openSAP

Building Apps with the ABAP RESTful Application Programming Model

Week 3 Unit 1

00:00:05 Hola, y bienvenido al curso de openSAP Building Apps con el modelo de programación de aplicaciones RESTful ABAP, semana tres, unidad uno, El escenario empresarial mejorado.

00:00:13 Mi nombre es Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP, y lo guiaré a través de las unidades de esta semana.

00:00:21 En la semana anterior, aprendió a crear una aplicación de informe de lista de viajes de solo lectura paso a paso.

00:00:26 También aprendió cómo crear el modelo de datos CDS, la proyección del modelo de datos y cómo enriquecerlo con semántica de IU.

00:00:33 Además, le mostramos cómo exponer su modelo de datos CDS como servicio OData y cómo consumirlo utilizando la vista previa de la aplicación de elementos de SAP Fiori.

00:00:42 Esta semana habilitaremos las capacidades transaccionales de la aplicación de informes de lista paso a paso.

00:00:48 Comenzaremos creando la definición de comportamiento básica, seguida de la proyección de definición de comportamiento.

00:00:54 Luego realizaremos un pequeño desglose introduciendo el lenguaje de manipulación de entidades necesario para implementar la lógica empresarial real.

00:01:02 Seguiremos ampliando la definición de comportamiento con validaciones, determinaciones y acciones, seguidas de la implementación correspondiente utilizando ABAP.

00:01:13 En la unidad siete, habilitaremos la gestión de borradores para nuestro business object. Por último, examinaremos algunas herramientas y enfoques de resolución de problemas.

00:01:22 En la unidad actual, veremos el flujo de desarrollo y el tiempo de ejecución gestionado que utilizaremos.

00:01:28 La aplicación resultante contendrá funciones de creación, actualización y eliminación. Utilizaremos determinaciones para, por ejemplo, fijar el estado inicial de nuestros viajes.

00:01:37 Las validaciones verificarán la consistencia de los datos introducidos y las acciones nos permitirán aprobar y rechazar viajes.

00:01:44 La aplicación contendrá un control de características estático y dinámico, así como un control de autorización.

00:01:50 La activación de borrador permitirá al usuario final almacenar datos modificados en el back end y continuar en un momento posterior o desde un dispositivo diferente, incluso si la aplicación

00:02:01 finaliza de forma inesperada. Utilizaremos los artefactos que creamos en la semana dos.

00:02:07 La definición de comportamiento se creará para la jerarquía de composición, en nuestro caso las vistas de interfaz CDS de viaje y reserva.

00:02:15 La implementación de comportamiento tiene lugar en ABAP y requiere el uso del lenguaje de manipulación de entidades que forma parte del lenguaje ABAP.

00:02:23 Al igual que la proyección CDS, también proyecta las capacidades transaccionales que ha definido en su definición de comportamiento base, utilizando la proyección de definición de

00:02:33 comportamiento. En el escenario gestionado, el framework de business object implementa la fase de interacción y

00:02:39 la secuencia de grabación de forma genérica. La infraestructura de tiempo de ejecución del business object transfiere los aspectos de la

00:02:48 implementación técnica. Este escenario aborda los casos de uso en los que todas las partes esenciales de una aplicación

00:02:55 se desarrollan desde cero, el llamado enfoque de nuevo campo. Estas nuevas aplicaciones pueden beneficiarse en gran medida del soporte listo para usar para el

00:03:04 procesamiento transaccional mediante el uso del escenario gestionado. Las operaciones estándar, crear, actualizar y borrar solo se deben especificar en la definición

00:03:13 de comportamiento para obtener un business object listo para ejecutarse. El desarrollador de aplicaciones puede centrarse en la lógica empresarial que se implementa

00:03:22 utilizando, por ejemplo, determinaciones, validaciones, acciones e interacción del usuario. El tiempo de ejecución de business object correspondiente gestiona todo el ciclo de vida de sus

00:03:31 business objects y cubre todos los aspectos del desarrollo de su aplicación empresarial. El escenario gestionado es el escenario de implementación recomendado para nuevas aplicaciones.

00:03:41 Ahora estamos al final de esta unidad. En esta unidad, ha aprendido sobre las mejoras transaccionales del escenario empresarial que

00:03:48 implementamos esta semana. El flujo de desarrollo que cubriremos en las siguientes unidades y la implementación del tiempo

00:03:55 de ejecución gestionado. Para más información sobre esta unidad, véase el apéndice de la presente presentación.

00:04:02 Gracias por escucharle y verle en la siguiente unidad, donde crearemos la definición de comportamiento.

Week 3 Unit 2

00:00:06 Hola y bienvenido al curso de openSAP "Creación de aplicaciones con el modelo de programación de aplicaciones RESTful ABAP", semana tres, unidad dos, "Definición del comportamiento básico de business object".

00:00:15 Me llamo Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP. En la unidad anterior, vio un resumen del escenario empresarial ampliado en el que trabajaremos esta semana.

00:00:27 Ahora comenzaremos con la activación de las capacidades transaccionales de nuestra aplicación de informes de lista de viajes de SAP Fiori de solo lectura creando la definición de comportamiento de business object.

