Crear una aplicación J2C para una transacción IMS que contiene matrices

Contenido

Crear una aplicación J2C para una transacción IMS que contiene matrices

En esta guía de aprendizaje se describe cómo utilizar el asistente de bean Java J2C para crear una aplicación web sencilla que procese una transacción IMS con datos de entrada y de salida que contengan matrices.

Objetivos educativos

Esta guía de aprendizaje le permite:

- Utilizar el asistente Bean J2C para crear un bean Java J2C que ejecute una transacción IMS.
- Crear un método Java para que el bean ejecute la transacción IMS.
- Crear una clase de Java de proxy para probar la aplicación

Tiempo necesario

30 minutos

Información relacionada

Ver la versión PDF

Ejemplo de matrices de entrada y salida

Introducción a la creación de una aplicación J2C para una transacción IMS que contiene matrices

En esta guía de aprendizaje se describe cómo utilizar el asistente de bean Java J2C para crear una aplicación web sencilla que procese una transacción IMS con datos de entrada y de salida que contengan matrices.

Esta guía de aprendizaje puede necesitar algunos componentes instalables opcionales. Consulte la lista de requisitos del sistema para asegurarse de haber instalado los componentes opcionales adecuados.

Esta guía de aprendizaje se divide en varios ejercicios que deben realizarse siguiendo el orden indicado para cumplir los objetivos marcados. En esta guía de aprendizaje aprenderá a utilizar el asistente Bean Java J2C para crear un bean Java que ejecute una transacción en IMS. Mientras realiza los ejercicios podrá:

- Utilizar el asistente Bean J2C para crear un bean Java J2C que ejecute una transacción IMS.
- Crear un método Java para el bean, `runInOut.java`, para ejecutar la transacción IMS.
- Crear una clase Java proxy, `TestInOutProxy.class`, para construir el mensaje de entrada de la transacción IMS, invoque el método de bean Java J2C que ejecuta la transacción IMS y visualice los datos de salida devueltos por la transacción IMS.

  Nota: La clase Java `TestInOutProxy.java` se creó para un entorno local inglés; es posible que deba hacer modificaciones en el código para otros entornos locales.

Tiempo necesario

La consecución de esta guía de aprendizaje debe tardar aproximadamente 30 minutos. Si explora otros conceptos relacionados con esta guía de aprendizaje, puede tardar más en completarla.

Nivel de habilidad

Experimentado

Público

Esta guía de aprendizaje está pensada para los usuarios familiarizados con sistemas de información de empresa (EIS) y con IMS en concreto.

Requisitos del sistema

Para completar esta guía de aprendizaje, necesita tener instalados los componentes y las herramientas siguientes:

- IBM WebSphere Application Server, versión 6.1
- Herramientas J2EE Connector (J2C)
- **Información acerca del entorno IMS**: en esta guía de aprendizaje, la aplicación interactúa con un programa de aplicación en IMS. Debe obtener información como por ejemplo el nombre de sistema principal y el número de puerto de IMS Connect y el nombre del almacén de datos de IMS en el que se ejecutará la transacción. Póngase en contacto con el administrador del sistema IMS para obtener esta información. Además, debe realizar algún trabajo de configuración en IMS si desea ejecutar este ejemplo.
- **Una copia del archivo COBOL `InEqualsOut.cbl`**: encontrará este archivo en el directorio de instalación del producto: <dir_instal>\IBM\SDP70Shared\plugins\com.ibm.j2c.cheatsheet.content_7.0.0\samples\IMS\InOutArray. Si desea almacenarlo localmente, puede copiar el código de aquí: ../resources/inequalsout.dita
- **Un espacio de trabajo limpio**.

Para utilizar esta guía de aprendizaje, debe tener un servidor de aplicaciones instalado y configurado. Para verificar que tiene disponible un entorno de tiempo de ejecución de servidor, pulse **Ventana → Preferencias**, expanda **Servidor** y pulse **Entornos de tiempo de ejecución instalados**. Puede utilizar este panel para añadir, eliminar o editar definiciones de tiempo de ejecución de servidor instaladas. También puede descargar e instalar soporte para un servidor nuevo.

