(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 609 508**

(51) Int. CI.:

G06F 9/44 (2006.01)
G06F 9/48 (2006.01)
G06Q 10/10 (2012.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **29.08.2006 PCT/US2006/034122**

(87) Fecha y número de publicación internacional: **22.03.2007 WO07032926**

(96) Fecha de presentación y número de la solicitud europea: **29.08.2006 E 06802747 (3)**

(97) Fecha y número de publicación de la concesión europea: **12.10.2016 EP 1924927**

(54) Título: **Acciones de control definidas de forma declarativa**

(30) Prioridad:

12.09.2005 US 716562 P
17.01.2006 US 333870

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
20.04.2017

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

KOTHARI, NIKHIL y
SANABRIA, ANDRES, M.

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Acciones de control definidas de forma declarativa

Antecedentes

Las aplicaciones de soporte lógico informático se están desarrollando cada vez más de una forma declarativa para simplificar el proceso de desarrollo. En los marcos de trabajo de desarrollo de aplicaciones de usuario y de interfaz, el comportamiento de la lógica y no de interfaz de usuario está integrado en componentes de interfaz de usuario para ampliar las propiedades de control asociadas con los componentes de interfaz de usuario. El comportamiento no de interfaz de usuario se puede integrar en composiciones de componentes de interfaz de usuario. Los controles compuestos pueden ser muy grandes, con muchas propiedades diferentes, cuya modificación por parte de los usuarios para dar cabida a diferentes necesidades es difícil. El gran número de propiedades complica la personalización del control debido a que el usuario puede no estar familiarizado con la totalidad de las propiedades asociadas. El modelo de objeto se sobredimensiona de tal modo que, cuando el control está incluido en un formulario, los usuarios se pueden ver abrumados por la funcionalidad que es ofrecida por el control.

El documento EP1 403 765 A se refiere a un sistema de manejo de eventos que comprende un modelo, una vista y un controlador. La vista presenta el modelo a un usuario y genera un evento de sistema. El controlador manipula el modelo en respuesta al evento de sistema. La vista asigna el evento de sistema a una acción que es proporcionada por el controlador. El controlador realiza una llamada a un código de aplicación a través de una función de manejo de eventos que está asignada a la acción. Es expone que los controladores de vista de modelo de la técnica anterior permiten sustituir una vista sin modificación adicional del controlador con respecto al enlace de datos, pero no con respecto al manejo de eventos. Por lo tanto, se propone un sistema en el que un evento de sistema está desacoplado del evento de aplicación correspondiente. Dicho de otra forma, no es necesario que el manejador de eventos de sistema implemente una lógica de aplicación de manejo de eventos, sino que simplemente invoca el manejador de eventos de aplicación correspondiente. Esto permite que un desarrollador de aplicaciones sustituya vistas en el patrón de diseño de controladores de vista de modelo ampliado sin modificar el controlador o modelo correspondiente.

Sumario

El objeto de la presente invención es proporcionar un marco de trabajo de diseño de control ampliable para definir de forma declarativa un control (tanto visual como no visual) y otros componentes.

Este objeto se soluciona mediante la materia objeto de las reivindicaciones independientes.

En las reivindicaciones dependientes se dan realizaciones.

La funcionalidad de un control se puede ampliar añadiendo de forma declarativa unos comportamientos y una semántica por medio de un objeto que se denomina acción. El control no se vuelve excesivamente complejo por la adición de los comportamientos y de la semántica. El objeto de acción encapsula una funcionalidad bien definida y está asociado con un control existente. El objeto de acción también está asociado con un evento o con algún otro desencadenador, de tal modo que la funcionalidad se ejecuta de forma automática cuando se genera el evento. La funcionalidad se puede empaquetar en un componente de aplicación independiente, de tal modo que un usuario que no está familiarizado con el código de programación puede definir con facilidad una lógica y una funcionalidad para una aplicación en un entorno de diseño.

El presente Sumario se proporciona para introducir una selección de conceptos de una forma simplificada que se describe adicionalmente en lo sucesivo en la Descripción Detallada. El presente Sumario no tiene como objeto identificar características clave o características esenciales de la materia objeto que se reivindica, y no tiene como objeto usarse como una ayuda en la determinación del ámbito de la materia objeto que se reivindica.

Breve descripción de los dibujos

La figura 1 es un diagrama de arquitectura de sistema informático que ilustra un sistema informático que se usa en y que es proporcionado por las diversas realizaciones de la invención.
La figura 2 es un componente de aplicación a modo de ejemplo que implementa unos controles definidos de forma declarativa.
La figura 3 es un componente de aplicación a modo de ejemplo que implementa unos controles definidos de forma declarativa.
La figura 4 es un componente de aplicación a modo de ejemplo que implementa unos controles definidos de forma declarativa.
La figura 5 es un componente de aplicación a modo de ejemplo que implementa unos controles definidos de forma declarativa.
La figura 6 es un diagrama de flujo operativo que ilustra un procedimiento para unas acciones de control definidas de forma declarativa.
La figura 7 es un código a modo de ejemplo que muestra cómo se implementa el marco de trabajo de diseño de

control en un diseñador de control.
La figura 8 es un código a modo de ejemplo que muestra cómo un control básico puede proporcionar una funcionalidad para definir de forma declarativa unas acciones de control.

Descripción detallada

A continuación se describirán más plenamente unas realizaciones de la presente divulgación con referencia en lo sucesivo en el presente documento a los dibujos adjuntos, los cuales forman parte de la misma, y los cuales muestran, a modo de ilustración, unas realizaciones a modo de ejemplo específicas para poner en práctica la invención. No obstante, la presente divulgación se puede materializar de muchas formas diferentes y no se debería interpretar como limitada a las realizaciones que se exponen en el presente documento; en su lugar, estas realizaciones se proporcionan de tal modo que la presente divulgación sea exhaustiva y completa, y transmita en su totalidad el ámbito a los expertos en la materia. Entre otras cosas, la presente divulgación se puede materializar como procedimientos o dispositivos. Por consiguiente, la presente divulgación puede adoptar la forma de una realización en su totalidad en soporte físico, una realización en su totalidad en soporte lógico o una realización que combina unos aspectos de soporte lógico y de soporte físico. Por lo tanto, la siguiente descripción detallada no se ha de tomar en un sentido limitante.

Haciendo referencia a continuación a los dibujos, en los que números similares representan elementos similares, se describirán diversos aspectos de la presente invención. En particular, la figura 1 y el análisis correspondiente tienen como objeto proporcionar una descripción breve y general de un entorno informático adecuado en el que se puedan implementar unas realizaciones de la invención. A pesar de que la invención se describirá en el contexto general de los módulos de programa que se ejecutan en sistemas informáticos de servidor y personales, los expertos en la materia reconocerán que la invención también se puede implementar en combinación con otros tipos de sistemas informáticos y de módulos de programa.

En general, los módulos de programa incluyen rutinas, programas, componentes, estructuras de datos y otros tipos de estructuras que realizan tareas particulares o que implementan tipos de datos abstractos particulares. Además, los expertos en la materia apreciarán que la invención se puede poner en práctica con otras configuraciones de sistemas informáticos, que incluyen dispositivos de mano, sistemas de múltiples procesadores, electrónica de consumo programable o basada en microprocesadores, miniordenadores, ordenadores centrales, y similares. La invención también se puede poner en práctica en entornos informáticos distribuidos, en los que las tareas son realizadas por dispositivos de procesamiento remotos que están enlazados a través de una red de comunicaciones. En un entorno informático distribuido, los módulos de programa pueden estar ubicados en dispositivos de almacenamiento de memoria tanto locales como remotos.

