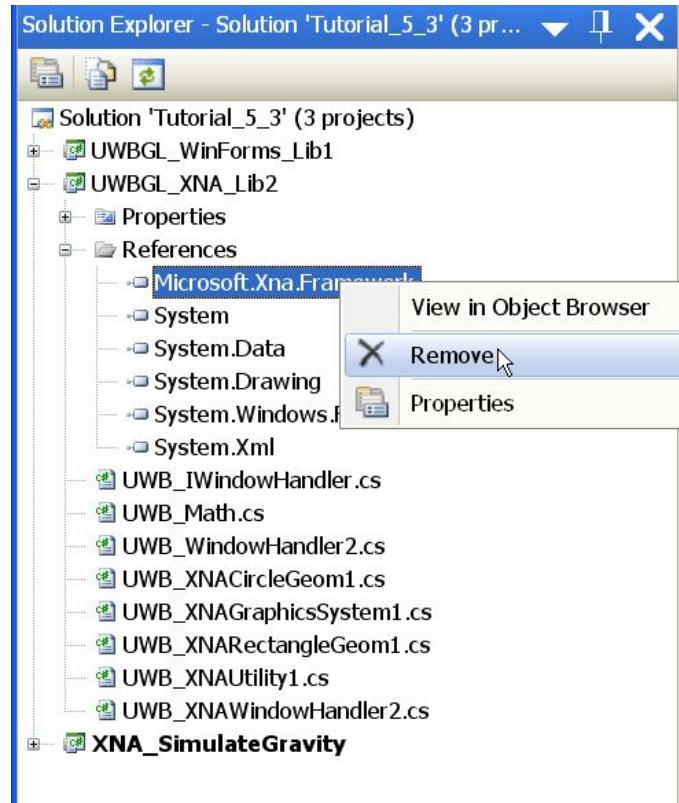


Carpeta Referencias

- Mira la carpeta **Referencias** de un proyecto:



Qué piensas que son estas referencias?



Visual Studio y proyectos de tipo “Librería”.

- **Los proyectos de tipo *librería* se pueden crear independientemente de que tengan un proyecto de tipo “aplicación gráfica o de consola” asociados en la misma solución.** De hecho, suele ser lo habitual.
- **Podemos crear nuestros proyectos de tipo *librería* que posteriormente podemos reutilizar en aplicaciones concretas.**
- Es una buena manera de dividir el trabajo dentro de un grupo de programadores. Cada *subgrupo* se dedica a crear una librería.

En la práctica en grupo...

- Una solución con 2 proyectos:
 - Proyecto Web (con referencia al proyecto de librería)
 - Contendrá la interfaz, manejadores y validaciones
 - Proyecto de Librería de clases con las Capas EN y CAD
 - Carpeta EN
 - Clases EN para entidades del sistema
 - Carpeta CAD
 - Por cada EN una clase CAD para el acceso a datos

Presentaciones efectivas

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Normas presentación

- **CADA GRUPO:**
 - video/slides+audio: 15 minutos de presentación
 - TODAS las personas presentan y TODAS las personas del grupo deben asistir a la corrección y pueden recibir preguntas.

CONTENIDO

- ¿QUÉ contar????

Consejos contenido

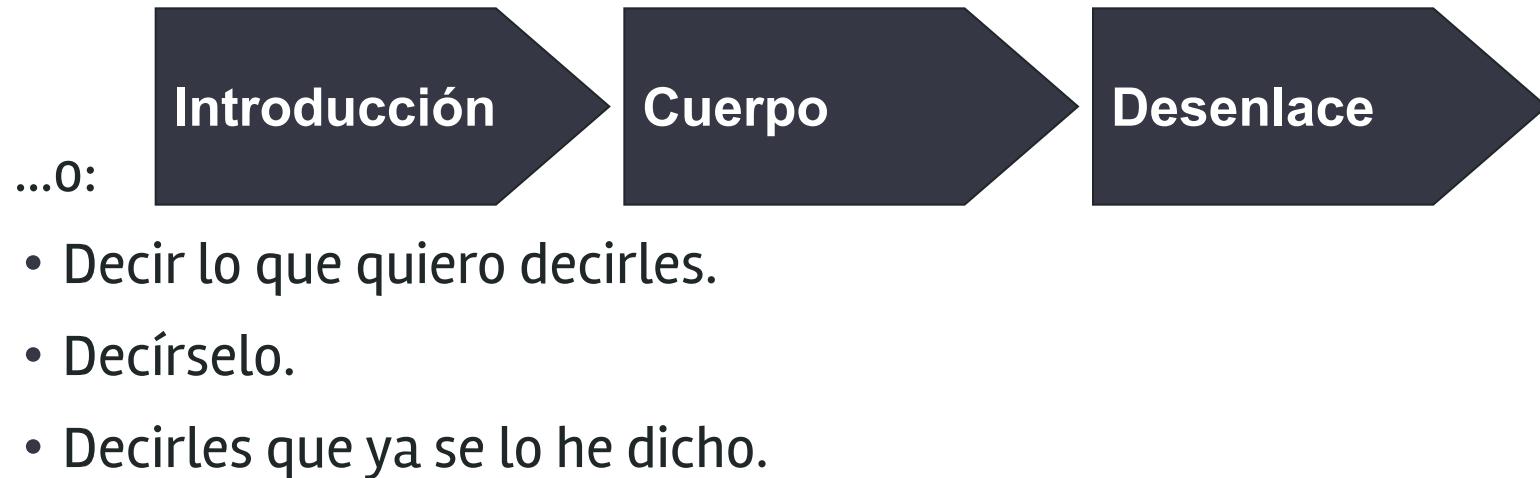
- **Contad lo que os diferencia del resto**
- Destacad las **mejoras** hechas
- La aplicación se intenta **vender** en la presentación a una persona que sabe de informática.

FORMA

- ¿CÓMO contar lo????

Estructura de la presentación

“El principio de 3”



Introducción



- Captar la atención del auditorio.
- Definir los objetivos de la presentación.
- Establecer tu credibilidad.
 - ¿Qué estamos haciendo por ellos?
 - Empezar por lo último o más reciente
 - Contextualizar a la actualidad

Cuerpo



1. Tormenta de ideas.
2. Identificar las ideas principales y materializarlo en mensajes.
(borrador).
3. Seleccionar el número de puntos que se van a utilizar.
4. Organizar y reordenar los puntos (guión o esquema).

Cuerpo



“Prevenir la divagación”

- Visualizar el total de la presentación.
- Asegurar que la idea principal está presente en todos los mensajes.
- Seleccionar sólo los puntos indispensables y relevantes.

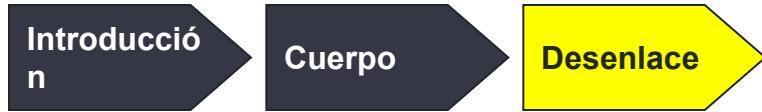
Cuerpo



“¿Qué más? ¿Cuánto más?”

- Siempre es preferible centrarse en **pocos puntos, pero buenos.**
- Hay que tener un plan para eliminar determinado material si el tiempo se acorta.
- Pensar qué podemos ofrecer como ayuda a cambio de algún tema que no nos da tiempo a explicar (material de consulta, anexos o referencias).

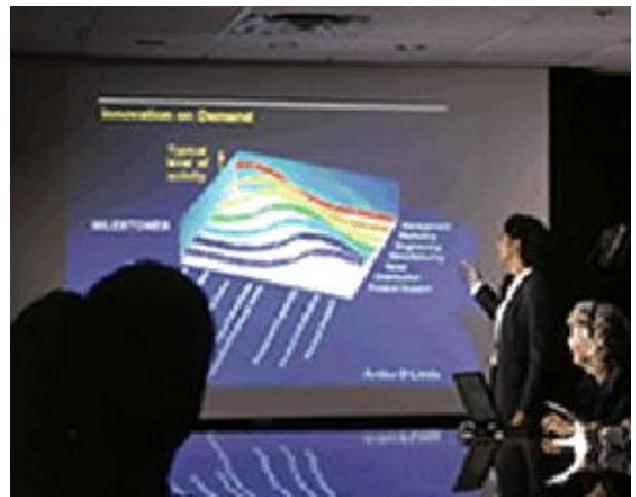
Desenlace



- Reiterar nuestro mensaje principal.
- Integrar los puntos de la introducción en las frases de conclusión: “cerrar el círculo”.
- Llamar a la acción: Propuestas, objetivos inmediatos, propósitos.