Prerrequisitos

Para poder realizar esta guía de aprendizaje de principio a fin, debe estar familiarizado con:

- Programación J2EE y Java
- Conceptos básicos de IMS Transaction Manager (IMS TM)

Lección 1.1: Seleccionar el adaptador de recurso

En esta lección seguirá los pasos detallados para seleccionar el adaptador de recursos y configurarlo para la conexión con el servidor IMS.

En esta guía de aprendizaje se utilizan estructuras de datos COBOL para describir los mensajes de entrada y salida de transacción IMS. Los mensajes de entrada y salida son idénticos y contienen una matriz de elementos de cliente, seguidos de un campo que contiene un código de función. La matriz puede tener un máximo de ocho elementos, pero en esta guía de aprendizaje solamente se entrarán tres elementos en el programa de aplicación IMS, el cual solamente devolverá cuatro elementos.

La transacción IMS que se utiliza en esta guía de aprendizaje no es uno de los programas de verificación de instalación de IMS. Utiliza DFSDDLT0, un programa de aplicación IMS que emite llamadas a IMS basándose en información de sentencias de control. A continuación encontrará las sentencias de control

DFSDDLT0 de esta guía de aprendizaje. No obstante, debe configurar su entorno para DFSDDLT0 y proporcionar el JCL necesario, si desea ejecutar la guía de aprendizaje. En esta guía se utiliza SKS2 como código de transacción para la aplicación DFSDDLT0.

Sentencias de control DFSDDLT0

```
S11 1 1 1 1    TP     1
L        GU
E      OK
E  Z0088 DATA  SKS2 03CN001Cathy Tang          CN002Haley Fung         X
                CN003Steve Kuo           123456
WTO IC4JINOU: Single segment received from JITOC
L        GN
E      QD
WTO IC4JINOU: End of input segments from JITOC
L        ISRT  JITOC53
L  Z0113 DATA  TRNCD04CN001Cathy T.            CN002Haley F.           X
                CN003Steve K.            CN004Kevin F.           65432X
               1
E      OK
WTO IC4JINOU: Single segment inserted - 3 elements !!!!!!!!!!!!!
L        GU
```

Conexión con el servidor IMS

1. Si el icono J2EE,  , no aparece en la pestaña superior derecha del espacio de trabajo, deberá pasar a la perspectiva J2EE. En la barra de menús, seleccione **Ventana > Abrir perspectiva > Otras**. Se abre la ventana Seleccionar perspectiva.
2. Seleccione **J2EE**.
3. Pulse **Aceptar**. Se abre la perspectiva J2EE.
4. En la perspectiva J2EE, seleccione **Archivo > Nuevo > Otro**.
5. En la página Nuevo, seleccione **J2C > Bean Java J2C**. Pulse **Siguiente**
6. En la página Selección de adaptadores de recurso, seleccione el adaptador de recursos IMS J2C 1.5. Para esta guía de aprendizaje, seleccione **IMS Connector para Java (IBM: 9.1.0.2.3)**. Pulse **Siguiente**.
7. En la página Propiedades de conexión, deseleccione **Conexión gestionada** y seleccione **Conexión no gestionada**. (En esta guía de aprendizaje utilizará una conexión no gestionada para acceder directamente a IMS.) Acepte el nombre de clase de conexión por omisión de `com.ibm.connector2.ims.ico.IMSManagedConnectionFactory`. En los campos en blanco, entre toda la información de conexión necesaria. Los campos obligatorios, indicados con un asterisco (*), son los siguientes:
   a. **Para conexión TCP/IP**:
      1) **Nombre de sistema principal**: (Obligatorio) Dirección IP o nombre de sistema principal de IMS Connect.
      2) **Número de puerto**: (Obligatorio) Número de puerto que utiliza la conexión IMS destino.
   b. **Para la conexión de opción local**:
      1) **Nombre de IMS Connect**: (Obligatorio) Nombre de la conexión IMS destino.
   c. **Para ambos**:
      1) **Nombre de almacén de datos**: (Obligatorio) Nombre del almacén de datos IMS destino.
8. Puede obtener la información de conexión solicitándola a su administrador de sistema IMS. Cuando haya proporcionado la información de conexión necesaria, pulse **Siguiente**.