00:00:37 El área en la que trabajaremos en esta unidad se resalta en el flujo de desarrollo, la definición de comportamiento que define dichas capacidades de transacción de nuestro business object.

00:00:47 Utilizaremos el menú contextual para crear la definición de comportamiento a partir de la vista de interfaz CDS raíz que creamos en la semana anterior. Al final de esta unidad, dispondremos de la definición de comportamiento base, que define nuestro business object de viaje gestionado con las operaciones

00:01:04 básicas de creación, actualización y borrado, así como las asociaciones necesarias. Ahora entraremos en el sistema, crearemos este nuevo artefacto y probaremos la aplicación.

00:01:14 Las definiciones de comportamiento se crean para la vista CDS raíz y definen el comportamiento para todas las entidades incluidas. Haga clic con el botón derecho en la vista CDS raíz de interfaz y seleccione Nueva definición de comportamiento.

00:01:30 La asignación de paquete se realiza automáticamente, el nombre de la definición de comportamiento debe ser el mismo que la vista CDS raíz. Esa es la razón por la que no se puede cambiar el nombre.

00:01:42 Asegúrese de mantener el tipo de implementación gestionado, haga clic en Siguiente. Asigne una solicitud de transporte y haga clic en Finalizar.

00:01:56 La definición de comportamiento está predeterminada según el tipo de implementación que haya seleccionado anteriormente. Lo primero que hemos definido es un alias para el viaje y la entidad de reserva.

00:02:19 Luego especificamos la persistencia proporcionando los nombres de tabla de base de datos para ambas entidades. Esto permite que el tiempo de ejecución de gestión realice operaciones de creación, actualización y borrado directamente en nuestras tablas de base de

00:02:48 datos. Lo siguiente es especificar el maestro de bloqueos para la entidad raíz.

00:02:54 Para ello, anule el comentario de la línea "Bloquear maestro". El nodo subordinado de reserva se convierte en un bloqueo dependiente y utiliza la asociación definida en la vista CDS para habilitar de forma transaccional la

00:03:21 asociación de viajes enumerada explícitamente en la entidad de reserva. Como se trata de un escenario basado en UUID, queremos que el tiempo de ejecución gestionado proporcione una clave cuando se creen nuevas instancias.

00:03:38 Para ello, necesitamos que la numeración de TravelUUID se gestione y que sea de solo lectura. Lo mismo es necesario para la entidad de reserva.

00:03:57 Hay una advertencia que indica que el UUID de viaje en la entidad de reserva debe fijarse en de solo lectura ya que se utiliza en la condición ON de la asociación.

00:04:07 Para resolver esto, agregamos la siguiente línea. En nuestro business object gestionado, queremos hacer uso del llamado bloqueo optimista.

00:04:21 Esto se realiza definiendo el maestro ETag en ambas entidades. La definición de dos maestros de etiquetas electrónicas tiene lugar a propósito.

00:04:42 Es un enfoque recomendado para tener una etiqueta electrónica local para cada entidad. Esto se define especificando un maestro de ETag en cada nodo.

00:04:50 El último mensaje de advertencia que queda es la información de asignación que falta. Dado que hemos proporcionado alias en las vistas CDS de interfaz para los nombres de elemento, tenemos que decirle al framework cómo asignar los nombres

00:05:09 de elemento en el modelo de datos CDS a los campos de tabla correspondientes. Lo mismo se debe hacer para la entidad de reserva.

00:05:30 Grabe y active la definición de comportamiento. Inicie o actualice su aplicación.

00:05:42 Como resultado, no verá ninguna modificación: sin creación, sin actualización, sin opción de borrado. Esto se debe a que aún no hemos proyectado nuestra definición de comportamiento y, por lo tanto, todavía tenemos una aplicación de solo lectura.

00:05:57 Eso fue para la demostración del sistema. Ahora estamos al final de esta unidad.

00:06:00 Permítanme hacer una breve recapitulación. En esta unidad, ha aprendido a utilizar las herramientas de desarrollo ABAP para crear la definición de comportamiento base para nuestro business object de viaje

00:06:10 gestionado y definir el comportamiento para las entidades de viaje y reserva. Para más información sobre esta unidad, véase el apéndice de la presente presentación.

00:06:20 Gracias por escuchar y verle en la siguiente unidad sobre la creación de la proyección de comportamiento de business object.

Week 3 Unit 3

00:00:05 Hola, y bienvenido al curso de openSAP Building Apps con el modelo de programación de aplicaciones RESTful ABAP Semana tres, unidad tres, para crear la proyección de comportamiento de business object.

00:00:16 Me llamo Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP.

00:00:21 En la unidad anterior, utilizamos las herramientas de desarrollo ABAP para crear la definición de comportamiento base para nuestro business object de viajes gestionados, definiendo el comportamiento para

00:00:31 las entidades de viaje y reserva. Ahora proyectaremos las capacidades transaccionales que queremos habilitar en nuestra aplicación creando

00:00:39 la proyección de comportamiento de business object. El área en la que trabajaremos en esta unidad se resalta en el flujo de desarrollo, la proyección de

00:00:47 comportamiento, que proyecta las capacidades transaccionales desde la definición de comportamiento base.