Entorno operativo ilustrativo

Haciendo referencia a continuación a la figura 1, se describirá una arquitectura informática ilustrativa para un ordenador 100 que se usa en las diversas realizaciones de la invención. La arquitectura informática que se muestra en la figura 1 ilustra un ordenador de escritorio o portátil convencional, que incluye una unidad de procesamiento central 110 (“CPU”, *central processing unit*), una memoria de sistema 120, que incluye una memoria de acceso aleatorio (“RAM”, *random access memory*) 120 y una memoria de solo lectura (“ROM”, *read only memory*) 124, y un bus de sistema 130, que acopla la memoria a la CPU 110. En la ROM 124 está almacenado un sistema básico de entrada/salida que contiene las rutinas básicas que ayudan a transferir información entre los elementos dentro del ordenador, tales como durante el arranque. El ordenador 100 incluye adicionalmente un dispositivo de almacenamiento en masa 140 para almacenar un sistema operativo 142, programas de aplicación y otros módulos de programa, los cuales se describirán con mayor detalle en lo sucesivo.

El dispositivo de almacenamiento en masa 140 está conectado a la CPU 110 a través de un controlador de almacenamiento en masa (que no se muestra) que está conectado al bus 130. El dispositivo de almacenamiento en masa 140 y sus medios legibles por ordenador asociados proporcionan un almacenamiento no volátil para el ordenador 100. A pesar de que la descripción de los medios legibles por ordenador que está contenida en el presente documento se refiere a un dispositivo de almacenamiento en masa, tal como un disco duro o una unidad de CD-ROM, los expertos en la materia deberán apreciar que los medios legibles por ordenador pueden ser cualquier medio disponible al que se pueda acceder por medio del ordenador 100.

A modo de ejemplo, y no de limitación, los medios legibles por ordenador pueden comprender medios de almacenamiento informático y medios de comunicación. Los medios de almacenamiento informático incluyen medios volátiles y no volátiles, extraíbles y no extraíbles, que están implementados en cualquier procedimiento o tecnología para el almacenamiento de información, tales como instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos. Los medios de almacenamiento informático incluyen, pero no se limitan a, RAM, ROM, EPROM, EEPROM, memoria flash u otra tecnología de memoria de estado sólido, CD-ROM, discos versátiles digitales (“DVD”, *digital versatile disk*) u otro almacenamiento óptico, casetes magnéticos, cinta magnética, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que se pueda usar para almacenar la información deseada y al cual se pueda acceder por medio del ordenador 100.

De acuerdo con diversas realizaciones de la invención, el ordenador 100 puede operar en un entorno de red usando conexiones lógicas a ordenadores remotos a través de una red 150, tal como Internet. El ordenador 100 se puede conectar a la red 150 a través de una unidad de interfaz de red 160 que está conectada al bus 130. La unidad de interfaz de red 160 también se puede usar para conectarse a otros tipos de redes y de sistemas informáticos remotos. El ordenador 100 puede también incluir un controlador de entrada/salida 170 para recibir y procesar entradas de datos a partir de un número de otros dispositivos, que incluyen un teclado, un ratón, un lápiz electrónico (que no se muestran en la figura 1). De forma similar, el controlador de entrada/salida 170 puede proporcionar una salida a una pantalla de presentación visual, una impresora u otro tipo de dispositivo de salida.

Tal como se ha mencionado brevemente en lo que antecede, un número de módulos de programa y de archivos de datos se pueden almacenar en un dispositivo de almacenamiento en masa 140 y una RAM 122 del ordenador 100, incluyendo una aplicación 148 y el sistema operativo 142 que es adecuado para controlar el funcionamiento de un ordenador personal en red, tal como el sistema operativo WINDOWS XP de MICROSOFT CORPORATION de Redmond, Washington. El dispositivo de almacenamiento en masa 140 y la RAM 122 también pueden almacenar uno o más módulos de programa. En particular, el dispositivo de almacenamiento en masa 140 y la RAM 122 pueden almacenar un programa de aplicación de explorador web 144. El programa de aplicación de explorador web 144 se puede operar para solicitar, recibir, presentar y proporcionar interactividad con documentos electrónicos, tales como una página web 146, a la que se ha dado formato usando HTML. De acuerdo con una realización de la invención, el programa de aplicación de explorador web 144 comprende el programa de aplicación de explorador web INTERNET EXPLORER de MICROSOFT CORPORATION. No obstante, se debería apreciar que también se pueden usar otros programas de aplicación de explorador web facilitados por otros fabricantes para materializar los diversos aspectos de la presente invención, tales como la aplicación de explorador web FIREFOX facilitada por la FUNDACIÓN MOZILLA.

Acciones de control definidas de forma declarativa

Un marco de trabajo de diseño de control ampliable se usa para definir de forma declarativa un control (tanto visual como no visual) y otros componentes. La funcionalidad de un control se puede ampliar añadiendo de forma declarativa unos comportamientos y una semántica por medio de un objeto que se denomina acción. El control no se vuelve excesivamente complejo por la adición de los comportamientos y de la semántica. El objeto de acción encapsula una funcionalidad bien definida y está asociado con un control existente. El objeto de acción también está asociado con un evento o algún otro desencadenador, de tal modo que la funcionalidad se ejecuta de forma automática cuando se genera el evento. La funcionalidad se puede empaquetar en un componente de aplicación independiente, de tal modo que un usuario que no está familiarizado con el código de programación puede definir con facilidad una lógica y una funcionalidad para una aplicación en un entorno de diseño.

El marco de trabajo simplifica el desarrollo rápido de aplicaciones (RAD, *rapid application development*). Un diseñador puede seleccionar y colocar los componentes de aplicación usando una superficie de diseño visual. Los componentes de aplicación se pueden conectar usando la superficie de diseño visual. Los componentes de aplicación también se pueden personalizar con unas propiedades por medio de la escritura de un código para manejar eventos y para realizar operaciones relacionadas con el evento. Los componentes de aplicación se pueden invocar de forma automática cuando se genera el evento. El marco de trabajo se puede aplicar a otros entornos de diseño que incluyen diseño web, programación basada en secuencias de comandos, programación basada en formularios, y similares.

El marco de trabajo de diseño de control se explica con referencia a un evento de clic que está asociado con un control de botón. Se debería apreciar que, a pesar de que las realizaciones que se describen en el presente documento se presentan en el contexto de un evento de clic que está asociado con un control de botón, se puede usar cualquier evento o desencadenador que se defina en las interfaces de control.

El marco de trabajo de diseño de control empaqueta la funcionalidad de control en componentes. Los componentes se asocian usando acciones y comportamientos declarativos.

La figura 2 ilustra un componente a modo de ejemplo que implementa unos controles definidos de forma declarativa. Un desarrollador puede crear un componente de inicio de sesión 220 como un control compuesto en un formulario con unos controles de cuadro de texto individuales 210, 212 y un control de botón 220. Cada control dentro del componente de inicio de sesión 200 es una entidad independiente a la que se puede acceder de forma singular en el formulario, de tal modo que los controles se pueden personalizar y recolocar. Un usuario puede introducir valores en los cuadros de texto 210, 212 y hacer clic en el control de botón 220. El evento de clic que se genera en el control de botón 220 está asociado con un procedimiento de inicio de sesión 230.

El procedimiento de inicio de sesión 230, incluye un código a modo de ejemplo para definir de forma declarativa unos controles. El procedimiento de inicio de sesión 230 extrae, de los controles del cuadro de texto 210, 212, unos valores cuando se genera el evento de clic que está asociado con el control de botón 220. El procedimiento de inicio de sesión 230 realiza una lógica sobre los valores y la interfaz de usuario se actualiza de acuerdo con el resultado de la operación lógica de inicio de sesión (por ejemplo, se concede o se deniega el inicio de sesión).