Errores que debemos evitar

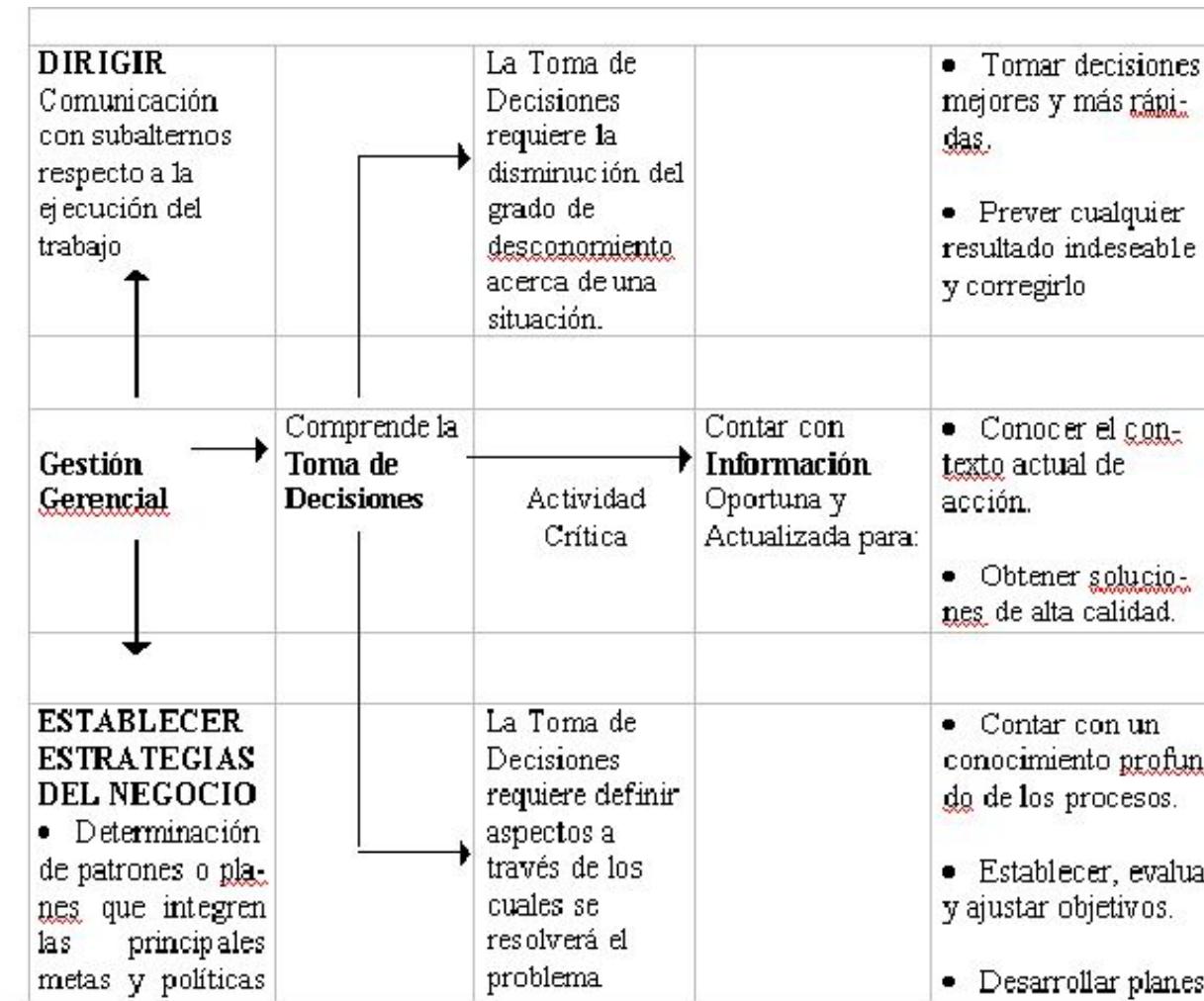
- Demasiado en poco espacio
- Demasiadas palabras en poco espacio
- Imágenes deslucidas
- Omisiones por falta de revisiones previas
- Errores de transcripción y/o redacción



Demasiadas palabras

- Sistemas de Procesamiento de Datos: Son aquellos *sistemas de información computarizados que se desarrollan para procesar grandes volúmenes de información generada en las funciones administrativas*, tales como nómina o el control de inventario. Los sistemas de procesamiento de datos liberan el tedio y la rutina a las tareas que se realizan manualmente; sin embargo, el elemento humano sigue participando, al llevar a cabo la captura de los datos requeridos.
- Las transacciones más comunes incluyen: facturación, entrega de mercancía, pago a empleados, depósitos, retiros y transferencias bancarias.
- El procesamiento de transacciones, que es el conjunto de procedimientos para el manejo de éstas, incluye entre otras, las siguientes actividades: Cálculos, Clasificación, Ordenamiento, Almacenamiento y Recuperación, Generación de Resúmenes.

Demasiada información



El procesamiento de transacciones, que es el conjunto de procedimientos para el manejo de éstas, incluye entre otras, las siguientes actividades: Cálculos, Clasificación, Ordenamiento, Almacenamiento y Recuperación, Generación de Resúmenes.

...lo ideal es...

- No más de 6 aspectos por diapositiva con oraciones claras y concisas que indiquen al auditorio el aspecto que trata en ese momento el ponente.



Tamaño de imágenes



Evitar el uso de imágenes difusas o con poca resolución

Combinaciones Fondos – Letras recomendados

Fondos oscuros y letras claras

Fondos claros y letras oscuras

Combinación

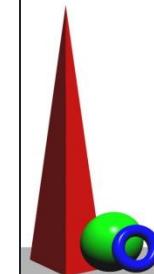
Combinación

Combinación

Combinación

Combinación

Combinación



Combinaciones Fondos – Letras no recomendados

Fondos oscuros y letras oscuras

Fondos claros y letras claras

Fondos recargados donde las letras no se aprecian bien

Combinaciones que hacen contrastes erróneos



Tamaño de letras

Tipo de letras más utilizados Times New Roman y Arial

- Informática 8
- Decanato 10
- Administración 12
- Matemática 14
- Inglés 16
- Proyecto 18
- Examen Final 20

• Informática 24

• Decanato 28

• Matemática 44

No se aprecian bien en el Auditorio

1 36

Mínimo Recomendado



Diapositivas efectivas

- Simples.
- Gráficos, iconos y símbolos.
- Palabras claves, no frases.
- Una idea por diapositiva.
- 3–6 puntos por diapositiva.
- Tamaño 20 como mínimo para las fuentes.
- Usar el color, pero no excesivamente.

*Por favor no usar
efectos sonoros*

Presentación en equipo

- Todos los miembros deben tener un papel.
- Se debe acordar previamente “Quién hace qué”.
 - Concretar la versión final para cada persona.
 - Ensayar en equipo.
 - Determinar quién puede responder a las preguntas.
- **Establecer la transición entre oradores.**
- Presentación y cierre por el líder.
- ¿Cuántos miembros en el equipo?
 - Depende del auditorio.

Realizando la
presentación...

En una encuesta se preguntó sobre los miedos más comunes y la respuesta fue:

Fuego
3%

Arañas
15%

**Hablar
en
público**

54%

Muerte
16%

Ahogarse
5%

Enfermedad
7%

Voz y lenguaje corporal

Ojos

- Hacer contactos cada cinco segundos o visión general.

Voz

- Hablar al fondo de la habitación.
- Evitar ruidos (Hmm, Umm, ehh).
- Variar el tono y acentos.
- Despacio, sin prisas. Ritmo.
- Ir al grano.

Voz y lenguaje corporal II

Posturas y movimientos

- Pies quietos y movimiento.
- Canalizar el nerviosismo de las manos.
- Estar relajado, pero no apoltronado.

Gestos

- Uso armónico de las manos.
- Evitar los bolsillos.
- No esconderse.
- Sonreír, expresar.
- Mostrarse natural.

Equipamiento

Regla de oro: Chequeo previo siempre

- Equipo proyector:
 - Saber cómo se enciende.
 - **No llegar demasiado tarde.**
 - No hablar a la pantalla.
- Pizarras:
 - Escribir con grandes letras (Mayúsculas).