00:00:52 Utilizaremos el menú contextual para crear la definición de comportamiento a partir de la vista de proyección CDS de raíces que hemos creado en la semana anterior.

00:01:01 Al final de esta unidad, tendremos la proyección de definición de comportamiento proyectando las operaciones de creación, actualización y borrado de nuestro business object de gestión de viajes.

00:01:12 Ahora pasemos al sistema, crearemos este nuevo artefacto y probaremos la aplicación. Haga clic con el botón derecho en la vista CDS raíz de consumo y seleccione Nueva definición de

00:01:26 comportamiento. El nombre de la definición de comportamiento debe ser el mismo que el de la vista CDS raíz.

00:01:32 Esa es la razón por la que no se puede cambiar el nombre. Asegúrese de mantener el tipo de implementación como proyección.

00:01:39 Haga clic en Siguiente. Asigne una solicitud de transporte y haga clic en Finalizar.

00:01:48 La definición de comportamiento está predeterminada según el tipo de implementación que haya elegido anteriormente.

00:01:53 Las operaciones de creación, actualización y borrado se transfieren automáticamente de la definición de comportamiento base mediante la palabra clave "use".

00:02:02 Lo primero que definimos es el alias para la entidad Viajes y Reserva. También queremos activar el tratamiento de ETag, es necesario añadirlo para todas las entidades.

00:02:33 Grabe y active la definición de comportamiento. Inicie o actualice su aplicación.

00:02:43 En el entorno de prueba, puede tardar un poco hasta que se reflejen los cambios. Como resultado, verá que ahora las operaciones de creación, actualización y borrado ahora están

00:02:59 habilitadas. No dude en jugar con su aplicación.

00:03:04 Eso fue para la demostración del sistema. Ahora estamos al final de esta unidad.

00:03:08 Permítanme hacer una breve recapitulación. En esta unidad, ha aprendido a utilizar las herramientas de desarrollo ABAP para crear la

00:03:14 proyección de definición de comportamiento para nuestro business object de viaje gestionado, proyectando el comportamiento para las entidades Viajes y Reservas.

00:03:23 Para más información sobre esta unidad, véase el apéndice de la presente presentación. Gracias por escucharle y verle en la siguiente unidad con una introducción al lenguaje de manipulación de

00:03:32 entidades.

Week 3 Unit 4

00:00:05 Hola, y bienvenido al curso de openSAP Building Apps con el modelo de programación de aplicaciones RESTful ABAP, semana tres, unidad cuatro, Lenguaje de manipulación de entidades.

00:00:15 Me llamo Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP.

00:00:20 En las unidades anteriores, utilizamos las herramientas de desarrollo ABAP para crear la definición de comportamiento base y la proyección de definición de comportamiento para nuestro business

00:00:28 object problemático. En estas definiciones de comportamiento, utilizamos las capacidades estándar de creación, actualización y

00:00:35 borrado que no requieren su propia implementación ya que se gestionan en el tiempo de ejecución gestionado por RAP.

00:00:42 En esta unidad, proporcionamos una introducción al lenguaje de manipulación de entidades que necesitará cuando, por ejemplo, añada determinaciones, validaciones o acciones a su definición de

00:00:51 comportamiento de objeto empresarial. El lenguaje de manipulación de entidades, en pocas palabras, EML, forma parte del lenguaje ABAP y le

00:00:59 permite interactuar con sus propios y otros business objects en una sintaxis similar a SQL. Le permite controlar el comportamiento de los business objects en el contexto del modelo de

00:01:09 programación de aplicación RESTful ABAP. El EML también le proporciona acceso basado en API a otros objetos empresariales RAP.

00:01:17 El EML estándar proporciona una lectura segura de tipo y la modificación del acceso a los datos y escenarios de desarrollo transaccional.

00:01:23 Este es el caso de uso más común. Para implementaciones genéricas, por ejemplo, al desarrollar sus propios marcos, también puede

00:01:32 utilizar una implementación EML genérica. El tiempo de ejecución ABAP garantiza la consistencia transaccional similar a la unidad lógica de trabajo

00:01:41 que conoce de SQL. La primera operación que observaremos es la operación de lectura, que permite el rendimiento de una lectura

00:01:48 transaccional de instancias de business object. Lectura transaccional significa que la lectura se realiza desde la memoria intermedia transaccional.

00:01:55 Si la instancia no existe en la memoria intermedia, el tiempo de ejecución gestionado la leerá automáticamente en la memoria intermedia de la base de datos.

00:02:02 Como todas las sentencias EML están habilitadas en masa de forma predeterminada, debe especificar una tabla de claves para la que desea realizar la operación de lectura.

00:02:10 Esto le permite realizar la lectura para un conjunto de instancias con una única columna EML. Junto al resultado real, las sentencias EML también proporcionan información fallida y notificada.

00:02:21 Reflejan las claves para las que se solicita que la operación no se haya podido ejecutar. Por ejemplo, leer una instancia de viaje que no existe.