El marco de trabajo de diseño de control permite que se definan de forma declarativa diferentes escenarios de aplicación sin crear unos controles personalizados para cada escenario específico. Haciendo referencia al código a modo de ejemplo en el procedimiento de inicio de sesión 230, se implementa un escenario de inicio de sesión sin un control de inicio de sesión. El componente de inicio de sesión 200 se define con unos controles tales como dos cuadros de texto 240, 250 y un botón 260. Los cuadros de texto 240, 250 y el botón 260 se pueden personalizar y recolocar de forma independiente debido a que los controles no están limitados a una estructura de formulario particular. Cada cuadro de texto 240, 250, tiene una propiedad de texto asociada 265, 270 (por ejemplo, un nombre, una contraseña), y el botón 260también está asociado con una propiedad de texto 275 (por ejemplo, Haga Clic Aquí).

El botón 260 está asociado con unas ClickActions (acciones de clic) 280. Las ClickActions 280 incluyen una LoginAction (acción de inicio de sesión) 285. La LoginAction 285 es un componente de inicio de sesión que se puede personalizar (por ejemplo, un nombre y una contraseña). La LoginAction 285 se desencadena de forma automática cuando se genera el evento de clic debido a que la LoginAction 285 está incluida en las ClickActions 280. El componente de inicio de sesión definido de forma declarativa 200 tiene la misma funcionalidad que un control de inicio de sesión que se define por medio de un código escrito. El control definido de forma declarativa se beneficia de la flexibilidad añadida de programar usando unos controles de bloque de creación en una superficie de diseño visual. Si es necesario, un desarrollador puede personalizar adicionalmente los controles en la superficie de diseño visual.

La figura 3 ilustra cómo el marco de trabajo de diseño de control adjunta comportamiento adicional a un control. Se describe la anexión de comportamiento adicional con referencia a un botón 310 en una interfaz de usuario 300. La interfaz de usuario 300 puede también incluir unos cuadros de texto 305 para que un usuario introduzca valores de datos. Los cuadros de texto 305 se pueden agrupar (por ejemplo, los cuadros de texto 305 se agrupan y se nombran “Grupo1”). El botón 310 está asociado con un procedimiento de ampliación de propiedad 320.

El procedimiento de ampliación de propiedad 320 define un control de botón simple 330 con una propiedad de texto 340. Cuando se hace clic en el botón 310 se genera un evento de clic (por ejemplo, una ValidationAction (acción de validación) 350, una CounterAction (acción de contador) 360 y una WorkflowAction (acción de flujo de trabajo) 370).

La semántica de aplicación se puede adjuntar al botón 310 de una forma ampliable, de tal modo que las propiedades de control se puedan cambiar con facilidad. Las propiedades de control pueden desencadenar o invocar cualquier lógica de aplicación (imperativa o declarativa) según se haya configurado. Por ejemplo, la ValidationAction 350, la CounterAction 360 y la WorkflowAction 370 se pueden realizar cuando se hace clic en el botón 310. La ValidationAction 350 valida los valores de datos en los cuadros de texto 305 que se identifican como el Grupo1 en la interfaz de usuario 300 cuando se hace clic en el botón 310. La CounterAction 360 mantiene un recuento del número de veces que se ha hecho clic en el botón 310. La WorkflowAction 370 inicia o reanuda la ejecución de cualquier lógica (imperativa o declarativa) que esté asociada con una aplicación (tal como un flujo de trabajo) cuando se hace clic en el botón 310. Por lo tanto, se adjuntan de forma declarativa propiedades adicionales al botón 310 sin complicar un modelo de objeto existente o solicitar la creación de un modelo de objeto nuevo.

La figura 4 ilustra cómo el marco de trabajo de diseño de control proporciona una capacidad de descubrimiento y un enfoque con establecimiento inflexible de tipos a un control. Un control de FormView (vista de formulario) está asociado con el contenido a partir de una fila de base de datos. Un control de botón que está asociado con el control de FormView realiza unas operaciones de la base de datos en la fila. Por ejemplo, se puede asociar un botón de “Update” (actualizar) con el control de FormView tal como se muestra en lo sucesivo:

```
<asp:Button CommandName = “Update”/>
```

en el que “Update” es una propiedad de cadena.

El control de FormView busca de forma específica un nombre de comando que se denomina “Update”. La lógica de actualización se ejecuta cuando se hace clic en el botón. No obstante, la lógica de actualización no se puede descubrir en los marcos de trabajo de control actuales. Dicho de otra forma, un desarrollador que usa el control de FormView en un programa puede no conocer los subcomandos asociados con el control de FormView. Además, no queda claro si el nombre del comando es sensible a mayúsculas y minúsculas. Por ejemplo, el control de FormView puede buscar, de forma específica, “Update”, de tal modo que “update” puede dar como resultado un error y se puede pasar por alto. Además, un error tipográfico puede dar lugar a que falle el formulario. Por ejemplo, puede que la lógica de actualización no se realice cuando se hace clic en el botón si se busca “Update1” en lugar de “Update”. Los marcos de trabajo de programación de control conocidos requieren que un usuario explore el código para localizar el error tipográfico.

El marco de trabajo de diseño de control proporciona un enfoque con establecimiento inflexible de tipos para los controles de programación. La expresión “con establecimiento inflexible de tipos” se refiere a nombrar y teclear de forma explícita los procedimientos y los miembros variables de una clase. El tipo de datos se conoce en tiempo de compilación y, en ese sentido, permite que las asignaciones de tipos de datos incorrectas se resuelvan en tiempo de compilación en lugar de en tiempo de ejecución, ahorrando procesamiento y disminuyendo la posibilidad de error en los códigos. El marco de trabajo de diseño de control también proporciona varias acciones integradas. Tal como se

ha analizado en lo que antecede, los desarrolladores de aplicaciones y terceras partes pueden desarrollar nuevas acciones que sean específicas para escenarios particulares. Las acciones se deberían poder descubrir en el diseñador de control, u otros usuarios no familiarizados con las acciones recién definidas no sabrían que estas existen.

La capacidad de descubrimiento y el enfoque con establecimiento inflexible de tipos se describen con referencia a un botón 410 en una interfaz de usuario 400. La interfaz de usuario 400 también puede incluir un cuadro de texto 405 para que el usuario introduzca un valor de datos tal como “artículo”. El botón 410 está asociado con un procedimiento de capacidad de descubrimiento y de enfoque con establecimiento inflexible de tipos 420. El procedimiento de capacidad de descubrimiento y de enfoque con establecimiento inflexible de tipos 420 define un control de FormView 430 y un control de botón simple 440 con una propiedad de texto 445.

Un desarrollador no necesita conocer el conjunto específico de cadenas que son usadas por el control de FormView 430. En su lugar, el modelo de objeto de una acción asociada proporciona esa información. El control de botón 440 no incluye un nombre de comando. El control de botón 440 incluye una FormViewAction (acción de vista de formulario) 450 dentro de las ClickActions 460 que están asociadas con el botón 440. La FormViewAction 450 incluye una propiedad de Operation = “Edit” (Operación = “Editar”) 470. La propiedad de Operation = “Edit” 470 es un tipo enumerado (en lugar de una cadena) de tal modo que la propiedad está asociada con un conjunto fijo de valores (por ejemplo, editar, suprimir, actualizar, etc.). Los tipos enumerados permiten que un editor y un analizador exijan un valor válido para la enumeración. Por ejemplo, un desarrollador puede teclear “Operation =“. Entonces se puede presentar visualmente un menú desplegable 480 con los diferentes valores de la enumeración, de tal modo que el usuario puede seleccionar un valor válido. Por lo tanto, un desarrollador puede descubrir los subcomandos que están asociados con un control.

El desarrollador puede introducir un error tipográfico cuando se teclea la propiedad 470. Por ejemplo, el desarrollador puede haber tecleado “Edit1” o “edit” en lugar de “Edit”. Cuando se analiza la página, “Edit1” da como resultado un error para el analizador debido a que “Edit1” es un valor no conocido. “Edit” también puede dar como resultado un error para el analizador si la función de búsqueda es sensible a mayúsculas y minúsculas. Podría no darse como resultado un error si el valor fuera una cadena. En su lugar, un evento de clic que se genera en el botón 410 podría no dar como resultado acción alguna. Por lo tanto, la propiedad de tipo enumerado es un valor con establecimiento inflexible de tipos que elimina una clase de errores que se producen por las equivocaciones de tecleo de los desarrolladores.