Tema 11. AJAX en .net

Herramientas Avanzadas para el Desarrollo de Aplicaciones

1

Introducción a AJAX

AJAX= Asynchronous JavaScript and XML

- Cura para “enfermedades comunes” de las aplicaciones basadas en navegador
 - Usa el objeto **XmlHttpRequest** para recuperar datos del servidor de forma asíncrona y **JavaScript** para actualizar el contenido de la página.
 - Elimina el parpadeo y usa el ancho de banda de manera más eficiente.
- Se utiliza en Google Suggests, Google Maps, Microsoft Virtual Earth, Outlook Web Access... etc

http DESVENTAJAS

- Período demasiado largo al refrescar la página
- El usuario debe esperar a que la petición (request) finalice para seguir interactuando con la aplicación.
- SOLUCIÓN: Manejo de la petición de un modo asíncrono
- ***Llamadas asíncronas que no bloquean la interfaz de usuario mientras se producen.***

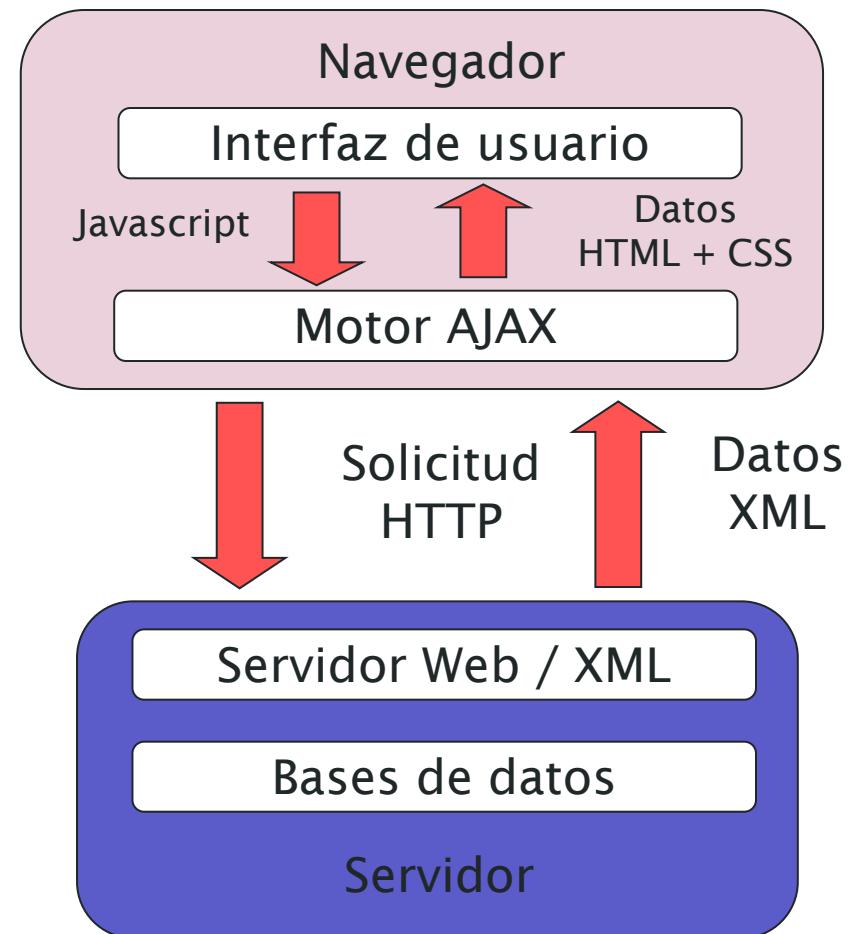
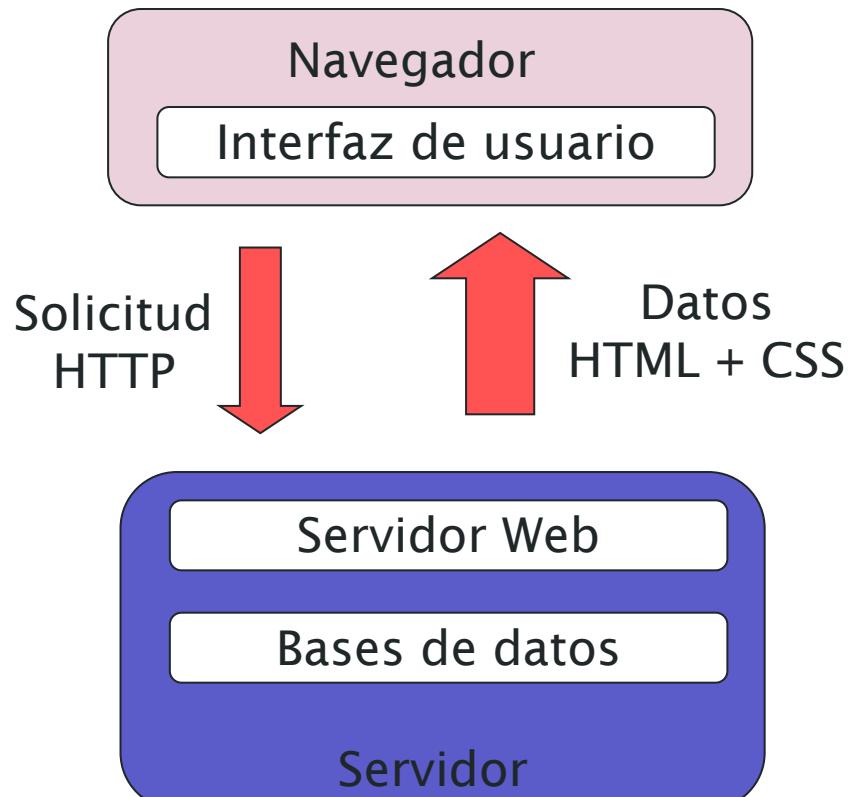


- Tecnología de Microsoft para desarrollar aplicaciones Web basadas en AJAX
- Provee librerías script en el cliente que incorporan tecnologías **JavaScript** y HTML dinámico (**DHTML**), y las integra con la plataforma de desarrollo basada en **ASP.NET**
- También incluye componentes básicos en el servidor para soportar esas llamadas asíncronas del cliente
- Utilizando ASP.NET AJAX, se puede **mejorar la experiencia de usuario y la eficiencia de las aplicaciones Web.**

¿Por qué usar AJAX?

- **Eficiencia mejorada** ejecutando partes significantes de una página Web en el navegador.
- **Elementos de IU familiares**, típicas de una aplicación de escritorio, como ventanas pop-up, indicadores de progreso y tooltips.
- **Actualizaciones parciales de la página** que refrescan sólo las partes de la página Web que han cambiado.
- **Soporte para los navegadores más populares** y utilizados, como Microsoft Internet Explorer, Mozilla Firefox, y Apple Safari.

Modelo Clásico de aplicaciones web vs AJAX



Una aplicación Ajax

- Se elimina la naturaleza *start-stop-start-stop* de la interacción de la Web introduciendo un intermediario — **un motor Ajax** — entre el usuario y el servidor.
- En lugar de cargar una página Web, al inicio de la sesión, el navegador carga el **motor Ajax** — escrito en JavaScript , usualmente en un marco oculto.

Motor Ajax

- Este motor es responsable de presentar la interfaz al usuario y de comunicar con el servidor de parte del usuario.
- Permite que la interacción del usuario con la aplicación sea **asíncrona**.
- **Cada acción del usuario** que normalmente generaría una petición HTTP se convierte en una **Llamada JavaScript al motor Ajax**.

Motor Ajax (II)

- Cualquier respuesta a una acción del usuario que no requiera volver al servidor es manejada por el motor.
 - como validación de datos simple, editar datos en memoria, y incluso alguna navegación
- Si el motor necesita algo del servidor para elaborar la respuesta el motor hace estas peticiones de manera **asíncrona**, usualmente utilizando XML, sin paralizar la interacción del usuario con la aplicación.

ASP.Net

2

Creación de una aplicación Web AJAX

Controles de servidor ASP.net

AJAX

- Consisten en código de cliente y de servidor que se integran para producir el comportamiento AJAX.
- **ScriptManager**
 - Registra el script de la librería de microsoft AJAX en la página. Esto permite al script de cliente admitir características como la representación parcial de páginas y las llamadas a servicios web.
El control ScriptManager es necesario para utilizar los demás controles.

