# JSP

Ahora echemos un vistazo a las formas de producir aplicaciones basadas en páginas de servidor Java. Bueno, necesitamos entender cómo funcionan las páginas del servidor Java. ¿Cuál es la sintaxis? Y en realidad, algunas opciones allí. El uso del lenguaje de expresión, el uso de beans CDI, bibliotecas de etiquetas y formas de manejar errores con JSP.

En primer lugar, debo decirle que realmente una página de servidor Java es en realidad un servlet. Cuando implementa la aplicación JSP, cada página se convierte primero en un servlet. Entonces, en términos de capacidades de las páginas del servidor Java y lo que pueden hacer, cómo se ejecutan, son absolutamente idénticas a los servlets. Entonces, en términos de las opciones de tiempo de ejecución, es lo mismo. Tengo el mismo ciclo de vida, los mismos eventos. Son exactamente los mismos comportamientos.

Sin embargo, obviamente es un estilo diferente de escribir la página. Un servlet es solo una clase de Java. Codificas Java. Una página es una página. Así que visualmente podrías editar eso. Está trabajando con marcas HTML en lugar de simplemente escribir código Java. Por supuesto, la página puede contener fragmentos de código Java, como este, por ejemplo. Eso es claramente un trozo de código Java. Está recortado en la estructura de la página y, en general, se ve mucho más amigable para el diseñador web que el servlet.

Así que supongo que es un intento de hacer que el proceso de diseño de una página se adapte más a los gustos del desarrollador de la interfaz de usuario en lugar de solo a un codificador. Ahora, todo el código y las técnicas del servlet siguen siendo aplicables. Así que los beans CDI y los filtros web y las operaciones del ciclo de vida del contenedor son lo mismo. Realmente se trata de un estilo diferente de codificación.

Esta es la sintaxis de la página del servidor Java. Entonces, ¿qué puedes poner en una página? Puedes poner en comentarios. Ahí tienes Puedes poner bits de código que llamamos directivas. Ahora, hay varios tipos de directivas allí, page, include y taglib. De hecho, vamos a echar un vistazo a otras directivas un poco más tarde.

Pero página, por ejemplo, que describe cosas globales sobre la página. Entonces es, por ejemplo, tipo de contenido, codificación. Bien adivina que. Esa instrucción (página, tipo de contenido y codificación) se traducirá a un código futuro de un servlet que se basará en esa página, se creará a partir de esa página y ese servlet enviará los encabezados HTTP relevantes al cliente. Así que configurando el tipo de contenido.

O, por ejemplo, ese bit donde pondrías una importación. Solo estamos importando otra clase de Java aquí. Y nuevamente, eso se traduce al código de un servlet. Eso solo tendrá esa pieza de lógica Java allí.

La página general podría contener cualquier HTML, de verdad. Como ves, es un documento HTML. Pero además de contener elementos HTML de texto sin formato, también puede contener declaraciones, scriptlets y expresiones. Una declaración, la que tiene este signo de exclamación, si coloca la declaración en la página, se creará un fragmento de código que se incluirá en el futuro servlet fuera de un servicio de método. Entonces, si desea agregar una variable de instancia o crear otro método, coloque un signo de exclamación. Ese código irá fuera del futuro servicio de método de servlet.

Si pones un scriptlet, ese es el que tiene los signos de porcentaje. Estos son los guiones. Esto es solo un poco de cualquier código Java que va dentro del servicio de método. Entonces, lo que quieras codificar, como manejar parámetros de solicitud y cualquier lógica.

Una expresión es básicamente exactamente lo mismo que si imprimiera LAN. Es exactamente la misma idea. Así que estás imprimiendo algo al cliente. Podría haber escrito un scriptlet, supongo, que genera ese valor. Pero podría ser sólo una expresión. Así que es exactamente lo mismo que una impresión.

También tenga en cuenta que si está escribiendo un servlet y está escribiendo su método de servicio, tiene el control total sobre cómo le gustaría que se llamen sus parámetros. Objetos de solicitud de servlet HTTP y de respuesta de servlet HTTP. Puedes, cuando codificas un servlet, llamarlo como quieras. No tiene que llamar a los parámetros de solicitud y respuesta.