Lección 1.2: Configurar un proyecto Web y la interfaz e implementaciones Java

En la lección 1.2 seguirá los pasos necesarios para crear un proyecto Web y una interfaz y unas implementaciones Java

Antes de empezar, debe completar la Lección 1.1. En esta lección realizará las siguientes tareas:

- Crear un bean Java J2C
- Crear un proyecto Web dinámico

1. Todo el trabajo que se realiza en el entorno de trabajo debe asociarse a un proyecto. Los proyectos proporcionan una vista organizada de los archivos y directorios de trabajo, optimizada con funciones basadas en el tipo de proyecto. En el entorno de trabajo, todos los archivos deben residir en un proyecto, por lo que antes de crear el bean Java J2C, deberá crear un proyecto donde colocarlo. En la página Nuevo bean Java J2C, escriba el valor `InOutArray` en el campo **Nombre de proyecto**.
2. Pulse el botón **Nuevo** situado junto al campo **Nombre de proyecto** para crear el nuevo proyecto
3. Ahora creará un proyecto Web dinámico Web. En la página Creación de un nuevo proyecto fuente, seleccione **Proyecto Web** y pulse **Siguiente**.
4. En la página Proyecto Web dinámico, asegúrese de seleccionar los valores siguientes:
   a. **Nombre de proyecto**: InOutArray
   b. **Contenido del proyecto**: acepte el valor predeterminado
   c. **Tiempo de ejecución destino**: WebSphere Application Server v6.1
   d. **Configuraciones**: acepte el valor predeterminado
   e. **Añadir proyecto a un EAR**: marcado
   f. **Nombre de proyecto EAR**: InOutArrayEAR
5. Pulse **Siguiente**.
6. En la página Facetas de proyecto, acepte los valores predeterminados.
7. Pulse **Siguiente**.
8. En la página Módulos Web, acepte los valores predeterminados.
9. Pulse **Terminar**.
10. Aparecerá un recuadro de diálogo solicitando si desea cambiar a la perspectiva Web dinámica. Pulse **Sí**.
11. En la página Propiedades de salida del bean Java J2C:
    a. En el campo **Nombre de paquete**, pulse **Examinar** y seleccione el proyecto `InOutArray`. Pulse **Aceptar**.
    b. Escriba `sample.ims` en el campo **Nombre de paquete**.
    c. Escriba `InOut` en el campo **Nombre de interfaz**.
    d. Escriba `InOutImpl` en el campo **Nombre de implementación**.
    e. Deje sin marcar **Guardar sesión como script ant**.
12. Pulse **Terminar**.

Lección 1.3: Crear un método Java

En la lección 1.3 se describe la creación de un método Java.

Antes de empezar, debe completar la Lección 1.2. En esta lección realizará las siguientes tareas:

- Crear un método Java
- Crear la correlación de datos de entrada y salida entre COBOL y Java

1. **Primero creará un método Java:** en la vista Explorador de proyectos, expanda el proyecto **InOutArray** de Proyectos Web dinámicos.
2. Pulse con el botón derecho del ratón sobre **InOutImpl.java** en JavaSource, y seleccione **Fuente > Añadir método a bean Java J2C**.
3. En la página Método Java, pulse **Añadir**.
4. En el campo **Nombre de método Java** especifique `runInOut` para el nombre del método.
5. Pulse **Siguiente**.
6. **A continuación, creará la correlación de datos de entrada entre COBOL y Java:** junto al campo **tipo de entrada** de la página del método Java, pulse **Nuevo**.
7. En la página Importación de datos, asegúrese de que el campo **Elegir correlación** sea **COBOL a Java**. Pulse **Examinar** junto al campo **Archivo COBOL**.
8. Desplácese hasta la ubicación del archivo InEqualsOut.cbl. (encontrará una copia en la carpeta de instalación del producto: <installdir>\IBM\SDP70Shared\plugins\ com.ibm.j2c.cheatsheet.content_6.0.0\Samples\IMS\InOutArray).
9. Pulse **Abrir**.
10. Pulse **Siguiente**.
11. En la página Importador COBOL, pulse **Mostrar valores avanzados**.
    a. Seleccione las siguientes opciones:

Tabla 1. Valores de parámetros de importador COBOL

Parámetro	Valor
Nombre de plataforma	Z/OS
Página de códigos	IBM-037
Nombre de formato de coma flotante	IBM Hexadecimal
Signo decimal externo	EBCDIC
Nombre Endian	Big
Nombre endian de entero remoto	Big
Nombre de comillas	DOUBLE
Nombre Trunc	STD
Nombre Nsymbol	DBCS

    b. Pulse **Consulta** para cargar los datos.
    c. Se muestra una lista de estructuras de datos. Seleccione **IN-OUT-MSG** en el campo **Estructuras de datos**.
    d. Pulse **Siguiente**.
12. En la página Guardar propiedades,
    a. Seleccione **Valor predeterminado** para **Estilo de generación**.
    b. Pulse **Examinar** junto al **Nombre de proyecto** y elija el proyecto Web **InOutArray**.
    c. En el campo **Nombre de paquete**, escriba `sample.ims.data`.
    d. En el campo **Nombre de clase**, acepte el valor predeterminado **INOUTMSG**. Pulse **Terminar**.
    e. Deje sin marcar **Guardar sesión como script Ant**.
13. En la página Método Java, seleccione **Utilizar tipo de entrada para la salida**.
14. Pulse **Terminar**.
15. Pulse **Terminar** para completar la definición del método.

Lección 1.4: Crear una clase proxy Java para probar la aplicación

En la lección 1.4 se describe la creación de una clase de prueba Java para probar la aplicación.

Antes de empezar, debe completar la Lección 1.3. En esta lección realizará las siguientes tareas:

- Crear una clase de prueba Java.
- Editar la clase utilizando el código que se ofrece a continuación.
- Ejecutar la clase de prueba para probar la aplicación.

1. **Primero creará una clase de prueba Java:** expanda el proyecto **InOutArray**, expanda **Recursos Java** y seleccione el paquete **sample.ims**.
2. Pulse con el botón derecho del ratón y seleccione **Nuevo**. Seleccione la opción de clase  para crear una clase Java nueva.
3. En el nombre de clase Java, escriba `TestInOutProxy`. Tenga en cuenta que la clase TestInOutProxy sólo se ofrece como ejemplo; deberá cambiar el código de transacción para sus especificaciones de máquina IMS. Póngase en contacto con su administrador IMS para conocer el código de transacción. Puede localizar esta sentencia `input.setWs__trcd("SKS7 ");` en el código para efectuar los cambios.
4. Asegúrese de que el campo **Carpeta fuente** contiene **InOutArray/JavaSource** y que el campo **Nombre de paquete** contiene **sample.ims.data**.
5. Pulse **Terminar**.
6. Pulse dos veces en **TestInOutProxy** para abrir el archivo en el editor Java.
7. Copie todo el código que encontrará a continuación y péguelo en la clase **TestInOutProxy.java**. Sustituya todo el código existente en el editor:

   Nota: La clase Java `TestInOutProxy.java` se creó para un entorno local inglés; es posible que deba hacer modificaciones en el código para otros entornos locales.

```
/*
 * Created on 4-Oct-2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package sample.ims;
import sample.ims.data.*;
import com.ibm.connector2.ims.ico.IMSDFSMessageException;

/**
 * @author ivyho
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class TestInOutProxy
{
 public static void main (String[] args)
 {
  try
  {
   // ------------------------------------------------
   // Create the formatHandler, then create the input
   // message bean from the formatHandler.
   // ------------------------------------------------
   INOUTMSG input = new INOUTMSG();

   int sz = input.getSize();
   System.out.println("\nInitial size of input message is: " + sz);

   // ------------------------------------------------
   // Don't set the length (LL) field yet... wait until
```

```
// input message has been adjusted to reflect only
// the number of array elements actually sent.
// ------------------------------------------------
input.setWs__zz((short) 0);
input.setWs__trcd("SKS7 ");

// ------------------------------------------------
// Construct an array and populate it with the elements
// to be sent to the IMS application program.  In this
// case three elements are sent.
// ------------------------------------------------
Inoutmsg_ws__customer[] customers = new Inoutmsg_ws__customer[3];

Inoutmsg_ws__customer aCustomer1 = new Inoutmsg_ws__customer();
aCustomer1.setWs__cust__name("Cathy Tang");
aCustomer1.setWs__cust__number("CN001");
customers[0] = aCustomer1;

Inoutmsg_ws__customer aCustomer2 = new Inoutmsg_ws__customer();
aCustomer2.setWs__cust__name("Haley Fung");
aCustomer2.setWs__cust__number("CN002");
customers[1] = aCustomer2;

Inoutmsg_ws__customer aCustomer3 = new Inoutmsg_ws__customer();
aCustomer3.setWs__cust__name("Steve Kuo");
aCustomer3.setWs__cust__number("CN003");
customers[2] = aCustomer3;

// ------------------------------------------------
// Set the array on the input message.
// ------------------------------------------------
input.setWs__customer(customers);
input.setIndx((short) 3);

System.out.println("\nInitial value of INDX is: " + input.getIndx());

// ------------------------------------------------
// Reallocate the buffer to the actual size
// ------------------------------------------------
byte[] bytes = input.getBytes();
int size = input.getSize();
byte[] newBytes = new byte[size];
System.arraycopy(bytes, 0, newBytes, 0, size);

// ------------------------------------------------
// Set the bytes back into the format handler and set
// the length field of the input message, now that
// we know the actual size.
// ------------------------------------------------
input.setBytes(newBytes);
input.setWs__ll((short) size);
System.out.println("\nAdjusted size of input message is: " + size);
System.out.println("\nAdjusted size of INDX is: " + input.getIndx());

// ------------------------------------------------
// Set fields that follow the array after the input
// message has been adjusted.
// ------------------------------------------------
input.setWs__func__code("123456");

InOutImpl proxy = new InOutImpl();

INOUTMSG output = new sample.ims.data.INOUTMSG();
output = proxy.runInOut(input);

short outndx = output.getIndx();
System.out.println("\nOutput value of INDX is: " + outndx);
```

```
    Inoutmsg_ws__customer outArray[] = output.getWs__customer();

    for (int i = 0; i < outndx; i++)
    {
     System.out.println(
      "\n"
       + outArray[i].getWs__cust__name()
       + outArray[i].getWs__cust__number());
    }
   }
   catch (Exception e)
   {
    if (e instanceof IMSDFSMessageException)
    {
     System.out.println(
      "\nIMS returned message: "
       + ((IMSDFSMessageException) e).getDFSMessage());
    }
    else
    {
     System.out.println(
      "\nIMS Connector exception is: " + e);
    }
   }
  }
 }
```

8. Pulse **Control-S** para guardar los cambios.
9. **Ahora probará la aplicación:** expanda el proyecto **InOutArray** y el paquete **sample.ims**.
10. Pulse con el botón derecho del ratón la clase **TestInOutProxy.java**, expanda **Ejecutar** y seleccione **Ejecutar como < Aplicación Java**.
11. Deberá ver el siguiente mensaje en la consola:

```
Initial size of input message is: 217
Initial value of INDX is: 3
Adjusted size of input message is: 92
Adjusted size of INDX is: 3
Output value of INDX is: 4
Cathy T.              CN001
Haley F.              CN002
Steve K.              CN003
Kevin F.              CN004
```

Enhorabuena. Ha completado la guía de aprendizaje de matriz de entrada y salida.

Resumen de la creación de una aplicación J2C para una transacción IMS que contiene matrices

Esta guía de aprendizaje le ha llevado a través de los pasos detallados para crear una aplicación J2C para una transacción IMS que contenga matrices.

En esta guía de aprendizaje ha aprendido a utilizar el asistente de bean Java J2C para crear una aplicación web sencilla que procese una transacción IMS con datos de entrada y de salida que contengan matrices.

Lecciones aprendidas

Al realizar los ejercicios a aprendido a:

- Utilizar el asistente Bean J2C para crear un bean Java J2C que ejecute una transacción IMS.
- Crear un método Java para el bean, `runInOut.java`, para ejecutar la transacción IMS.
- Crear una clase Java proxy, `TestInOutProxy.java`, para construir el mensaje de entrada de la transacción IMS, invoque el método de bean Java J2C que ejecuta la transacción IMS y visualice los datos de salida devueltos por la transacción IMS.

  Nota: La clase Java `TestInOutProxy.java` se creó para un entorno local inglés; es posible que deba hacer modificaciones en el código para otros entornos locales.