Nuevo proyecto

Recente

.NET Framework 4.5

Ordenar por: Predeterminado

Buscar en la Plantillas instalado (Ctrl+E)

Instalado

Plantillas

Visual Basic

Visual C#

Escritorio de Windows

Web

Visual Studio 2012

Office/SharePoint

Cloud

LightSwitch

Prueba

Reporting

Silverlight

WCF

Workflow

Visual C++

Visual F#

SQL Server

Python

TypeScript

Otros tipos de proyectos

Proyectos de modelado

En línea

[Haga clic aquí para buscar plantillas en línea.](#)

Nombre:

WebApplication2

Ubicación:

c:\users\irene\documents\visual studio 2013\Projects

[Examinar...](#)

Nombre de la solución:

WebApplication2

 Crear directorio para la solución Agregar al control de código fuente[Aceptar](#)[Cancelar](#)

Aplicación web ASP.NET

Visual C#

Tipo: Visual C#

Una plantilla de proyecto para crear aplicaciones ASP.NET. Puede crear aplicaciones ASP.NET Web Forms, MVC o Web API y agregar muchas otras características en ASP.NET.

Default.aspx

```
<%@ Page Language="VB" AutoEventWireup="true" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <div>
        </div>
    </form>
</body>
</html>
```



AJAXEnabledWebSite2 - Microsoft Visual Studio

Archivo Editar Ver Sitio Web Generar Depurar Formato



Times New Roman 12pt

B *I* U



Explorador de servidores

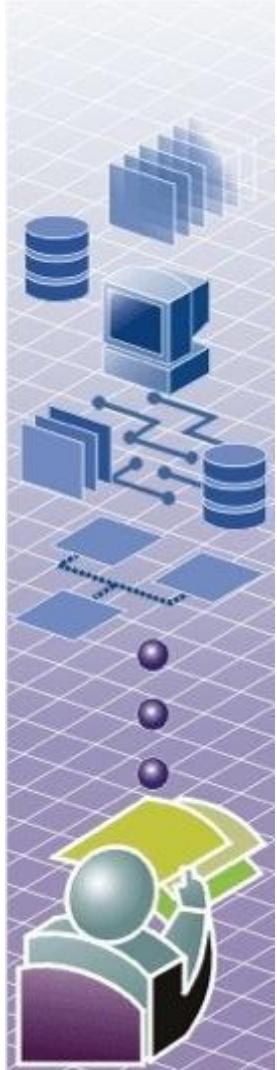
Default.aspx.vb Default.aspx Página de inicio

ScriptManager - ScriptManager1

Controles AJAX

- UpdatePanel
 - Permite refrescar las partes seleccionadas de la página, en lugar de refrescar la página completa utilizando un postback síncrono.
- UpdateProgress
 - Provee información de estado sobre actualizaciones parciales de la página en controles **UpdatePanel**.
- Timer
 - Ejecuta postbacks en intervalos de tiempo definidos. Puede utilizarse para enviar la página completa, o usarse junto al control **UpdatePanel** para realizar actualizaciones parciales de la página en un intervalo definido.

Ejercicio



Ejercicio

- ▶ En un formulario Web vamos a crear dos listas desplegadas (listBox) a las que vamos a insertar elementos desde un cuadro de texto.
- ▶ La primera lista se actualizará SIN recarga de la página.
- ▶ La segunda lista se actualizará normalmente.

Añadir elemento

elemento 1

añadir sin recarga

Añadir con recarga

no se recarga la página...

elemento 1



se recarga la página...

elemento 1



Código asociado

```
protected void Button1_Click(object sender, EventArgs e)
{
}
```

```
protected void Button2_Click(object sender, EventArgs e)
{
}
```

Código asociado

```
protected void Button1_Click(object sender, EventArgs e)  
{
```

```
    ListBox1.Items.Add(TextBox1.Text);
```

```
}
```

```
protected void Button2_Click(object sender, EventArgs e)
```

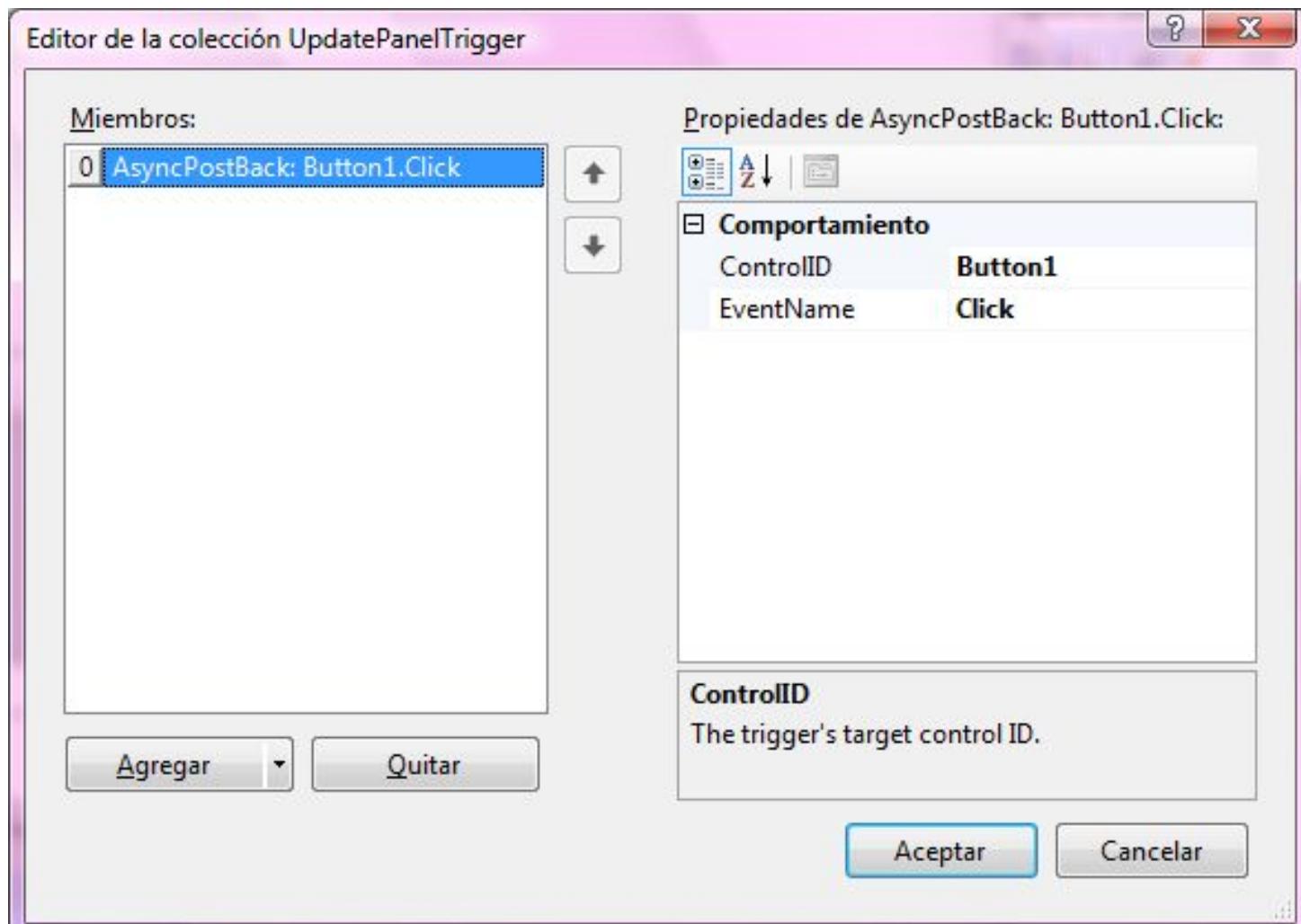
```
{
```

```
    ListBox2.Items.Add(TextBox1.Text);
```

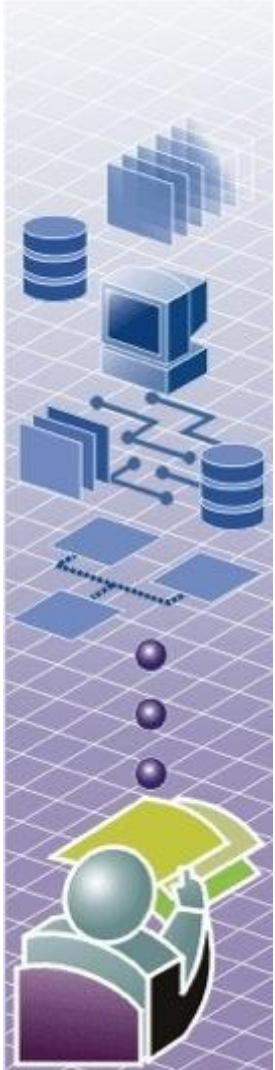
```
}
```

Update Panel

- Propiedad Triggers



Ejercicio



Ejercicio

- ▶ Vamos a añadir ahora un control UpdateProgress.
- ▶ En este control añadimos el texto “Actualizando...” y la imagen indicator.gif (cv).
- ▶ Vamos a añadir un retardo de 2 segundos para hacerlo realista.