00:02:29 Informado opcionalmente contiene información explicativa como mensajes T100. Las operaciones de modificación se utilizan para realizar modificaciones en la memoria intermedia

00:02:39 transaccional. Si un registro no existe en la memoria intermedia, se leerá de la base de datos antes de que se ejecute la

00:02:44 operación. La creación de modificación se utiliza para crear una instancia raíz.

00:02:49 A través de la llamada creación por asociación, también se puede utilizar para crear instancias secundarias a lo largo de asociaciones.

00:02:55 Tenga en cuenta que las operaciones EML se ejecutan en la memoria intermedia transaccional. Solo se graban de forma persistente a través de la base de datos cuando se realiza la declaración de

00:03:03 entidad commit. La actualización de modificación se utiliza para actualizar instancias.

00:03:07 También puede actualizar instancias que se han creado anteriormente y que aún no se han grabado de forma persistente en la base de datos.

00:03:13 Para ello, debe utilizar el ID de contenido. La eliminación de modificaciones se utiliza para eliminar instancias.

00:03:20 La operación de ejecución de modificación se utiliza para ejecutar acciones. A diferencia de las funciones, las acciones pueden modificar datos y, por lo tanto, se realizan como

00:03:28 parte de las operaciones de modificación. Vamos a entrar en el sistema, crear una clase pequeña y realizar algunas llamadas EML a los datos

00:03:38 existentes. Haga clic con el botón derecho en su paquete y seleccione Nuevo, Clase ABAP.

00:03:46 Introduzca el nombre, ZCL_RAP_EML y su ID de grupo, y proporcione una descripción significativa. Haga clic en Siguiente.

00:04:00 Asigne una solicitud de transporte y haga clic en Finalizar. En la sección pública, añada la interfaz if_oo_adt_class run.

00:04:18 Utilice la corrección rápida para añadir la implementación para el método principal. El primer ejemplo EML realiza una operación de lectura.

00:04:35 Para ello, utilizamos la declaración de entidad leída, seguida de nuestro objeto empresarial RAP Travel.

00:04:52 A continuación, tenemos que especificar la entidad para la que queremos realizar la operación, en nuestro caso, la entidad Travel.

00:05:03 Como todas las sentencias EML están habilitadas en masa por defecto, es necesario especificar una lista de claves para las que queremos realizar la operación de lectura.

00:05:21 El resultado de la lectura se colocará en la tabla en línea declarada. Por último, utilizamos la consola para mostrar el contenido de la tabla de resultados.

00:05:48 Antes de poder continuar, necesitamos el UUID de viaje. Abra la vista previa de tabla de su tabla de viajes y seleccione un UUID que desee utilizar en este

00:05:58 ejemplo. Haga doble clic en el valor para elegir solo el UUID.

00:06:15 Recuerde los ID de viaje asociados para que vuelva a encontrar el registro más adelante. Pegue el UUID en el marcador de posición.

00:06:36 Asegure y active y ejecute la clase como una aplicación de consola. Se realiza la lectura, pero solo se completa el campo clave.

00:06:56 Esto se debe a que no hemos especificado los campos que queremos leer y las claves se proporcionan por defecto.

00:07:04 En el siguiente ejemplo, denominamos el campo que queremos leer utilizando el suplemento de campos. Grabe y active y ejecute la clase como una aplicación de consola.

00:07:46 Ahora el ID de agencia y el ID de cliente también se proporcionan en el resultado. También es posible leer todos los campos de la entidad utilizando todos los campos con adición.

00:08:17 Grabe y active y ejecute la clase como una aplicación de consola. Ahora se devuelven todos los campos de la entidad.

00:08:33 Otra opción es realizar una lectura por asociación siguiendo la asociación definida en la definición de comportamiento.

00:09:00 Grabe y active las modificaciones y ejecútelas de nuevo como aplicación de consola. Como resultado, obtenemos todas las reservas con todos los campos asociados a este viaje.

00:09:15 Al realizar operaciones de lectura EML, no solo debe tener en cuenta la tabla de resultados, sino también las tablas fallidas e informadas.

00:09:24 Fallido se utiliza para transmitir operaciones fallidas. Notificado como opcionalmente utilizado para proporcionar mensajes T100 relacionados.

00:09:32 Intentamos realizar una lectura de un UUID que no existe. La tabla de resultados está vacía, las tablas fallidas y notificadas son tablas anidadas y la

00:10:06 salida de la consola no las admite. Por lo tanto, pongamos un punto de ruptura y verifiquemos el contenido de las tablas.

00:10:35 Fallido contiene una entrada para la lectura incorrecta con una causa fallida NOT_FOUND. La tabla notificada está vacía porque la causa del error explica suficientemente el motivo.

00:10:53 Ahora veamos cómo se modifican las operaciones. Vamos a realizar una modificación, actualizando la descripción de nuestro registro de viajes.

00:11:27 Grabe y active y ejecute la clase como una aplicación de consola. Utilice la aplicación de elementos Fiori para verificar la descripción actualizada.