La figura 5 ilustra cómo se puede usar el marco de trabajo de diseño de control para aplicar un efecto a un control. Una interfaz de usuario 500 está dispuesta para recibir la entrada de datos para realizar una búsqueda. Un cuadro de texto 510 está dispuesto para recibir términos de búsqueda. El usuario puede hacer clic en un botón de búsqueda 520 para iniciar la búsqueda. Los resultados se presentan visualmente como una lista de búsqueda en un panel 530.

El botón de búsqueda 520 se puede asociar con un procedimiento de aplicación de efecto 540, de tal modo que se puede aplicar un efecto a la lista de búsqueda cuando se presenta visualmente en el panel 530. Por ejemplo, se puede aplicar un efecto de animación de aparición progresiva tal como un FadeInEffect (efecto de aparición progresiva) 550 a un objeto objetivo 560, tal como la lista de búsqueda. La lista de búsqueda aparece de forma progresiva cuando se presenta visualmente en el panel 530. El FadeInEffect 550 incluye un procedimiento de reproducción 570 que es invocado por el evento de clic 580. El evento de clic 580 también acopla una propiedad 590 al panel 530, de tal modo que el panel 530 se hace visible. Cuando se hace clic en el botón de búsqueda 520, se aplica el FadeInEffect 550 al panel 530 y el panel 530 se hace visible. Por lo tanto, se puede definir de forma declarativa un procedimiento mediante la invocación del procedimiento y la definición de un objeto particular.

La figura 6 es un diagrama de flujo operativo que ilustra un proceso para definir de forma declarativa unas acciones de control. El proceso comienza en una operación de inicio en la que se dispone un control para realizar una función.

El procesamiento continúa en la operación 600, en la que una acción se asocia con un control. Por ejemplo, el control puede ser un control de botón, un control de cuadro de texto o un control de vista de formulario. La acción encapsula unos comportamientos y una semántica, de tal modo que se amplía la funcionalidad del control. La acción es un componente de aplicación independiente, de tal modo que el control no se vuelve excesivamente complejo cuando se asocia la acción con el control. El componente de aplicación independiente permite que la acción se defina como un bloque de creación en un entorno de diseño de aplicación. Un desarrollador puede desarrollar con facilidad una aplicación al ensamblar los bloques de creación para definir una lógica y una funcionalidad para la aplicación. En una realización, la acción incluye una propiedad enumerada que identifica un conjunto fijo de valores. El conjunto fijo de valores se puede presentar visualmente a un desarrollador cuando se teclea la propiedad. Si el desarrollador teclea de forma errónea un valor de propiedad, se obtiene como resultado un error debido a que el valor tecleado de forma errónea no está incluido en el conjunto fijo. En otra realización, se puede asociar más de un control con una acción. Por ejemplo, una acción se puede anidar dentro de otra acción. Todavía en otra realización, se pueden asociar unas propiedades adicionales con el control.

Avanzando a la operación 610, se asocia un evento con el control. Por ejemplo, si el control es un botón, el evento asociado puede ser un evento de clic que se genera en el botón. Avanzando a la operación 620, el control detecta

que se va a generar el evento asociado. En una realización, el control detecta que el evento asociado se va a generar cuando se recibe un valor en el control.

Pasando a la operación 630, la acción se ejecuta cuando se genera el evento. La acción se puede ejecutar mediante la realización de una operación lógica sobre el valor. Los ejemplos de ejecución de la acción incluyen el procesamiento de valores, la validación de valores, el recuento del número de veces que se genera un evento, el inicio de la ejecución de un flujo de trabajo, la reanudación de la ejecución de un flujo de trabajo, el descubrimiento de propiedades asociadas con un control, la resolución de asignaciones de tipos de datos incorrectas en tiempo de compilación, la aplicación de un efecto a un control y similares.

También se pueden ejecutar otras acciones independientes que estén asociadas con el evento. En una realización, se repite el proceso para ejecutar una secuencia de acciones. Una acción en la secuencia se puede ejecutar y retornar un valor “falso”, interrumpiendo de ese modo la secuencia de acciones. El procesamiento termina entonces en una operación de fin.

La figura 7 es un segmento de código a modo de ejemplo 700 que muestra cómo se implementa el marco de trabajo de diseño de control en un diseñador de control. El diseñador de control puede ofrecer de forma automática un editor de acciones para los controles que soportan acciones por medio de un panel de tareas. El editor de acciones presenta el conjunto de acciones disponibles en los conjuntos a los que hace referencia la aplicación. Las acciones proporcionan una funcionalidad de diseñador asociada para proporcionar una experiencia de edición rica más allá de una red de propiedades. El segmento de código a modo de ejemplo 700 indica que una acción se puede ejecutar antes o después de que tenga lugar un evento.

Se proporciona una clase de base para todas las acciones de control. Cada acción puede especificar si la acción se ejecuta antes o después de que se genere el evento. El valor de retorno es significativo para las acciones que se ejecutan antes de que se genere el evento. La acción podría retornar “verdadero” para cancelar la posterior generación del evento. Una acción se puede usar en la personalización del comportamiento de devolución. Los controles pueden realizar una llamada a acciones que implementan una interfaz para las acciones que están asociadas con eventos relacionados con la devolución.

La figura 8 es un segmento de código a modo de ejemplo 800 que muestra cómo un control de botón básico puede proporcionar una funcionalidad para definir de forma declarativa unas acciones de control.

La memoria descriptiva, ejemplos y datos anteriores proporcionan una descripción completa de la fabricación y el uso de la composición de la invención. Debido a que se pueden realizar muchas realizaciones de la invención sin apartarse del ámbito de la invención, la invención reside en las reivindicaciones que se adjuntan en lo sucesivo en el presente documento.

REIVINDICACIONES

1. Un procedimiento implementado por ordenador para definir de forma declarativa una acción de control, comprendiendo el procedimiento:

asociar (600) una primera acción con un control (200, 300, 400, 500) dentro de un código (230, 320, 420, 540) para definir de forma declarativa el control, en el que la primera acción y el control son unos componentes que se pueden personalizar de forma independiente;
asociar (610) un evento con el control;
recibir un valor en el control;
ejecutar (630) la primera acción cuando se genera el evento (620), en el que ejecutar la primera acción comprende realizar una operación lógica sobre el valor;
asociar (610) una segunda acción con la primera acción y el control dentro del código, en el que la primera acción, la segunda acción y el control son unos componentes que se pueden personalizar de forma independiente, en el que al menos una de la primera y la segunda acciones está asociada con una propiedad de tipo enumerado (470) que tiene un conjunto fijo de valores que se presentan visualmente en un menú desplegable (480) dentro del código para asegurar que una selección de usuario de uno de los valores de entre el conjunto fijo de valores de la propiedad de tipo enumerado da lugar a que una selección de valor válida se asocie con las al menos una primera y una segunda acciones; y
ejecutar (630) la segunda acción cuando se genera el evento.

2. El procedimiento implementado por ordenador de la reivindicación 1, en el que la operación lógica sobre el valor es específica de una aplicación que está asociada con el control.

3. El procedimiento implementado por ordenador de la reivindicación 1, en el que ejecutar la primera o la segunda acción comprende adicionalmente determinar un número de veces que se genera el evento (360).

4. El procedimiento implementado por ordenador de la reivindicación 1, en el que ejecutar la primera o la segunda acción comprende adicionalmente empezar la ejecución de una lógica que está asociada con una aplicación (285).

5. El procedimiento implementado por ordenador de la reivindicación 1, en el que ejecutar la primera o la segunda acción comprende adicionalmente reanudar la ejecución de una lógica que está asociada con una aplicación (370).