Añadir elemento

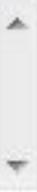
elemento 2

Actualizando... 

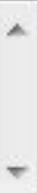
Añadir sin recarga

Añadir con recarga

no se recarga la página...

elemento 1 

se recarga la página...

elemento 1 

Código a añadir...

```
protected void Button1_Click(object sender, EventArgs e)
{
    ...
    System.Threading.Thread.Sleep(2000);
}
```

Código a añadir...

```
protected void Button1_Click(object sender, EventArgs e)
{
    ListBox1.Items.Add(TextBox1.Text);
    System.Threading.Thread.Sleep(2000);
}
```

ASP.Net

ASP.net Ajax Control Toolkit

3

ASP.NET AJAX Control Toolkit

- Está desarrollado en base a ASP.NET AJAX y contiene una serie de controles Web y extendedores con los que podremos utilizar las avanzadas características de ASP.NET AJAX .

ASP.NET AJAX Control Toolkit

- Vamos a distinguir entre controles Web y extendedores, donde los primeros tienen una entidad por sí mismos, mientras que los segundos únicamente añaden un comportamiento a un control Web existente.
- Estos controles van desde un simple botón con una alerta asociada, hasta un complejo panel que podemos arrastrar por la pantalla;
 - en ambos casos, mandando y recogiendo información entre el cliente y el servidor sin ningún tipo de recarga de página.
- Su uso hará que nuestra Web sea mucho más atractiva, potente y efectiva.

Pasos instalación toolkit

1. Descargar el instalador en:
<https://www.devexpress.com/products/ajax-control-toolkit/>



What's New in AJAX Control Toolkit

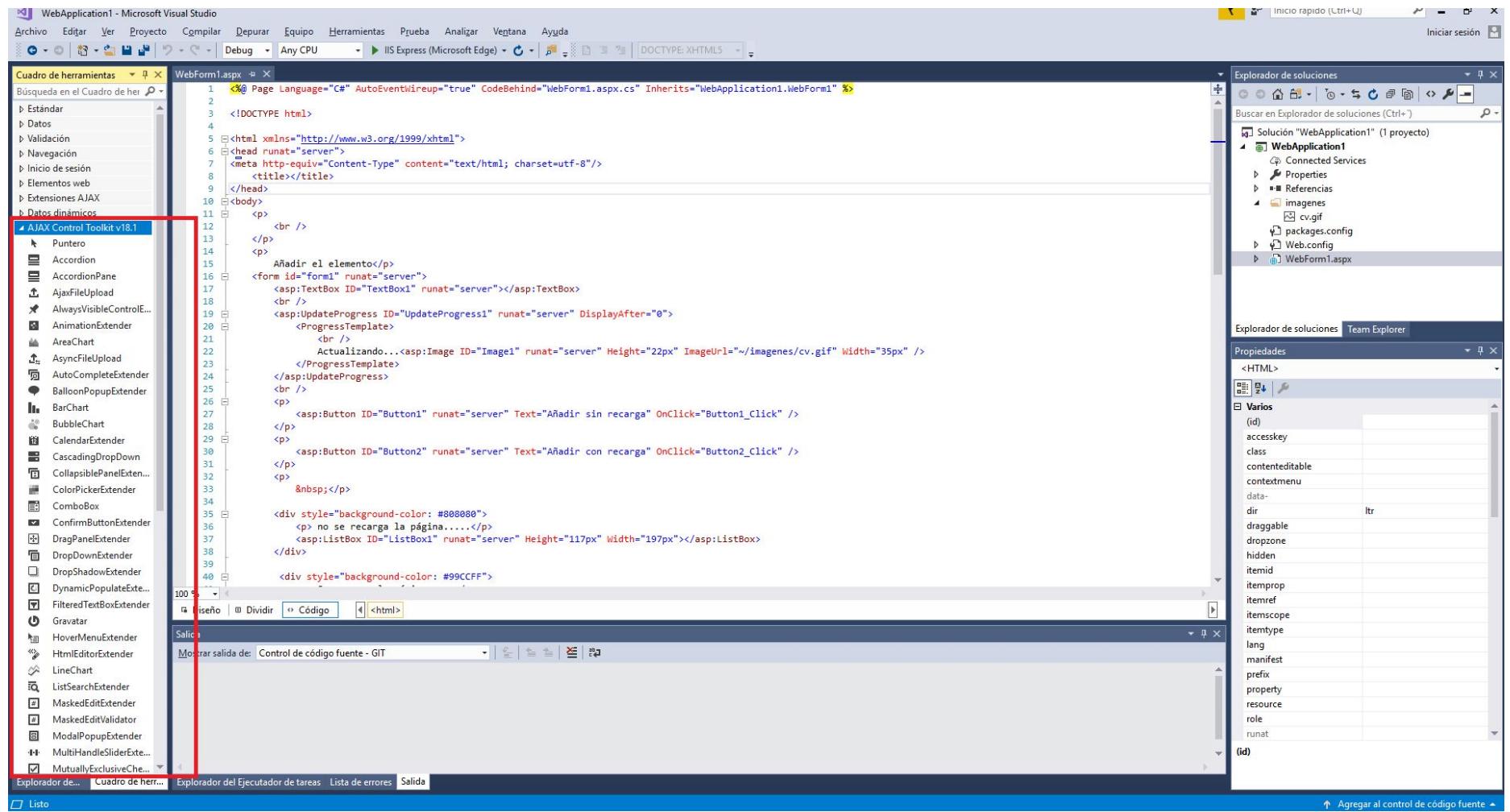
Free to use. More Demandable than Ever Before.

Pasos instalación toolkit

- Instalarlo (es un exe): lo que hace es añadir una librería DLL al visual studio con los controles nuevos



Comprobar instalación



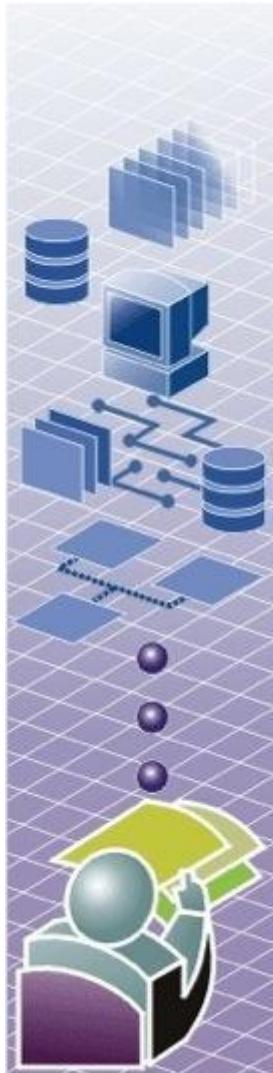
EJEMPLOS DE CONTROLES AJAX

- <http://www.ajaxcontroltoolkit.com/>

AlwaysVisible extender

- El extendedor **AlwaysVisibleControl** permite colocar un control en la página Web que permanece visible y “flotante” sobre el contenido de la página mientras hacemos scroll o redimensionamos la ventana del navegador.
- Cualquier control puede tener asociado este extendedor.

Exercise



Ejercicio

- Añadir un panel estándar, añadirle color de fondo y borde y el texto “siempre visible”.
- Copiar de internet una página web para que haya texto suficiente para hacer scroll.
- Añadir el extendedor alwaysvisible al panel.