00:11:58 Como podemos ver, la descripción sigue siendo la antigua. Esto se debe a que no hemos realizado un compromiso de nuestros cambios.

00:12:08 Añada la sentencia COMMIT. Grabe y active y ejecute la clase como una aplicación de consola.

00:12:32 Verifique el resultado en la aplicación de elementos Fiori. Ahora vamos a crear un nuevo registro de viaje utilizando la declaración de creación de

00:13:06 modificación. Actívela y ejecútela como una aplicación de consola.

00:13:22 Verifique el resultado en la aplicación de elementos Fiori. Para las operaciones que crean instancias, se devuelve una tabla asignada que asigna la instancia

00:13:50 creada al ID de contenido proporcionado. Los borrados se realizan mediante la sentencia de borrado modificada.

00:14:24 Active y ejecute la clase como una aplicación de consola. Verifique el resultado en la aplicación de elementos Fiori.

00:14:45 El registro borrado ha desaparecido. Las acciones también se pueden ejecutar utilizando el lenguaje de manipulación de entidades una vez que las

00:14:54 acciones se definen en la definición de comportamiento. Eso fue para la demostración del sistema.

00:14:59 Ahora estamos al final de esta unidad. Permítanme hacer una breve recapitulación.

00:15:02 En esta unidad, ha aprendido sobre la sintaxis básica del lenguaje de manipulación de entidad y cómo le permite leer y modificar datos transaccionales.

00:15:11 Para más información sobre esta unidad, véase el apéndice de la presente presentación. Gracias por escucharle y verle en la siguiente unidad, donde mejoramos el comportamiento del

00:15:20 business object de viaje y, posteriormente, utilizamos EML para implementar la lógica empresarial correspondiente.

Week 3 Unit 5

00:00:05 Hola, y bienvenido al curso de openSAP Creación de aplicaciones con el modelo de programación de aplicaciones RESTful ABAP, semana tres, unidad cinco, Mejora del comportamiento de business

00:00:14 object con lógica específica de aplicación. Me llamo Volker Drees y soy un experto en productos para el modelo de programación de

00:00:20 aplicaciones RESTful ABAP. En la unidad anterior, aprendió cómo utilizar el lenguaje de manipulación de entidad para

00:00:26 interactuar con la memoria intermedia transaccional de su business object de viaje. Ahora mejoraremos el comportamiento de nuestro business object definiendo determinaciones,

00:00:35 validaciones y acciones. También agregaremos control de funciones estático y dinámico.

00:00:40 En la siguiente unidad, realizaremos la implementación de la lógica empresarial correspondiente.

00:00:46 El área en la que trabajaremos en esta unidad se resalta en el piso de desarrollo, la definición del comportamiento y la protección del comportamiento.

00:00:55 La definición de comportamiento basada en el resultado contendrá el control de funciones estático, el control dinámico de funciones, las acciones, las determinaciones y las validaciones.

00:01:04 Ahora entraremos en el sistema y mejoraremos nuestras definiciones de comportamiento. Abra la definición de comportamiento base de su business object.

00:01:16 Queremos separar la implementación de comportamiento en una clase ABAP por entidad. Para ello, movemos la implementación y la declaración de clase de la sección de cabecera a

00:01:36 la entidad Viajes y la duplicamos para la entidad Reserva. Añada un control de campo estático configurando los campos ID de viaje, precio total y estado de

00:01:52 viaje como de solo lectura. Proceda del mismo modo para los campos administrativos.

00:02:02 Establezca los campos ID de agencia e ID personalizado como obligatorios. Esto mostrará el pequeño símbolo de estrella roja en la IU de elementos Fiori.

00:02:10 Tenga en cuenta que esto aún requiere implementar la validación correspondiente. Para aprobar y rechazar viajes, definimos dos medidas con control de función dinámico, ambas

00:02:24 devuelven $mismo con cardinalidad uno como resultado. $auto-significa que se devuelve la instancia del mismo tipo en la que se realiza la operación.

00:02:38 Para el recálculo del precio total, definimos una acción interna, recalcular el precio total. La determinación de status inicial fijada se utiliza para fijar el status por defecto en n cada

00:02:54 vez que se crea una instancia nueva. Utiliza el desencadenador de creación al modificar.

00:03:03 La determinación del precio total se utiliza para actualizar el precio total siempre que se modifique la tasa de reserva o el código de moneda.

00:03:16 El identificador de viaje calculado se utiliza para determinar el ID de viaje cuando se crea una nueva instancia.

00:03:22 Se ejecuta al grabar. Tenga en cuenta que el cálculo del ID de viaje solo se utiliza para fines de demostración.

00:03:29 La clave primaria del business object de viaje sigue siendo el UUID de viaje. Definimos tres validaciones para validar la entrada de los campos ID de agencia, ID de

00:03:43 cliente, fecha de inicio y fecha de fin. Todos se inician al grabar.

00:03:53 Para la entidad Reserva , hemos especificado el control de función estática para el ID de reserva además del UUID de viaje.

00:04:11 Además, también definimos los campos administrativos creados por, modificado por última vez por y local modificados por como solo lectura.