Pero en una página del servidor Java, se forma un servlet para ti. Entonces, los nombres de los objetos de servlet estándar están precodificados. Y aquí dice "solicitud". Este será el nombre de una solicitud de servlet HTTP en un servlet generado en el futuro. Así se llamará. Así que realmente no puedes cambiar eso, porque se generará automáticamente para ti.

Así que los objetos estándar, ver aquí. Es una lista de estos objetos. Solicitud, respuesta, se llama al escritor de impresión. Sesión. La sesión HTTP se llama sesión. servlet. El objeto de contexto se denomina contexto de página.

Por ejemplo, la página de palabras clave es un sinónimo de la palabra clave de Java this. Podrías haber escrito simplemente "este" punto algo, supongo. Pero quienquiera que haya diseñado las páginas del servidor Java pensó que esa página sería más intuitiva para un diseñador web que pudiera desear escribir algún código de scriptlet y consultar la página actual. Así que es básicamente una página de sinónimos y esto. Estos son alias para objetos de servlet estándar.

Ahora, aquí hay uno interesante. Es una sintaxis de páginas de servidor Java alternativa, sintaxis XML. Este ejemplo está haciendo exactamente lo mismo que el ejemplo en una página anterior. No hay diferencia en absoluto. Eso es una repetición completa.

Pero en lugar de escribir directivas y declaraciones y scriptlets con estos signos de exclamación o signo igual o signos de porcentaje, está utilizando marcas XML. Rodean todo el documento con un elemento llamado JSP:root. Eso encierra todo el documento. Y luego pones todos los otros componentes como directivas.

Así que recuerde en la página anterior, la directiva tenía-- qué era-- estaba con un signo de arroba. Y luego tuvimos la declaración, y eso fue con un signo de exclamación. Y luego teníamos una expresión, por ejemplo, y eso era con un signo igual. Bueno, allá vas.

Entonces, en lugar de todos estos signos y signos de exclamación y signos de igual, solo lo está deletreando como un marcado XML real. Y luego, en un scriptlet, por ejemplo, o en una declaración o dentro de un elemento en particular, simplemente coloca su código Java como de costumbre.

Ahí tienes Entonces puede construir la página como un documento XML. Eso sería un poco complicado cuando se trata de algún contenido HTML en una página porque en el contenido HTML, esa parte que produce HTML también tendrá que cumplir con el estándar XML. Por lo tanto, es posible que desee no crear un elemento HTML, sino crear un elemento XML que describa un elemento HTML. Puede ser un poco largo, así que ahí lo tienes. Pero en cuanto a la funcionalidad, ambas sintaxis están logrando lo mismo. Es solo un estilo diferente de escribir la página, cualquiera que sea el estilo que prefiera.

Dentro de una página, puede escribir expresiones para acceder al código Java, que podría estar accediendo a beans CDI, objetos fuera de la página. Hay dos estilos de expresiones que podría usar con un signo de dólar, que se conoce como inmediato, y con el símbolo de almohadilla, libra, como lo llame, que es el diferido. Las expresiones diferidas son utilizadas principalmente por las caras del servidor Java y se tratan más adelante. Depende de la forma en que funciona el ciclo de vida de JSF.

Las caras del servidor Java tienen sus propias consideraciones de ciclo de vida, por lo que utilizan una notación de expresión ligeramente diferente. Y las páginas del servidor Java utilizan expresiones inmediatas, que es lo que estamos haciendo en este capítulo. Entonces, con un signo de dólar para este capítulo.

Las expresiones le permiten acceder a los beans CDI, sus propiedades. Hay estilos ligeramente diferentes. Puede usar la notación de puntos para acceder a una propiedad, o puede usar corchetes y el nombre de la notación de propiedad. Bueno, funciona de cualquier manera. No es realmente diferente.

Podría pasar parámetros e invocar operaciones. Podría manejar arreglos, usando índices de arreglos. O incluso puede escribir expresiones lambda directamente en su página con lenguaje de expresión. Ahí vas. Hay más en el lenguaje de expresión. Es esta página la que nos muestra cuáles son los principales operadores del lenguaje de expresión.

Entonces, por ejemplo, punto para acceder a una propiedad, coma como separador de declaraciones, corchetes para acceder, digamos, a un elemento de matriz. Parámetros de método con corchetes, operadores aritméticos, asignación, operadores relacionales, operadores lógicos y token lambda. Así que bastantes operadores diferentes. Eso significa que puede poner estas expresiones en su página y, a través de ellas, acceder a la lógica en sus beans CDI y sus objetos fuera de esa página.