AlwaysVisibleControl extender

Página sin título - Windows Internet Explorer
http://localhost:49596/animation.aspx

1 May 2007 ... We will configure this **extender** do the following: ... **Nothing** much to read here.
siempre visible by: Luke Zhang on September 14, 2007 12:00 AM ... with 7 or 8 columns, and the
animation is exactly what I needed to let the user ...
mattberseth.com/.../7_simple_steps_to_ajaxenable_y.html - En caché - Similares

[creativeUI - How To Create a Side Scrolling Page Effect - Part I ...](#) - [Traducir esta página]
1 Aug 2007 ... The #Body and #Page classes are **nothing** special...they basically make the ...
Their 4th class probably has something to do with the rest of ... The last part is simply to create
the **AnimationExtender** which looks like so: ...
www.creativeui.com/.../how-to-create-a-side-scrolling-page-effect-part-i-aspnet-ajax/ -
En caché - Similares

[Jeff Prosise's Blog : Creating Sophisticated Animations with the ...](#) - [Traducir esta página]
4 Apr 2007 ... The **AnimationExtender** control in the ASP ... NET AJAX to do animations on
non-ASP.NET platforms such as PHP and ColdFusion, you can't use the ASP ... NET
developers are supposed to use, although there's **nothing** ...
www.wintellect.com/.../Creating-Sophisticated-Animations-with-the-Microsoft-AJAX-Library.aspx
- En caché - Similares

Is the fast size **extender** a scam or **does** it work? - Yahoo! Answers - [Traducir esta página]
27 Sep 2009 Does the fast size **extender** really work? I saw an Penis size has **nothing**
Lista Internet | Modo protegido: desactivado 100%

código

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>

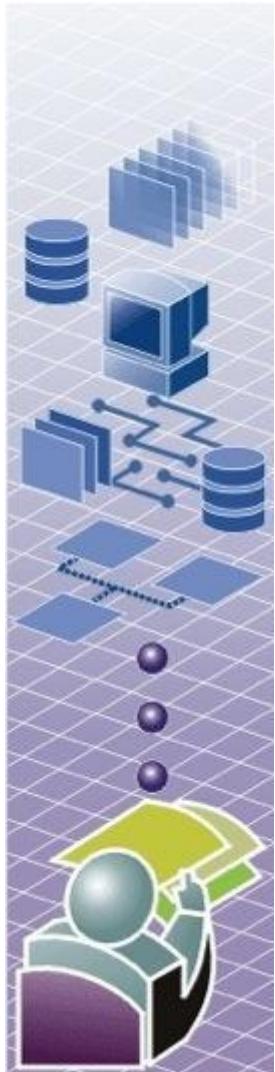
<asp:Panel ID="Panel1" runat="server" BackColor="#CC99FF">
    siempre visible
</asp:Panel>

<asp:AlwaysVisibleControlExtender
    ID="Panel2_AlwaysVisibleControlExtender"
    runat="server" Enabled="True" TargetControlID="Panel1">
</asp:AlwaysVisibleControlExtender>
```

CollapsiblePanel extender

- Con el CollapsiblePanel conseguiremos que cualquier control ASP.NET pueda ser maximizado o minimizado a nuestro antojo.
- Distinguiremos entre el contenido, que será el control que vaya a cerrarse y abrirse (por ejemplo un Panel) y el controlador, que será el control sobre el que deberemos hacer clic para cerrar y/o abrir el contenido.
- El estado del contenido (abierto o cerrado) es guardado a lo largo de los postbacks, por lo que permanecerá igual cuando recarguemos una página.

Ejercicio



Ejercicio

- Crear un formulario web con un panel estándar que será la cabecera, y otro panel que será el contenido a expandir/contraer.
- En la cabecera añadimos una imagen y una etiqueta que inicialmente dirá “(mostrar detalles...)”.
- En el contenido escribimos texto y añadimos una imagen. **IMPORTANTE, este panel tendrá height=0**
- Añadimos un collapsiblepanel extender sobre el panel de contenido

Vista diseño

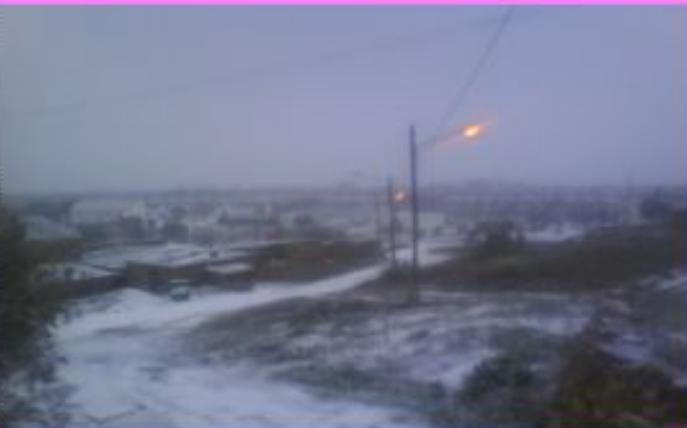
asp:Panel#Panel2

Información sobre mi ciudad [\(Mostrar detalles...\)](#)



Mi nombre es Joe. Vivo en un pueblo a las afueras

blabla



blabla

blabla

Código: Panel Titulo

```
<asp:Panel ID="Panel2" runat="server" BackColor="#FFCCFF"
    BorderColor="#993399" BorderStyle="Dashed" Height="28px"
    Width="648px">
    Información sobre mi ciudad &nbsnbsp;&nbsnbsp;&nbsnbsp;&nbsnbsp;
    <asp:Label ID="Label1" runat="server" Text="Label">
        (Mostrar detalles...)
    </asp:Label>

    <asp:Image ID="Image1" runat="server" Height="24px"
        ImageUrl("~/flechader.jpg" Width="23px" />

</asp:Panel>
```

Código: Panel contenido

```
<asp:Panel ID="Panel3" runat="server" BackColor="#FF99FF"  
    Height="0px" Width="652px">  
    Mi nombre es Joe. Vivo en un pueblo a las afueras<br />  
    <br />  
    blabla<br />  
    <br />  
    <asp:Image ID="Image2" runat="server" Height="177px"  
        ImageUrl="~/pueblonevado.jpg" Width="273px" />  
    <br />  
    blabla<br />  
    <br />  
    blabla  
</asp:Panel>
```

Código: CollapsiblePanel

```
<asp:CollapsiblePanelExtender  
    ID="Panel3_CollapsiblePanelExtender"  
    runat="server" Enabled="True"  
  
    TargetControlID="Panel3"  
    ExpandControlID="Panel2"  
    CollapseControlID="Panel2" Collapsed=true  
    TextLabelID="Label1" ExpandedText="Ocultar detalles"  
    CollapsedText="Mostrar detalles"  
    ImageControlID=Image1 ExpandedImage="~/flechaiz.jpg"  
    CollapsedImage="~/flechader.jpg" SuppressPostBack=true>  
  
</asp:CollapsiblePanelExtender>
```

Panel de contenido

Panel cabecera

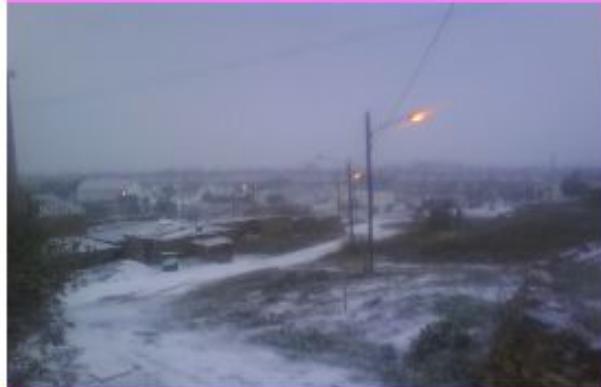
Ejecución

■ Información sobre mi ciudad (Mostrar detalles) 

■ Información sobre mi ciudad (Ocultar detalles) 

Mi nombre es Joe. Vivo en un pueblo a las afueras

blabla

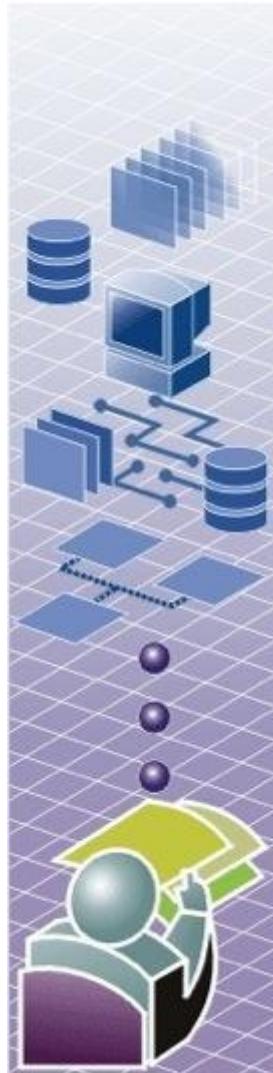


blabla

blabla

ModalPopup

- Con este extensor, conseguimos el efecto de mostrar contenido deshabilitando la interacción con el resto de la página.
- Podemos emular el efecto del famoso “window.open(...)” de javascript sin necesidad de salir de la página en que estamos ni de abrir una nueva ventana del navegador.



Ejercicio

- Crear un formulario web con los siguientes controles:
- Linkbutton
- Panel con el texto Seguro que desea salir? Y botones aceptar y cancelar
- ModalPopUp extensor

En ejecución

Página sin título - Windows Internet Explorer

http://localhost:49596/WebForm3.aspx

Google

Página sin título

Herramientas

[haz click para salir de la aplicación](#)

ASP.NET AJAX Control Toolkit nace como un proyecto conjunto entre la comunidad de programadores y Microsoft. Está desarrollado en base a ASP.NET AJAX y contiene una serie de controles Web y extendedores con los que podremos utilizar las avanzadas características de ASP.NET AJAX

sin más que un arrastre de ratón. Del mismo modo, con su descarga disponemos de ejemplos de uso, así como del propio código fuente de los controles. Y lo mejor de todo es que es totalmente gratuito.

Vamos a distinguir entre controles Web y extendedores, ya que los primeros son controladores que manejan una entidad por si mismos, mientras que los segundos únicamente añaden un comportamiento a un control Web.

Esta seguro que desea salir???

Aceptar Cancelar

Internet | Modo protegido: desactivado

100%

This screenshot shows a Windows Internet Explorer window running on a local host at port 49596. The page content discusses the ASP.NET AJAX Control Toolkit, mentioning its development by the community and Microsoft, and its availability as a free download with examples and source code. It also distinguishes between standard Web controls and extenders. A modal confirmation dialog box is displayed in the center of the page, asking if the user is sure they want to exit. The dialog has two buttons: 'Aceptar' (Accept) and 'Cancelar' (Cancel). The browser's status bar at the bottom indicates that protected mode is disabled.

Código Panel