00:04:23 Para determinar el ID de reserva, se añade la determinación de ID de reserva calculada. Para actualizar el precio total, agregamos el precio total calculado, que se desencadena cada

00:04:42 vez que se modifica el precio de inundación o el código de moneda. Grabe y active la definición de comportamiento.

00:04:50 Trataremos las advertencias en la siguiente unidad cuando llevemos a cabo la implementación de comportamiento.

00:05:01 Abra la definición de comportamiento de proyección de su business object. Proyecte las dos nuevas acciones añadiendo las declaraciones de uso correspondientes.

00:05:19 Grabe y active la definición de comportamiento. Abra la extensión de metadatos de la entidad Travel.

00:05:34 Desplácese hacia abajo hasta el elemento Status de viaje. Sustituya la partida individual existente y la anotación de identificación con las siguientes

00:05:52 líneas. Esto es para añadir las acciones recientemente definidas, aceptar viajes y rechazar viajes en la

00:05:59 IU de elementos Fiori. Grabe y active la ampliación de metadatos.

00:06:06 Eso fue para la demostración del sistema. Ahora estamos al final de esta unidad.

00:06:10 Permítanme hacer una breve recapitulación. En esta unidad, ha aprendido cómo ampliar la definición de comportamiento de business object y

00:06:16 cómo proyectar las ampliaciones relevantes en la proyección de comportamiento. Para más información sobre esta unidad, véase el apéndice de la presente presentación.

00:06:27 Gracias por escuchar y verle en la siguiente unidad, donde implementaremos la lógica empresarial para las mejoras que acabamos de definir.

Week 3 Unit 6

00:00:05 Hola y bienvenido al curso de openSAP “Building Apps with the ABAP RESTful Application Programming Model”, semana tres, unidad seis, “Implementación del comportamiento de business object”.

00:00:15 Me llamo Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP. En la unidad anterior, aprendió cómo ampliar el comportamiento del business object y cómo proyectar las ampliaciones relevantes mediante su definición de comportamiento

00:00:29 de proyección. Ahora realizaremos la implementación de la lógica empresarial correspondiente.

00:00:35 El área en la que trabajaremos en esta unidad se resalta en el flujo de desarrollo, la implementación del comportamiento. Utilizaremos el soporte de las herramientas de desarrollo ABAP para generar clases de implementación de comportamiento para cada uno de los nodos de business object,

00:00:49 viajes y reservas. Técnicamente, estas clases son pools de clases, pero en el contexto de RAP, las llamamos pools de comportamiento.

00:00:56 Estas clases contienen las firmas de método necesarias y las implementaciones de método vacías para ayudarnos con la tarea de implementación. Los pools de comportamiento resultantes contienen toda la lógica empresarial necesaria para las ampliaciones definidas en la unidad anterior.

00:01:14 Ahora entraremos en el sistema, realizaremos la implementación de comportamiento y probaremos el SAP mejorado. En la implementación de comportamiento, utilizaremos los mensajes T100 propios emitidos por una clase de excepción propia.

00:01:29 Por eso primero necesitamos crear una clase de mensaje propia. Haga clic con el botón derecho en el paquete.

00:01:35 Seleccione "Nuevo", "Otro objeto de ABAP Repository". Escriba el filtro "mensaje" y seleccione Clase de mensaje, haga clic en Siguiente.

00:01:50 Introduzca el nombre "zrap_msg" seguido de su ID de grupo. Proporcione una descripción significativa, haga clic en Siguiente y asigne la clase de mensaje a una solicitud de transporte.

00:02:16 Cree cinco mensajes con los números uno a cinco. Introduzca los siguientes textos breves.

00:02:49 Grabe los nuevos textos de mensaje. Haga clic con el botón derecho en su paquete y seleccione "Nuevo", "Clase ABAP".

00:03:05 Introduzca el nombre "ZCM_RAP" seguido de su ID de grupo. Y proporcione una descripción significativa.

00:03:22 La clase superior debe ser CX_STATIC_CHECK, haga clic en Siguiente. Asigne la solicitud de transporte y haga clic en Finalizar.

00:03:40 En la sección pública, añada la interfaz if_abap_behv_message, agregue las constantes para los cinco mensajes tal como se proporcionan en el fragmento de código. No olvide sustituir el marcador de posición por su ID de grupo individual.

00:04:17 Sustituya el constructor por el siguiente fragmento. Defina las variables necesarias.

00:04:38 En el constructor, asigne los parámetros de importación a las propiedades. Tenga en cuenta que utilizamos la conversión alfa para todos los parámetros del tipo NUMC, grabamos y activamos la clase ABAP.

00:05:06 Abra la definición de comportamiento base de nuestro business object de viaje. Utilice la corrección rápida para generar la implementación de comportamiento para la entidad de viaje.

00:05:21 En el asistente, deje los campos predeterminados y haga clic en Siguiente. Asigne una solicitud de transporte y haga clic en Finalizar.

00:05:45 La implementación de comportamiento tiene lugar en la etiqueta Tipos locales del pool de comportamiento generado. La herramienta ha generado una clase local que hereda de cl_abap_conducor_Handler.