Bueno, ¿qué son estos objetos? Puede acceder a cosas que están en diferentes ámbitos de memoria. Estos son los ámbitos de solicitud, página, sesión y aplicación. Acceda a eso desde su página. Obviamente, puede acceder a todos estos objetos JSP implícitos. Recuerde, los hemos mencionado antes.

Estamos diciendo que no le da un nombre a su objeto de solicitud. Se da por ti. Y es pedido. Ahí vas. Eso es como JSPWriter, la solicitud es HTTPServletRequest, la respuesta es HTTPServerResponse, etcétera. Estos son los objetos de tiempo de ejecución de servlet estándar.

También tiene implícito un interesante conjunto de objetos aquí que se ocupan del manejo del contenido HTTP. Entonces, parámetros, valores de parámetros, encabezado, valores de encabezado, cookies. Probablemente un objeto sobre el que me gustaría comentar rápidamente es initParam. Puede poner parámetros de inicialización para servlets en un archivo XML web. Por lo tanto, no es un usuario de parámetros el que pasa una URL. Eso es param. initParam es el bit que coloca en un archivo XML web para pasar parámetros de inicialización a un servlet.

Y finalmente, en esta esquina inferior derecha, podemos ver los ejemplos de expresiones, acceder a parámetros, acceder a beans CDI, acceder a algunos mensajes de excepción. Entonces podría poner ese código directamente en la página JSP para manipular con estos objetos de memoria.

Hablando de frijoles CDI. Aquí hay un ejemplo. Creaste un frijol. Bueno, es una solicitud enfocada en este caso particular. Solo para su comodidad, puede utilizar una anotación con nombre. Si no coloca la anotación con nombre, aún puede acceder al bean, pero debe usar un nombre de clase, lo que podría no ser tan conveniente. Entonces, el alias con nombre podría ser una forma más conveniente de acceder al bean. Y ahí tienes

Solo puede acceder a las propiedades del bean o llamar a los métodos del bean directamente desde la página. Muy conveniente. Y supongo, nuevamente, que esta es la forma de abordar la segregación adecuada del controlador de vista del modelo, porque luego tienes tu página que se enfoca en la producción de la interfaz de usuario o el diseño, y luego tus beans que llevan a cabo la lógica. Y esta forma bastante sencilla de vincular una página y frijoles a través de expresiones.

Ahora, las páginas del servidor Java vienen con un conjunto de bibliotecas estándar. Son conocidos como JSTL. Y esa es la biblioteca de etiquetas estándar de Java Server Pages. Las bibliotecas que vienen con el tiempo de ejecución de las páginas del servidor Java nos proporcionan elementos que nos permiten controlar la lógica de la página. Cada biblioteca de etiquetas debe declararse en una página como una referencia de taglib. La biblioteca de etiquetas se asociará con un prefijo. Entonces, cualquiera que sea el prefijo, depende de usted. Y luego está accediendo a elementos dentro de una biblioteca.

Bastantes bibliotecas diferentes, y muchos elementos en ellas. Así que intentaré darle algún tipo de ejemplo de lo que espera que hagan las bibliotecas. Y en este ejemplo particular, tenemos para cada elemento que nos permite crear un iterador y recorrer una colección de productos.

Suponga que tiene un bean CDI llamado orden. Supongamos que el orden de frijol CDI tiene una colección de productos, algún tipo de lista de productos. Desea recorrer esa lista de productos, y luego desea sacar cada producto de esa lista, asignarlo a la variable llamada producto, var producto. Después de eso, desea obtener el nombre del producto y el descuento del producto.

Pero ya ves, lo interesante es que con descuento quieres hacer alguna prueba primero. Así que hay esta condición si. Prueba que cualquier descuento es mayor que cero, y luego muestra la palabra venta si ese es el caso. Y en cuanto al nombre del producto y el precio, bueno, simplemente los muestra tal como están. Entonces c out le permite simplemente generar el valor.

Este es solo un ejemplo. Bastantes otros elementos en estas bibliotecas. Algunas otras bibliotecas. Se ocupan del formato de los objetos. Se ocupan de la lógica del manejo de página de estos objetos. Por lo tanto, nos permite codificar en una página del servidor Java, pero sin realmente escribir código de scriptlet directo. Entonces, está expresando la lógica de lo que se supone que debe hacer la página y, sin embargo, no está codificando directamente en Java.