```
<asp:Panel ID="Panel1" runat="server" BackColor="Silver"  
    BorderColor="#333333"  
        BorderStyle="Solid" Width="162px" Style="display:none">  
    <br />  
    <br />  
    &nbsp; Esta seguro que desea&nbsp;&nbsp;  
    <br />  
    &nbsp; salir??<br />  
    <br />  
    &nbsp;  
    <asp:Button ID="Button1" runat="server" Text="Aceptar" />  
    &nbsp;  
    <asp:Button ID="Button2" runat="server" Text="Cancelar" />  
    <br />  
</asp:Panel>
```

Código

```
<asp:ModalPopupExtender  
ID="LinkButton1_ModalPopupExtender" runat="server"  
DynamicServicePath="" Enabled="True"  
TargetControlID="LinkButton1" PopupControlID="Panel1">  
</asp:ModalPopupExtender>
```

SlideShow extender

- Imágenes con slideshow

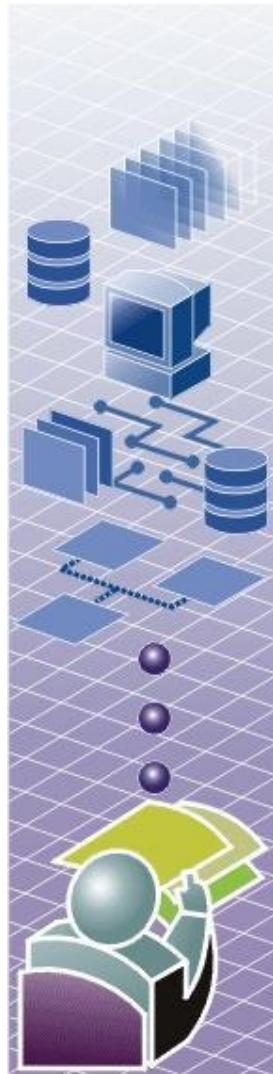


arbol

Prev

Next

Play



Ejercicio

- Crear un formulario web con los siguientes controles:
 - AjaxScriptManager
 - Control Imagen
 - Label
 - Tres botones (Prev, Next, Play)
 - Extensor SlideShow en la imagen

Código (I)

```
<asp:Image ID="Image1" runat="server" Height="300"  
    Style="border:1px solid black; width:auto" ImageUrl="images/Creek.jpg"  
    AlternateText="Rio"/>  
  
<asp:SlideShowExtender ID="Image1_SlideShowExtender" runat="server"  
    Enabled="True"  
    SlideShowServicePath="SlideService.asmx"  
    TargetControlID="Image1"  
    SlideShowServiceMethod="GetSlides"  
    AutoPlay="true"  
    ImageTitleLabelID="Label1"  
    NextButtonID="Button2"  
    PlayButtonID="Button3"  
    PreviousButtonID="Button1"  
    PlayButtonText="Play"  
    StopButtonText="Stop">  
</asp:SlideShowExtender>  
<br />
```

Necesitamos un
servicio Web

Código (II)

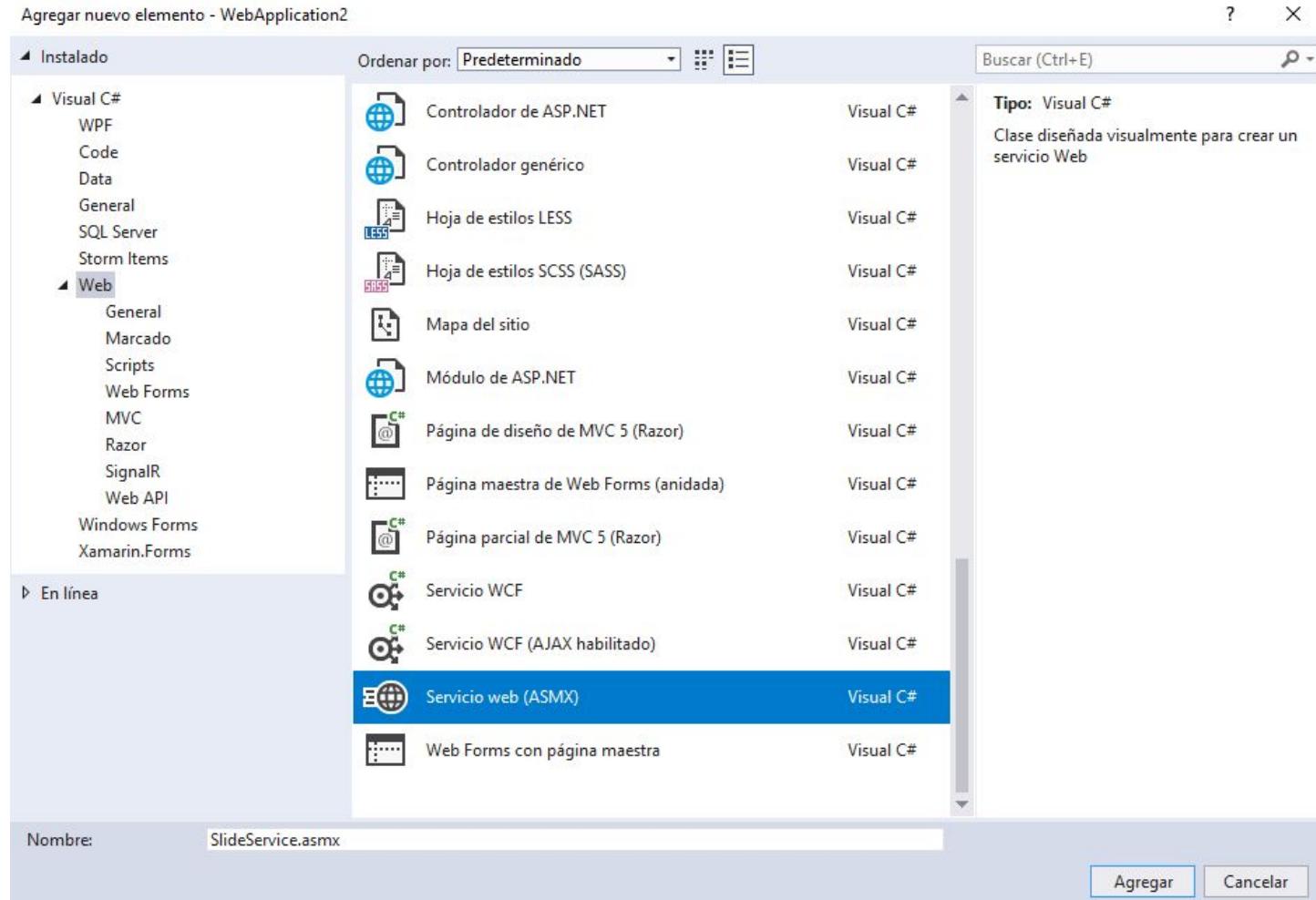
```
</div>
    <asp:Label ID="Label1" runat="server"></asp:Label>
    <br />