00:06:00 En la sección privada, añada constantes para el estado del viaje. El primer método que vamos a implementar es el método aceptTravel.

00:06:13 Fija el status en Aceptado para todas las claves proporcionadas mediante una sentencia de modificación EML. Utiliza la adición "en modo local", que, por ejemplo, nos permite incluso modificar campos de solo lectura mientras omitimos la característica y el control de

00:06:45 autorización. Después de modificar el estado del viaje, la acción debe proporcionar el resultado como se define en la definición de comportamiento.

00:06:53 En nuestro caso, el resultado es $ti mismo, lo que significa que debemos devolver la instancia de viaje con todos los valores para las claves dadas. Tenga en cuenta la utilización de %tky, que representa la clave transaccional.

00:07:20 En el caso de utilización no borrador, contiene el mismo valor que la clave de porcentaje, que es la clave de la entidad relacionada. En la siguiente unidad, cuando habilitamos la gestión de borradores, la clave de transacción contendrá automáticamente el indicador is_borrador.

00:07:35 El uso de una clave transaccional reduce la necesidad de volver a trabajar con la implementación cuando se habilita el borrador en su definición de comportamiento, ya que el código puede hacer frente a instancias activas y de borrador.

00:07:48 Muy similar a la acción de aceptar viaje es la acción de rechazar viaje. Fija el status del viaje en Rechazado.

00:08:11 La primera validación que vamos a implementar es el método validateAgency - esta validación verifica el ID de agencia proporcionado al guardar si la idea de agencia se modificó o cuando se crea una instancia.

00:08:25 Las implementaciones de validación normalmente empiezan con la lectura de los datos necesarios mediante EML. En nuestro caso, queremos leer el ID de agencia para las claves proporcionadas.

00:08:54 Derivamos una tabla interna con todos los ID de agencia distintos y realizamos una selección de base de datos para validar la existencia. A continuación, revisamos la tabla de viajes y verificamos si se ha proporcionado un ID de agencia y si existe.

00:09:30 Si el ID de agencia está vacío o no existe en la tabla de verificación, emitimos un mensaje utilizando nuestra clase de excepción. A medida que utilizamos los llamados mensajes estatales, primero tenemos que aclarar los mensajes existentes para el ámbito estatal de este caso y, en caso necesario,

00:09:48 plantear otros nuevos. Los mensajes de estado son importantes en el contexto del borrador, donde los mensajes se almacenan junto con el estado de la instancia.

00:09:58 En un caso de utilización no borrador, el framework convierte los mensajes de estado en mensajes transitorios. En la validación del cliente, seguimos un enfoque muy similar al del método validateAgency.

00:10:32 Leemos el ID de cliente, verificamos la existencia en la tabla de base de datos y emitimos mensajes de estado si se proporcionó un ID de cliente no válido. La tercera validación son las fechas de validación una.

00:10:47 Valida la fecha de inicio y de fin del viaje. La primera determinación que vamos a implementar es la determinación setInicialStatus.

00:11:12 Fija el estado del viaje como abierto cuando se crea una nueva instancia. Tenga en cuenta que las determinaciones deben ser idempotentes, lo que significa que el resultado no puede diferir aunque se ejecuten varias veces para la misma clave.

00:11:49 La implementación lee primero el status de viaje para todas las claves indicadas y fija el status en abierto para aquellos que aún no tienen ningún status. La determinación calculateTravelID se utiliza como ejemplo para demostrar una determinación al guardar y para proporcionar el identificador de viaje legible.

00:12:08 Tenga en cuenta que la clave del business object de viaje sigue siendo el UUID de viaje. En lugar de utilizar un rango de números, la implementación simplemente obtiene el ID de viaje más alto existente de la tabla de base de datos y aumenta el valor en uno.

00:12:41 Tenga en cuenta que este enfoque no garantiza los ID únicos o sin discontinuidades. El siguiente método que vamos a implementar es calcular el método TotalPrice.

00:12:52 Se trata de una determinación "on-Change" para los campos BookingRate y CurrencyCode. La implementación llama la acción interna recalcTotalPrice para todas las claves de viaje proporcionadas.

00:13:23 La acción interna recalcTotalPrice calcula el precio total mirando todas las reservas, cobrando todos los precios del vuelo y sumando la tasa de reserva. En el caso de diferentes códigos de moneda, también realiza la conversión de moneda.

00:13:38 La lectura de las reservas para una instancia de viaje se realiza mediante una asociación read-by. El método get_functions se utiliza para el control de características específico de instancia.

00:14:09 Espera en la tabla de resultados una entrada para cada clave solicitada, especificando si la función relacionada está habilitada o desactivada según el estado de la instancia.

00:14:37 En nuestro ejemplo, hablamos de aceptar viajes y rechazar acciones de viaje basadas en el estado actual. Grabe y active la implementación de comportamiento.

00:14:53 Abra la definición de comportamiento base de nuestro business object. Utilice la corrección rápida para generar la implementación de comportamiento para la entidad de reserva.