6. El procedimiento implementado por ordenador de la reivindicación 1, en el que ejecutar la primera o la segunda acción comprende adicionalmente aplicar un efecto al control (550).

7. El procedimiento implementado por ordenador de la reivindicación 1, en el que detectar que el evento se va a generar comprende adicionalmente recibir un valor en el control.

8. El procedimiento implementado por ordenador de la reivindicación 7, en el que ejecutar la primera o la segunda acción comprende adicionalmente validar el valor (350).

9. El procedimiento implementado por ordenador de la reivindicación 7, en el que ejecutar la primera o la segunda acción comprende adicionalmente resolver una asignación de datos incorrecta que está asociada con el valor (420).

10. El procedimiento implementado por ordenador de la reivindicación 1, en el que al menos la primera acción se ejecuta antes de que se genere el evento (700).

11. El procedimiento implementado por ordenador de la reivindicación 1, en el que al menos la segunda acción se ejecuta después de que se haya generado el evento (700).

12. Un sistema para definir de forma declarativa una acción de control, que comprende:

una primera acción (350) y una segunda acción (360) que está asociada con la primera acción (350) dentro de un código (230, 320, 420, 540) para definir de forma declarativa un control (200, 300, 400, 500), en el que al menos una de la primera y la segunda acciones está asociada con una propiedad de tipo enumerado (470) que tiene un conjunto fijo de valores que se presentan visualmente en un menú desplegable (480) dentro del código para asegurar que una selección de usuario de uno de los valores de entre el conjunto fijo de valores de la propiedad de tipo enumerado da lugar a que una selección de valor válida se asocie con las al menos una primera y una segunda acciones;
el control que está asociado con la primera acción (350) y la segunda acción (360) dentro del código, en el que la primera acción (350), la segunda acción (360) y el control son unos componentes que se pueden personalizar de forma independiente; y
un evento que está asociado con el control, en el que la primera acción (350) y la segunda acción (360) se ejecutan cuando se genera el evento.

13. El sistema de la reivindicación 12, en el que la segunda acción (360) está anidada dentro de la primera acción (350).

14. El sistema de la reivindicación 12, en el que la segunda acción (360) está asociada con unas propiedades de tipo enumerado, de tal modo que una selección de una de las propiedades de tipo enumerado da lugar a que una selección de propiedad válida se asocie con la segunda acción (360).

15. El sistema de la reivindicación 12, en el que la segunda acción (360) cancela la ejecución de una tercera acción (370) cuando se ejecuta la segunda acción (360).

16. Un medio legible por ordenador que tiene unas instrucciones ejecutables por ordenador para definir de forma declarativa una acción de control, comprendiendo las instrucciones:

asociar (600) una primera acción con un control (200, 300, 400, 500) dentro de un código (230, 320, 420, 540) para definir de forma declarativa el control, en el que la primera acción y el control son unos componentes que se pueden personalizar de forma independiente;
asociar (610) un evento con el control;
recibir un valor en el control;
ejecutar (630) la primera acción cuando se genera el evento (620), en el que ejecutar la primera acción comprende realizar una operación lógica sobre el valor;
asociar (610) una segunda acción con la primera acción y el control dentro del código, en el que la primera acción, la segunda acción y el control son unos componentes que se pueden personalizar de forma independiente;
asociar al menos una de la primera y la segunda acciones con una propiedad de tipo enumerado (470) que tiene un conjunto fijo de valores que se presentan visualmente en un menú desplegable (480) dentro del código para asegurar que una selección de usuario de uno de los valores de entre el conjunto fijo de valores de la propiedad de tipo enumerado da lugar a que una selección de valor válida se asocie con las al menos una primera y una segunda acciones; y
ejecutar (630) la segunda acción cuando se genera el evento.

17. El medio legible por ordenador de la reivindicación 16, en el que asociar la segunda acción con la primera acción comprende adicionalmente anidar la segunda acción dentro de la primera acción.

18. El medio legible por ordenador de la reivindicación 16, en el que ejecutar la primera acción comprende adicionalmente asociar una semántica lógica con el control, en el que la semántica lógica es específica de una aplicación que está asociada con el control.

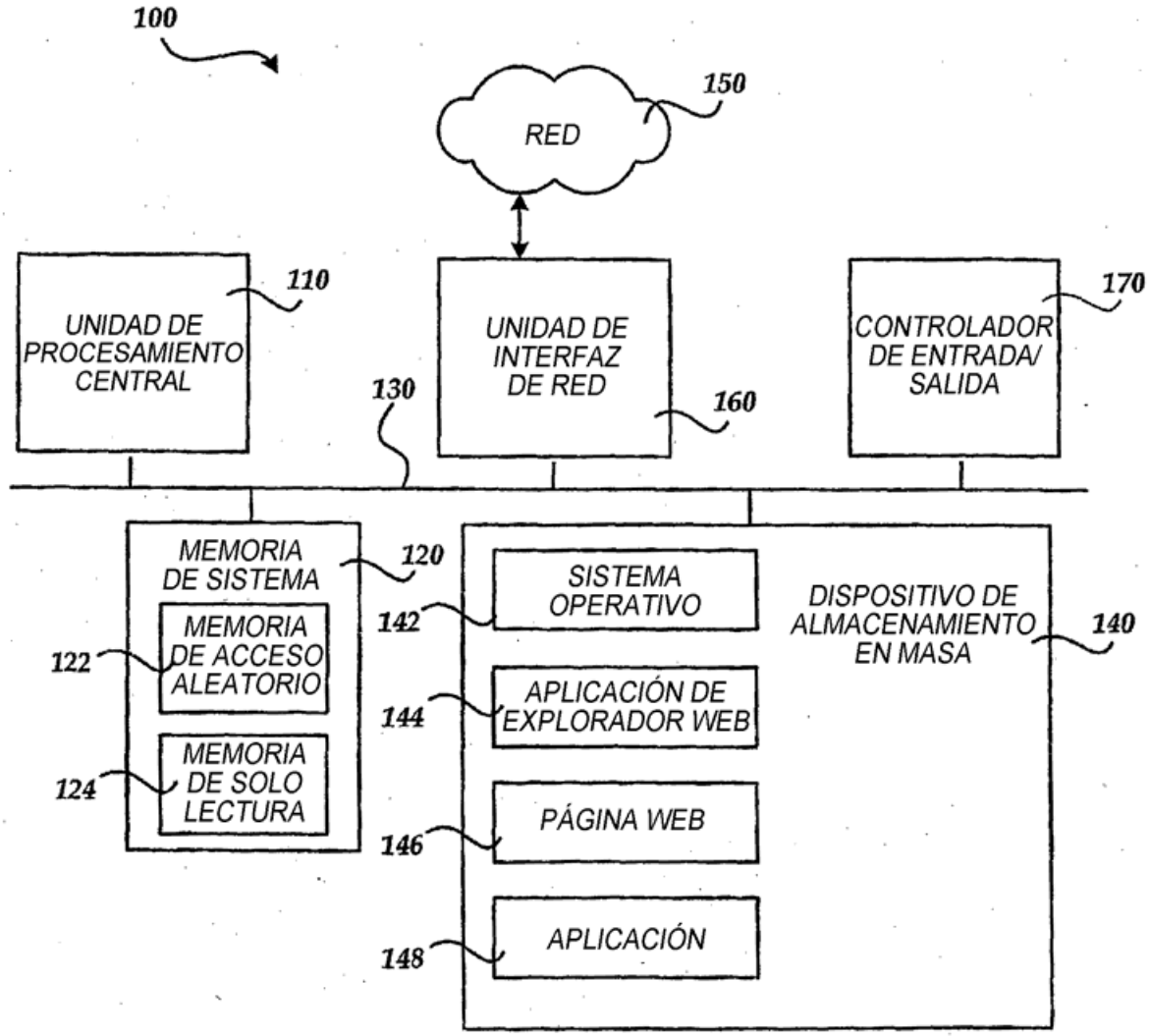
100
RED
150
UNIDAD DE PROCESAMIENTO CENTRAL
110
UNIDAD DE INTERFAZ DE RED
160
CONTROLADOR DE ENTRADA/ SALIDA
170
130
MEMORIA DE SISTEMA
120
MEMORIA DE ACCESO ALEATORIO
122
MEMORIA DE SOLO LECTURA
124
SISTEMA OPERATIVO
142
APLICACIÓN DE EXPLORADOR WEB
144
PÁGINA WEB
146
APLICACIÓN
148
DISPOSITIVO DE ALMACENAMIENTO EN MASA
140