    <asp:Button ID="Button1" runat="server" Text="Prev"
Font-Size=Larger/>

    <asp:Button ID="Button2" runat="server" Text="Next"
Font-Size=Larger/>

    <asp:Button ID="Button3" runat="server" Text="Play"
Font-Size=Larger/>
</div>
```

Añadir elemento Webservice



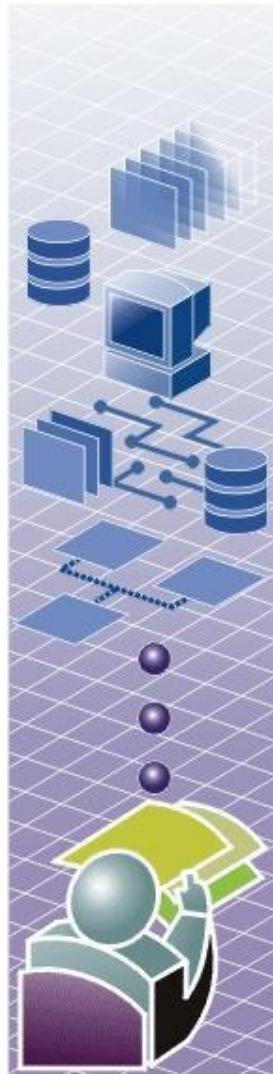
Añadir elemento Webservice

[System.Web.Script.Services.ScriptService]

```
public class SlideService : System.Web.Services.WebService
{
    [WebMethod]
    public AjaxControlToolkit.Slide[] GetSlides() {
        AjaxControlToolkit.Slide[] myslides=new AjaxControlToolkit.Slide[4];
        myslides[0]=new AjaxControlToolkit.Slide("images/Creek.jpg","Rio","Rio muy
bonito");
        myslides[1] = new AjaxControlToolkit.Slide("images/Dock.jpg", "puerto", "puerto
azul");
        myslides[2] = new AjaxControlToolkit.Slide("images/Garden.jpg", "jardin", "jardin
margaritas");
        myslides[3] = new AjaxControlToolkit.Slide("images/Tree.jpg", "arbol", "arbol
marron");
        return myslides;
    }
}
```

PasswordStrength

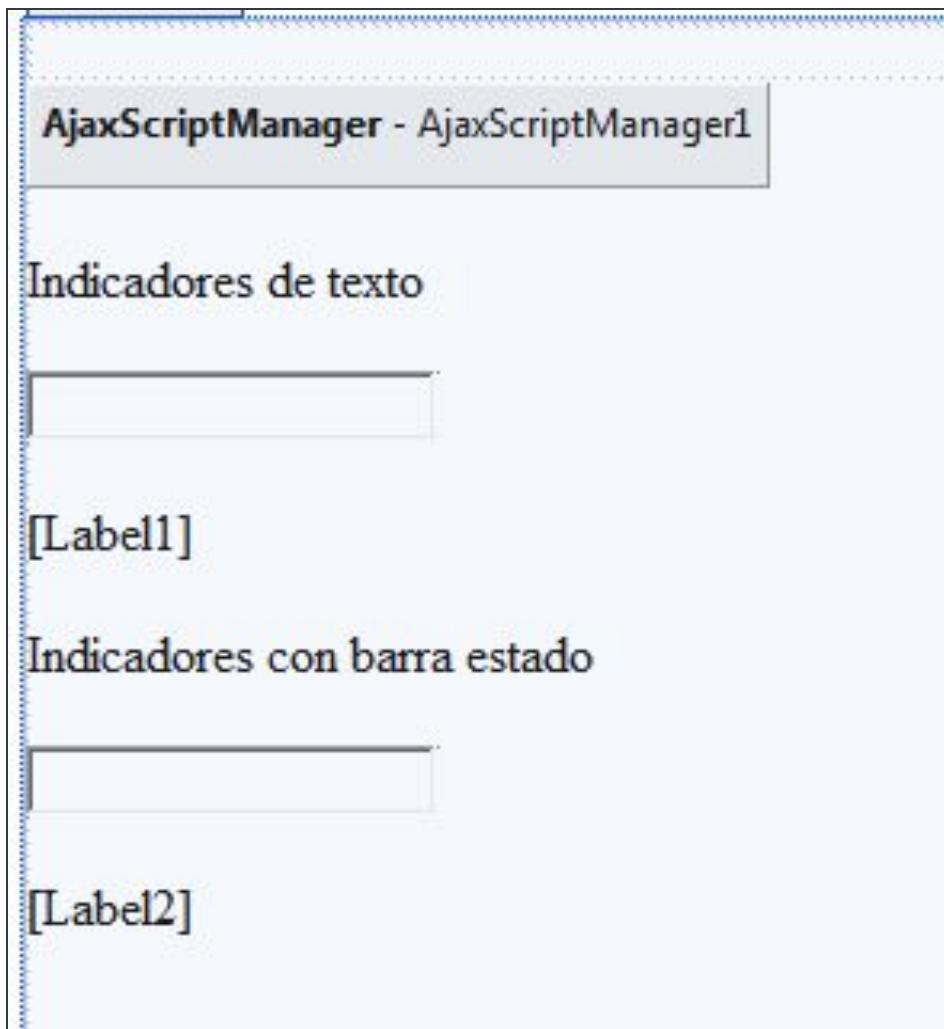
- El PasswordStrength es otro extendedor del TextBox.
- Con él podremos mostrar al usuario el nivel de fortaleza que tiene la contraseña que está escribiendo, en base a unos parámetros típicos de fortaleza que podemos configurar nosotros.



Ejercicio

- Crear un formulario web con los siguientes controles:
 - AjaxScriptManager
 - TextBox1
 - Label1
 - TextBox2
 - Label2
 - Extensor PasswordStrength a cada textbox

Diseño



Ejecución

Indicadores de texto

22

Seguridad: Muy baja

8 more characters

Indicadores con barra estado

12345hy@



2 more characters

Código (I)

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:PasswordStrength ID="TextBox1_PasswordStrength" runat="server"
    Enabled="True" TargetControlID="TextBox1"
    DisplayPosition="RightSide"
    StrengthIndicatorType="Text"
    PreferredPasswordLength="10"
    PrefixText="Seguridad:"
    TextStrengthDescriptions="Muybaja;Debil;Media;Fuerte;Excelente"
    MinimumLowerCaseCharacters="0"
    MinimumSymbolCharacters="0"
    HelpStatusLabelID="Label1"
    RequiresUpperAndLowerCaseCharacters="false">
</asp:PasswordStrength>
</p>
<asp:Label ID="Label1" runat="server"></asp:Label>
```

Código (II)

```
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
<asp:PasswordStrength ID="TextBox2_PasswordStrength" runat="server"
    Enabled="True" TargetControlID="TextBox2"
    DisplayPosition="RightSide"
    StrengthIndicatorType="BarIndicator"
    PreferredPasswordLength="10"
    PrefixText="Seguridad:"
    TextStrengthDescriptions="Muy baja;Dibil;Media;Fuerte;Excelente"
    MinimumNumericCharacters="1"
    MinimumSymbolCharacters="1"
    HelpStatusLabelID="Label2"
    RequiresUpperAndLowerCaseCharacters="true"
    BarBorderCssClass="barBorder"
    BarIndicatorCssClass="barInternal">
</asp:PasswordStrength>
```

CSS

```
<head>
  <style>
    .barBorder
    {
      border: solid 1px red;
      width: 300px;
    }
    .barInternal
    {
      background: yellow;
    }
  </style>
</head>
```

ConfirmButton

- Con el ConfirmButton conseguimos una sencilla funcionalidad. Lo asignaremos a un Button, LinkButton o HyperLink, de modo que cuando se haga clic sobre éste, el navegador nos muestre una ventana de confirmación.
 - Si se elige aceptar, el botón o enlace funciona normalmente.
 - De lo contrario, no se ejecutará el comportamiento definido. Opcionalmente se puede definir un script en el cliente llamándolo mediante la propiedad OnClientCancel.
- Útil para pedir confirmación al usuario al navegar a una página o borrar un link etc.

Código

```
<asp:Button ID="Button1" runat="server"
    Text="Button"
    onclick="Button1_Click" />

<asp:ConfirmButtonExtender
    ID="Button1_ConfirmButtonExtender"
    runat="server"
    ConfirmText="Seguro que deseas aceptar?"
    Enabled="True"
    TargetControlID="Button1">
</asp:ConfirmButtonExtender>
```

Al pinchar en el botón...