Cuanto más coloque cosas de scriptlet, código de scriptlet, en la página JSP, más hará que se vea como un servlet, lo que eventualmente anulará el punto de crear una página JSP en primer lugar. Entonces, ¿por qué no estás escribiendo un servlet? Así que estás tratando de mantener la página lo más declarativa posible donde sea más adecuada, está más orientada a la forma visual de editar eso.

Por lo tanto, está tratando de guardar todo el código, todo el manejo de la lógica empresarial compleja, en los beans CDI y acceder a ellos a través de estos elementos de las bibliotecas de etiquetas y mediante expresiones que lo ayudan a construir esa lógica de una manera mucho más declarativa.

Bueno, también debemos mencionar que debemos poder manejar errores para las páginas del servidor Java. Ahora, este bit en la parte inferior de la página, ¿reconoce estas líneas? Solicitar URI de un error. ¿Qué es una excepción? ¿Cuál es su tipo?

Estos son atributos del objeto de solicitud. Exactamente igual que vimos en el capítulo anterior sobre servlets, cuando creamos un controlador de errores de servlet. Bueno, verá, la página del servidor Java tiene acceso exactamente a los mismos atributos de requestScope, tal como lo hizo el servlet. Entonces significa que puede crear una página de error y acceder a toda esa información.

La diferencia, supongo, es esta. Además de la capacidad del servidor para registrar páginas de error en un descriptor de implementación WebXML, las páginas del servidor Java también pueden designar una página de error específica para una página determinada. Así que estás diciendo que hay una página llamada Some. Si algo sale mal en una página Some, desea llamar a Some manejador de errores JSP. Así que esa es esa otra página.

Y la otra página, tenga en cuenta, está marcada aquí con "es verdadero el controlador de errores". Oh, lo es. Entonces, una página dice que la otra es un controlador de errores para ella, y la otra página dice, sí, soy el controlador de errores. Y supongo que ahí es cuando comienzas a acceder a la información sobre el error y se la presentas a tu usuario de la manera agradable que quieras.

Bueno, supongo que eso es todo. Verá, el resto de las cosas sobre las páginas del servidor Java son exactamente las mismas que sobre los servlets. Entonces, por ejemplo, eventos y filtros de ciclo de vida, filtros web, todo es lo mismo. Solicitar despachadores, es todo lo mismo. Así que es solo un estilo realmente diferente de crear ese diseño que estamos tratando de hacer lo más declarativo posible sin tener que codificar directamente dentro de una página en Java como tal.

En términos del ejercicio, lo que va a hacer es cambiar su aplicación de servlet web actual que le muestra la lista de productos. Y va a mostrar la lista de productos en una página del servidor Java, no en un servlet. Sin embargo, seguirá utilizando el servlet por razones de controlador.

Entonces, la idea es que una búsqueda, qué productos está buscando, esa búsqueda aún será ejecutada por un servlet. Pero luego la visualización de los resultados será ejecutada por la página del servidor Java. El servlet puede reenviarle solicitudes. Así que vamos a utilizar despachadores de solicitudes aquí.

El otro ejemplo que está haciendo en este capítulo es que está creando una página ProductEdit para actualizar productos, mostrar el producto y darle a un usuario la capacidad de actualizarlo. Solo por el placer de probar una técnica diferente. la página ProductEdit JSP utilizará un filtro web como controlador.

Así que tienes dos páginas aquí. Tiene una página que muestra la lista de productos y otra página que muestra las capacidades de edición del producto. Y la lista uno está controlada por un servlet, y la edición está controlada por el filtro.

En una aplicación del mundo real, probablemente elegirá una forma u otra para implementar la disposición del controlador de vista del modelo. No usaría esta combinación de tecnologías en la misma aplicación. Probablemente sería una exageración. Por lo tanto, probablemente opte por un servlet como controlador o un filtro como controlador.

Bueno, en el ejercicio se le dice que haga ambas cosas solo para probar y poder comparar cómo una técnica difiere de otra. Por lo tanto, podrá tomar decisiones en sus propias situaciones de desarrollo en cuanto a qué técnica será más apropiada para usted.