Fig. 1

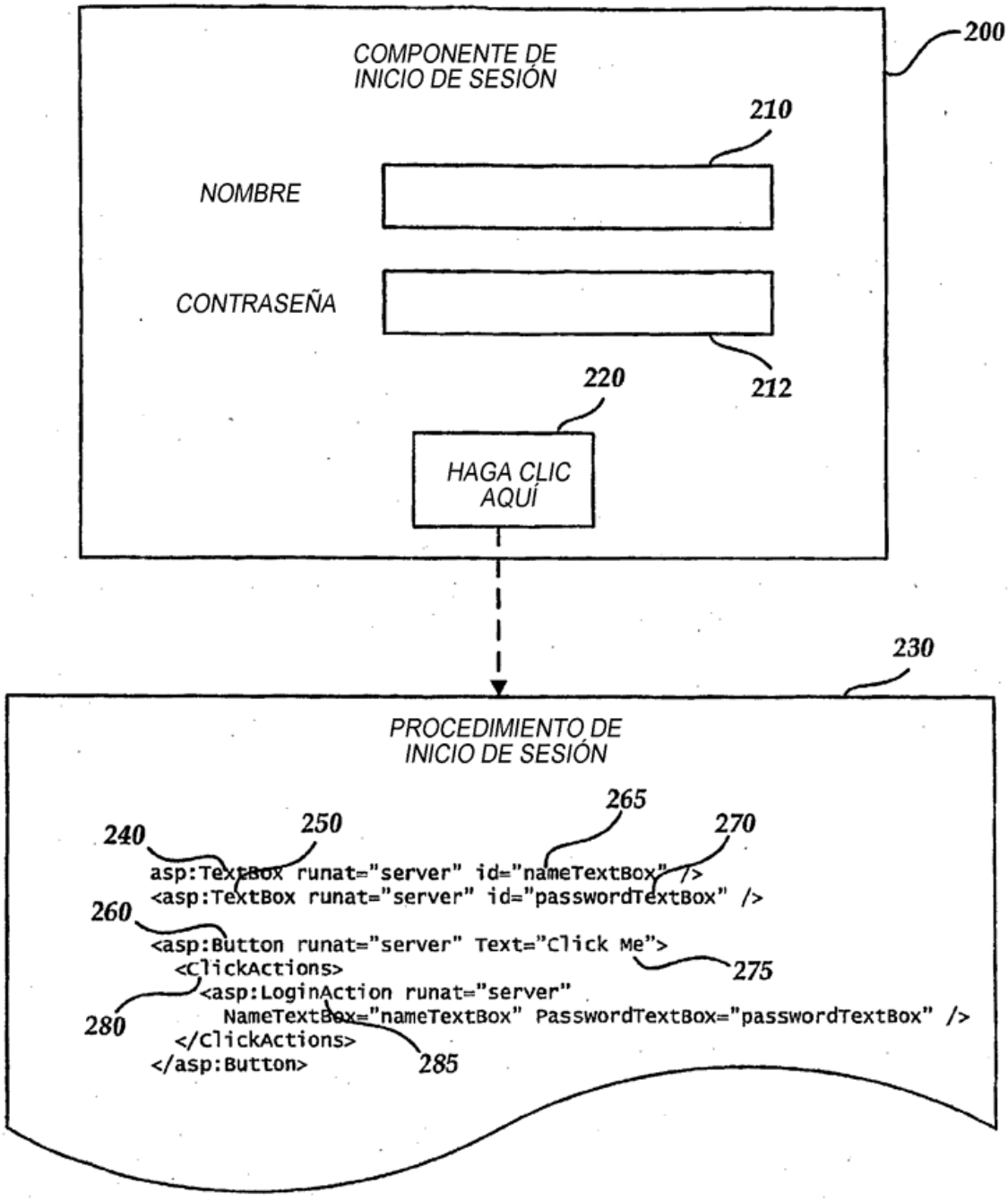
COMPONENTE DE INICIO DE SESIÓN
200
210
NOMBRE
CONTRASEÑA
212
220
HAGA CLIC AQUÍ
230
PROCEDIMIENTO DE INICIO DE SESIÓN
240
250
265
270
260
275
280
285
asp:TextBox runat="server" id="nameTextBox" />
<asp:TextBox runat="server" id="passwordTextBox" />
<asp:Button runat="server" Text="Click Me">
<ClickActions>
<asp:LoginAction runat="server"
NameTextBox="nameTextBox" PasswordTextBox="passwordTextBox" />
</ClickActions>
</asp:Button>

Fig. 2

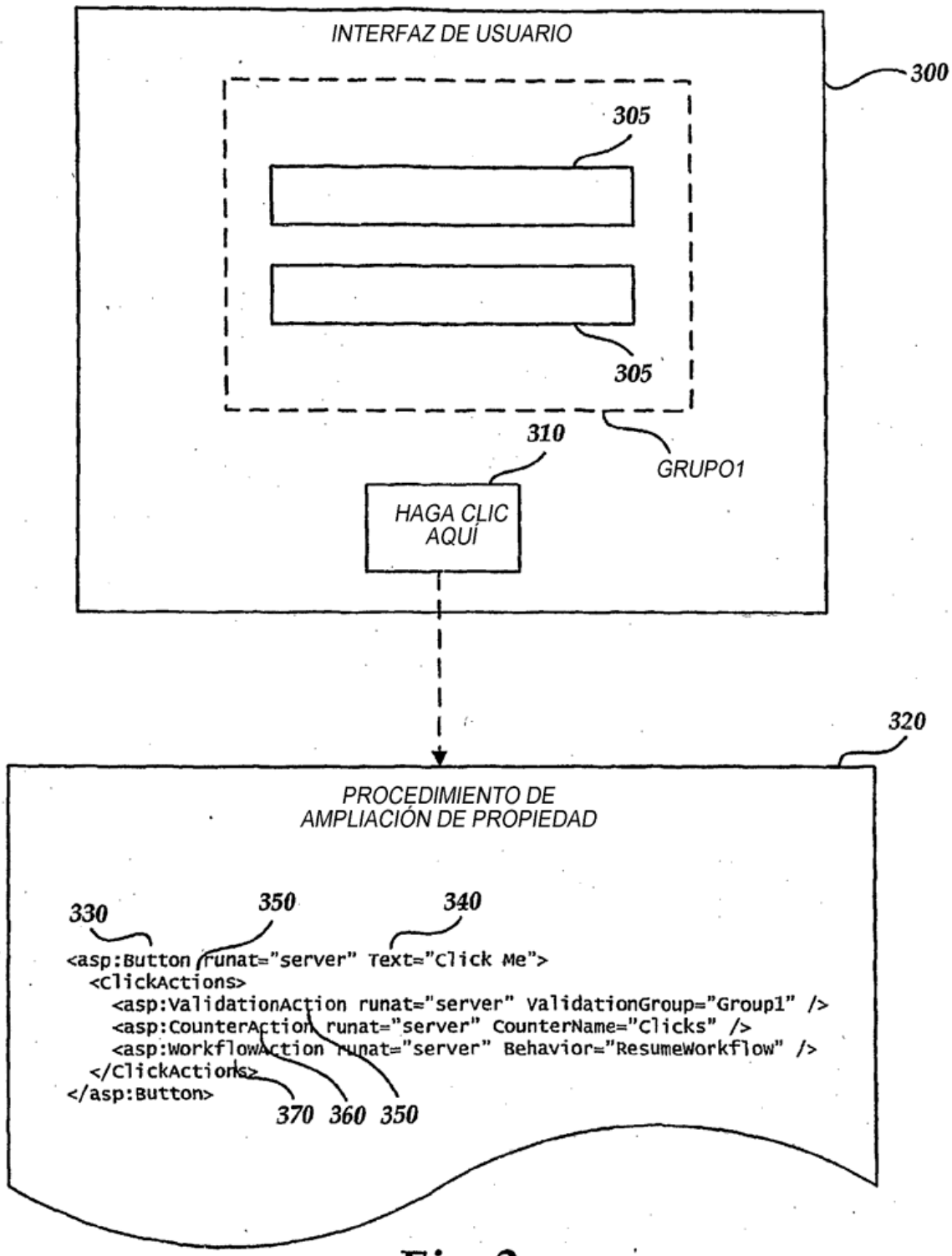
INTERFAZ DE USUARIO
300
305
305
310
GRUPO1
HAGA CLIC AQUÍ
320
PROCEDIMIENTO DE AMPLIACIÓN DE PROPIEDAD
330
350
340
<asp:Button runat="server" Text="Click Me">
<ClickActions>
<asp:ValidationAction runat="server" ValidationGroup="Group1" />
<asp:CounterAction runat="server" CounterName="Clicks" />
<asp:WorkflowAction runat="server" Behavior="ResumeWorkflow" />
</ClickActions>
</asp:Button>
370
360
350

