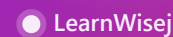


Controles Wisej personalizados, clases de cliente y temas

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Crea controles con aspecto nativo mediante clases de servidor, clases de widget de cliente, claves de apariencia y recursos incrustados. Qué significa esto para un desarrollador de integraciones con Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Crea controles con aspecto nativo mediante clases de servidor, clases de widget de cliente, claves de apariencia y recursos incrustados.

1. El problema de integración

Algunas integraciones deben comportarse como controles Wisej.NET de primera clase: reutilizables, con soporte de temas, diseñables y mantenibles en muchas aplicaciones.

2. Conceptos clave

Concepto	Cómo usarlo
Clase de control	Usa un control de servidor cuando se necesite una integración con aspecto nativo.
AppearanceKey	Conecta el aspecto visual del control con los estilos del tema.

3. Lección de producción

Un control Wisej.NET personalizado es la forma de integración más sólida. Conviene usarlo cuando el widget forma parte de la plataforma del producto, cuando los temas importan o cuando el control debe verse y comportarse como un control nativo de Wisej.NET. El patrón es similar a crear un control integrado: una clase de servidor expone propiedades y eventos .NET, y una clase de cliente recibe el estado renderizado y es dueña del objeto del navegador. La clase de cliente puede extender ``wisej.web.Control`` o ``wisej.web.Widget`` según el comportamiento deseado. Puede crear el objeto de terceros, implementar métodos apply de propiedades, gestionar cambios de tamaño o de apariencia y traducir los callbacks del proveedor en eventos de Wisej.NET. La clase de servidor puede sobrescribir ``OnWebRender`` para enviar solo las propiedades que necesita el cliente. Este enfoque requiere más código, pero ofrece la arquitectura más limpia a largo plazo. Es la opción adecuada para controles que se distribuyen como parte de un framework, una biblioteca de extensiones o un producto empresarial, donde

el wrapper debe aparecer en IntelliSense, funcionar en el diseñador y mantenerse estable entre versiones de la biblioteca subyacente del proveedor.

Un control de servidor para una integración más estrecha

Derivar de `Wisej.Web.Control` en lugar de `Wisej.Web.Widget` cambia la naturaleza de la clase. Un Widget es un host para un script; un Control es un miembro de primera clase del framework, con el mismo comportamiento de diseño, foco, tema y Designer que un botón o un panel integrados. El `SimpleGaugeControl` del laboratorio sigue este camino para que el indicador pueda arrastrarse, dimensionarse, anclarse y tematizarse como cualquier otro control.

La clase de servidor expone propiedades tipadas y eventos .NET, igual que el wrapper, pero ya no lleva un `InitScript` ni una bolsa de `Options`. En su lugar, declara qué clase de cliente la implementa y renderiza su estado mediante `OnWebRender`. Los setters de las propiedades marcan el control como pendiente de actualización, y el framework decide cuándo enviar el cambio.

Elige esta profundidad cuando el widget forme parte de la plataforma del producto, cuando los temas importen o cuando el control deba sentirse nativo para quienes construyen pantallas con él. El código adicional se amortiza con cada pantalla que usa el control sin tener que pensar en él.

Una clase de cliente en Platform

La mitad del navegador de un control personalizado es una clase JavaScript definida con el modelo de clases de cliente de Wisej.NET, basado en qooxdoo, y colocada en la carpeta `/Platform` de la aplicación para que el framework la cargue con el cliente. La clase extiende `wisej.web.Control`, o `wisej.web.Widget` cuando se busca el comportamiento de host de scripts, y declara sus propiedades con métodos `apply`.

Un método `apply` se ejecuta cada vez que el servidor cambia la propiedad correspondiente, y ahí es donde va la llamada al proveedor: el `apply` de `value` llama al método del proveedor que establece el valor, y el `apply` de `maximum` actualiza las opciones del proveedor. El constructor de la clase crea la instancia del proveedor dentro del elemento de contenido del control, y el destructor la destruye. Los cambios de tamaño y de apariencia llegan como eventos que la clase puede gestionar en un único lugar.

Mantén la clase de cliente libre de lógica de aplicación. Traduce entre el framework y el proveedor, y nada más, para que pueda reutilizarla cualquier control de servidor que se renderice en ella.

Renderizar el estado con OnWebRender

`OnWebRender` es la oportunidad del servidor para decidir exactamente qué ve el cliente. La sobrescritura recibe un objeto `config` dinámico, y el control escribe en él solo las propiedades que declara la clase de cliente: `config.value`, `config.minimum`, `config.maximum`, `config.caption`. Los nombres van en camelCase para coincidir con los nombres de las propiedades del cliente, y Wisej.NET envía solo los valores que cambiaron desde el último renderizado.