Fig. 3

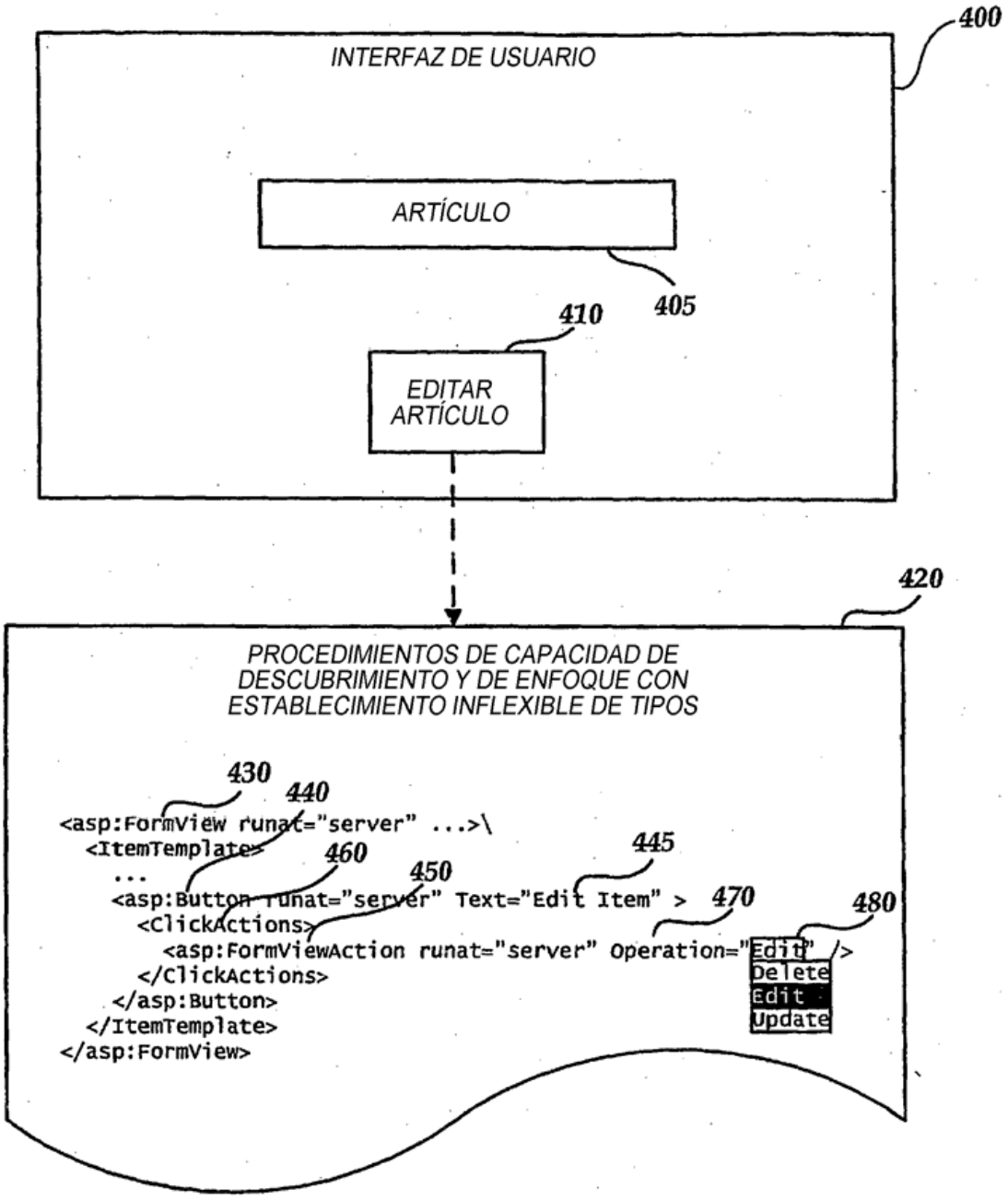
400
INTERFAZ DE USUARIO
ARTÍCULO
405
410
EDITAR ARTÍCULO
420
PROCEDIMIENTOS DE CAPACIDAD DE DESCUBRIMIENTO Y DE ENFOQUE CON ESTABLECIMIENTO INFLEXIBLE DE TIPOS
430
440
445
460
450
470
480
<asp:FormView runat="server" ...>\
<ItemTemplate>
...
<asp:Button runat="server" Text="Edit Item" >
<ClickActions>
<asp:FormViewAction runat="server" Operation="Edit" />
</ClickActions>
</asp:Button>
</ItemTemplate>
</asp:FormView>
Delete
Edit
Update

Fig. 4

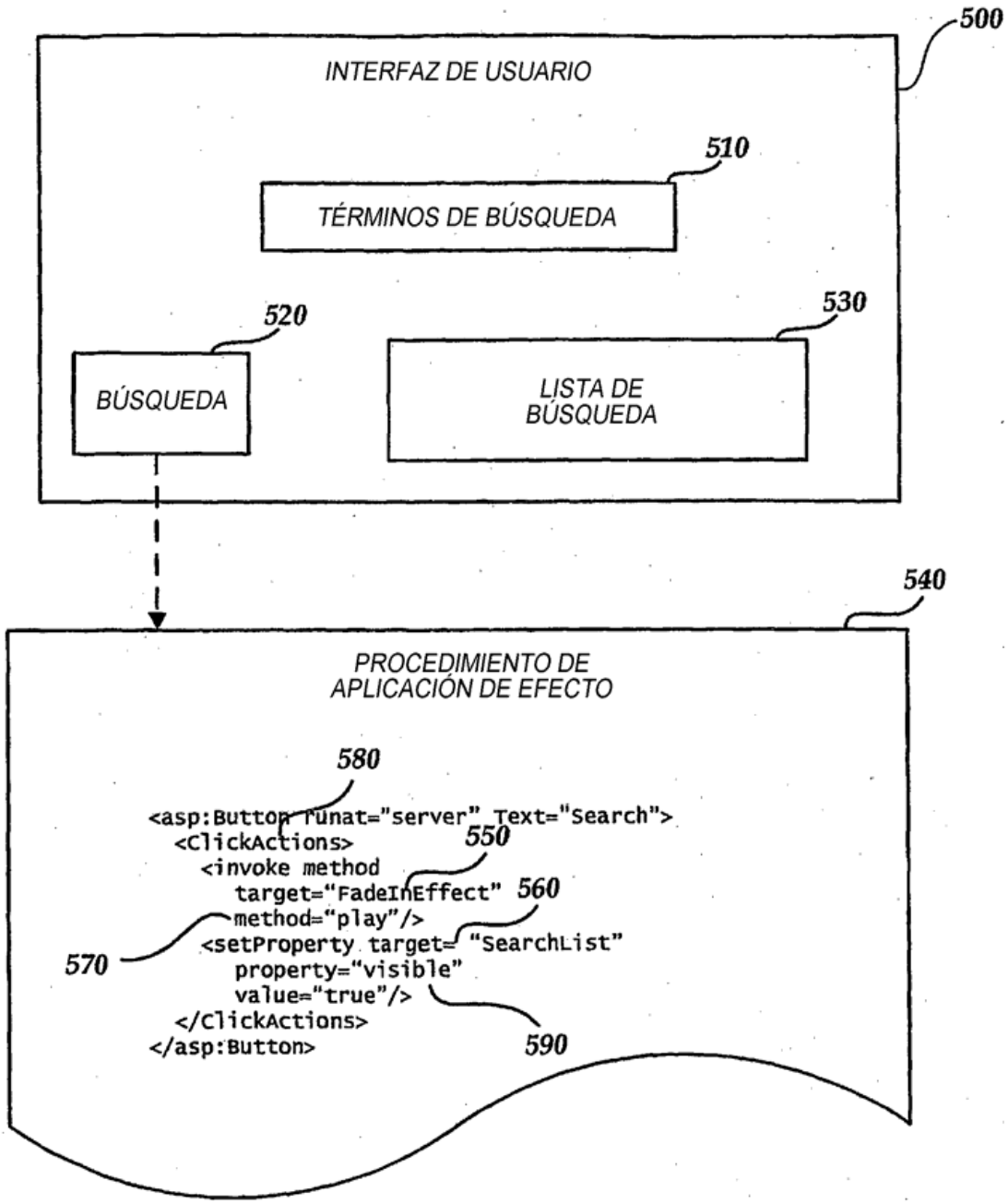
500
INTERFAZ DE USUARIO
510
TÉRMINOS DE BÚSQUEDA
520
BÚSQUEDA
530
LISTA DE BÚSQUEDA
540
PROCEDIMIENTO DE APLICACIÓN DE EFECTO
580
<asp:Button runat="server" Text="Search">
<ClickActions>
550
<invoke method
target="FadeInEffect"
560
method="play"/>
570
<setProperty target="SearchList"
property="visible"
value="true"/>
590
</ClickActions>
</asp:Button>

Fig. 5

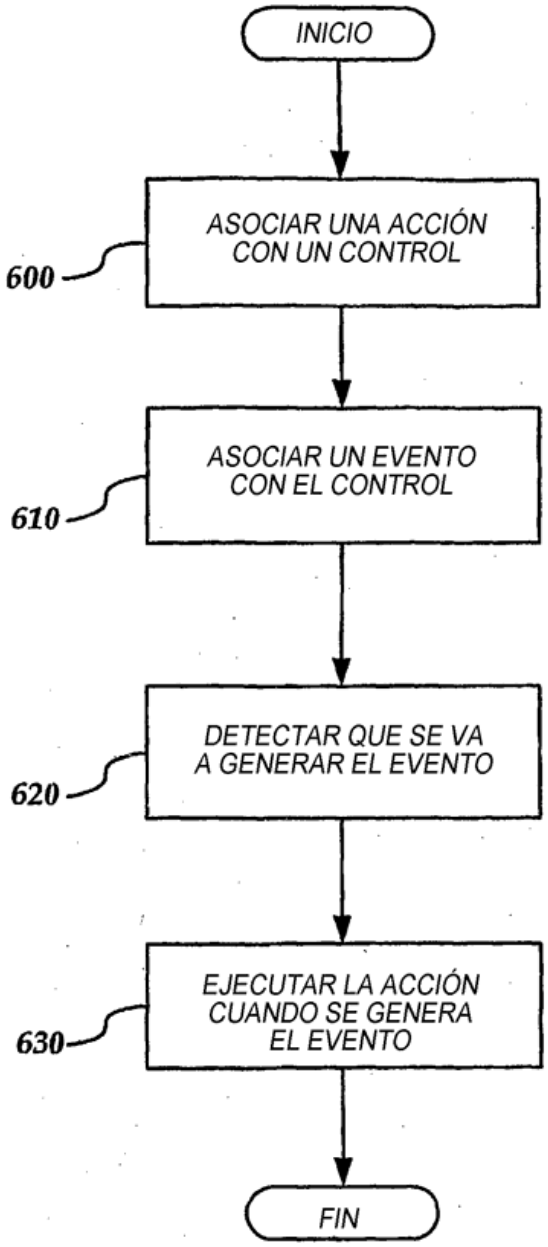
INICIO
ASOCIAR UNA ACCIÓN CON UN CONTROL
600
ASOCIAR UN EVENTO CON EL CONTROL
610
DETECTAR QUE SE VA A GENERAR EL EVENTO
620
EJECUTAR LA ACCIÓN CUANDO SE GENERA EL EVENTO
630
FIN

Fig. 6

700

```
/// <resumen>
/// Clase de base para todas las acciones de control.
/// Cada acción puede especificar si es necesario que esta se ejecute antes del evento propiamente dicho o después. Por ejemplo,
/// es necesario que ValidationAction se ejecute antes de que se genere el evento.
/// El valor de retorno es significativo para acciones que se ejecutan antes de que se genere el evento.
/// La acción podría retornar verdadero para cancelar la posterior generación del evento.
/// </resumen>
public abstract class ControlAction {

    protected ControlAction(ControlActionSequence actionSequence);

    public ControlActionSequence Sequence { get; }

    public abstract bool Execute(Control associatedControl, EventArgs e);
}

public sealed class ControlActionCollection : CollectionBase {

    public ControlActionCollection(Control associatedControl);

    // cosas de la colección

    public ControlAction this[string id] { get; }

    public bool ExecuteActions(ControlActionSequence sequence, EventArgs e);
}

public enum ControlActionSequence {

    BeforeEvent,
    AfterEvent
}


/// <resumen>
/// Permite que una acción participe en la personalización del comportamiento de devolución. Los controles
/// realizarían una llamada a acciones que implementan esta interfaz para las acciones que están asociadas con
eventos
/// relacionados con la devolución.
/// Por ejemplo, ValidationAction implementaría esta interfaz para activar el comportamiento de
/// validación de PostBackOptions.
/// </resumen>
public interface IPostBackCustomizer {

    void CustomizePostBack(PostBackOptions postBackOptions);
}
```

Fig. 7

800

```
public class Button : WebControl {

    public ControlActionCollection ClickActions {
        get {
            // ...
        }
    }

    protected virtual PostBackOptions GetPostBackOptions() {
        PostBackOptions options = new PostBackOptions(...);

        // Cualquier lógica necesaria para configurar las opciones

        foreach (ControlAction action in _clickActions) {
            IPostBackCustomizer postBackCustomizer = action as IPostBackCustomizer;
            if (postBackCustomizer != null) {
                postBackCustomizer.CustomizePostBack(options);
            }
        }

        return options;
    }

    protected virtual void OnClick(EventArgs e) {
        _clickActions. ExecuteActions(this, ControlActionSequence.BeforeEvent, e);

        // Generar evento

        _clickActions.ExecuteActions(this, ControlActionSequence.AfterEvent, e);

        // Generar evento de propagación
    }
}

public class ListControl : WebControl {

    public ControlActionCollection SelectionChangedActions { get; }
}

public class TextBox : WebControl {

    public ControlActionCollection TextChangedActions { get; }
}
```

Fig. 8