Esto es estado intencionado y no estado accidental. Una propiedad que existe en el servidor para el Designer, o para la lógica de negocio, simplemente no se escribe en `config` y nunca llega al navegador. Un valor que sería costoso de serializar, como un array de historial grande, puede reducirse a lo que necesita el indicador antes de asignarlo. La autoevaluación del laboratorio pregunta por el papel de `OnWebRender`, y la respuesta es que constituye el contrato entre las dos mitades del control.

Los eventos fluyen en sentido contrario mediante `OnWebEvent`, que trata el Módulo 7. Las dos sobrescrituras juntas definen todo el protocolo de comunicación del control.

Claves de apariencia, temas y tiempo de diseño

Un tema de Wisej.NET aplica estilos a los controles según su clave de apariencia. Dar al control del indicador una `AppearanceKey` como `gauge`, y añadir una entrada con esa clave al tema o a un mixin de tema, conecta los colores, las fuentes y los espaciados del control con el tema de la aplicación. Al cambiar de tema, el indicador se reestiliza junto con todos los demás controles, y esa es la diferencia entre un control que parece nativo y uno que parece pegado.

Los recursos del tema abarcan más que los colores. Las imágenes y los iconos que necesita el control pueden declararse como recursos del tema, de modo que un tema oscuro aporte variantes oscuras sin cambiar código. La clase de cliente lee sus estilos de la apariencia en lugar de codificarlos a mano, y la instancia del proveedor se configura con esos valores en los métodos apply.

El renderizado en tiempo de diseño es la prueba final. El Designer instancia el control y lo renderiza en vivo, así que la clase de cliente debe soportar datos de ejemplo, la ausencia de servidor y creaciones repetidas mientras el desarrollador lo redimensiona. Por eso el laboratorio pide una captura de pantalla en tiempo de diseño: si el control se renderiza bien en el Designer, se adoptará; si lanza errores, se evitará.



Regla de producción

Mantén claras la propiedad del estado en Wisej.NET y el comportamiento del widget de cliente. Documenta el contrato cliente/servidor antes de confiar en la integración en producción.



Error habitual que debes evitar

Copiar JavaScript de demostración en las pantallas sin documentar el ciclo de vida, los recursos, los contratos de eventos, la limpieza ni los límites de seguridad.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Crear un control de servidor derivado de `Wisej.Web.Control` cuando se necesite una integración más estrecha.
- ✓ Crear una clase de cliente en ``/Platform`` usando el modelo de clases de Wisej.NET/qx.
- ✓ Renderizar el estado mediante ``OnWebRender`` y asignarlo a propiedades u opciones del cliente.
- ✓ Entender `AppearanceKey`, los recursos del tema y las consideraciones del renderizado en tiempo de diseño.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el flujo de trabajo recomendado, los entregables requeridos y la autoevaluación antes de entregar.



Laboratorio práctico

Crea un control personalizado `SimpleGaugeControl` con una clase de servidor y una clase de cliente en `Platform`. Añade una clave de apariencia básica y haz que se renderice en modo de diseño.

Flujo de trabajo recomendado

- ✓ Abre el proyecto existente del curso de integración de Wisej.NET y ejecútalo en local.
- ✓ Identifica el componente de servidor, el widget de cliente, el archivo de recursos o el endpoint que intervienen en este módulo.
- ✓ Haz un solo cambio pequeño cada vez y pruébalo en el navegador antes de continuar.
- ✓ Mantén el JavaScript específico del proveedor dentro del adaptador o wrapper en lugar de repartirlo por las páginas.
- ✓ Actualiza la interfaz para que el estudiante vea con claridad el éxito, los errores, el progreso o los cambios de comunicación.
- ✓ Toma capturas de pantalla o notas que demuestren que la integración funciona y expliquen la decisión de diseño.

Entregables requeridos

- ✓ Clase de control de servidor
- ✓ Esqueleto de la clase qx de cliente
- ✓ Entrada de tema/apariencia
- ✓ Captura de pantalla o notas en tiempo de diseño

Autoevaluación antes de entregar

- ✓ ¿Cuándo compensa el trabajo adicional de un control personalizado?
- ✓ ¿Qué papel cumple `OnWebRender`?
- ✓ ¿Por qué importa la integración con los temas para la adopción en la empresa?