00:15:13 En el asistente, deje los campos predeterminados y haga clic en Siguiente. Asigne una solicitud de transporte y haga clic en Finalizar.

00:15:30 La determinación calculateBookingID se realiza al modificar cuando se crea una nueva instancia. Primero utiliza una asociación read-by para leer la cabecera de viaje relacionada.

00:15:57 Para cada instancia de viaje, se leen todas las reservas para encontrar el ID de reserva más alto y para calcular los ID de reserva para las entradas que aún no contienen un ID de reserva.

00:16:12 La determinación calculateTotalPrice llama a la acción interna recalc total price, que definimos e implementamos en el nivel raíz. Grabe y active la implementación de comportamiento.

00:16:42 Ahora vamos a añadir el control de autorización a nuestro business object. Abra la definición de comportamiento base.

00:16:54 Anule el comentario de la línea "registro maestro de autorización (instancia)". Indica el nodo raíz como maestro de autorizaciones.

00:17:02 Las solicitudes de autorización específicas de instancia se dirigen a la implementación del maestro de autorizaciones. Desplácese hasta la entidad de reserva y anule el comentario de la línea dependiente de autorización y especifique la asociación _Travel.

00:17:31 Esto declara que la entidad de reserva depende de la autorización, y las solicitudes de modificación a la entidad de reserva se dirigen al maestro de autorizaciones después de la asociación proporcionada.

00:17:43 Grabe y active la definición de comportamiento. Sitúe el cursor en la palabra clave maestra de la declaración maestra de autorización y utilice la corrección rápida para añadir la nueva declaración de método a la

00:18:12 implementación. Este patrón también funciona para otras declaraciones.

00:18:17 Si, por ejemplo, añade una acción nueva, puede utilizar la corrección rápida para permitir que las herramientas de desarrollo ABAP generen la declaración de método correspondiente, así como una implementación de método vacía.

00:18:31 En el entorno de prueba, no podemos influir en las autorizaciones reales, por lo que necesitamos simular si una determinada autorización, por ejemplo, la autorización de actualización, está presente o no.

00:18:42 Esto se hace a través de algunos métodos de ayuda que primero tenemos que declarar. Utilice la corrección rápida para generar las implementaciones para los tres métodos.

00:19:08 El método de ayuda is_create_sidy realiza una verificación de autorización utilizando el objeto de autorización que se definió en la semana anterior. Verifica si la creación está permitida, que es la actividad 01.

00:19:27 Con fines de prueba, fijamos los resultados en verdadero en todos los casos. Por supuesto, esto se puede ajustar para probar diferentes situaciones.

00:19:36 Al igual que la creación, se implementa el método auxiliar "update-check". La actualización está asociada a la actividad 02.

00:19:51 Si ya existe un registro, se utilizará el status de viaje del Before-Image en la verificación. El método "delete-helper" verifica la actividad 06.

00:20:19 Ahora podemos implementar el método get_Approvals. Simula el caso práctico de utilizar un campo editable, estado de viaje, para controlar la autorización.

00:20:41 Por lo tanto, la implementación lee el Before-Image de la persistencia activa y utiliza este valor al llamar los métodos auxiliares. Las diferentes opciones mediante la estructura de autorización solicitada se asignan a los valores de actividad, se crean, actualizan y borran, definidos en el objeto de

00:21:03 autorización. A continuación, se verifican para cada instancia de viaje.

00:21:08 La tabla de resultados se estructura en función del resultado de las respectivas verificaciones de autorización. Grabe y active la implementación de comportamiento.

00:21:26 Ahora hemos terminado con la implementación y podemos ejecutar la IU de elementos Fiori. Primero, verá dos nuevas acciones: Aceptar viaje y Rechazar viaje.

00:21:38 Se activan según el estado de la instancia de viaje seleccionada. Cree un viaje nuevo.

00:21:53 Proporcione algunos valores inconsistentes para el ID de agencia, el ID de cliente, la fecha de inicio y la fecha de fin. Al hacer clic en Grabar, se muestran los mensajes correspondientes.

00:22:15 Corrija los valores. Y haga clic en Grabar.

00:22:32 Ahora, las validaciones se han pasado correctamente. También puede ver que se han ejecutado las determinaciones, que se ha calculado la identificación del viaje y que el estado de la reserva se ha predeterminado.

00:22:49 Utilice las acciones Aceptar y Rechazar para verificar el controlador de funciones específico de instancia. En la tabla de reservas, haga clic en Crear para crear una nueva instancia de reserva para este viaje.

00:23:07 Proporcione algunas entradas válidas y haga clic en Guardar. Navegar hacia atrás muestra que la instancia de cabecera no se ha vuelto a cargar y, por lo tanto, se muestra el precio total anterior.

00:23:37 Esto requeriría la definición de un efecto secundario, que no estamos abordando en esta sesión. Una actualización manual de la IU muestra el precio total actualizado.

00:23:55 Puede hacer clic en Borrar para borrar la instancia de viaje. Eso fue para la demostración del sistema.

00:24:08 Ahora estamos al final de esta unidad. Permítanme hacer una breve recapitulación.

00:24:12 En esta unidad, ha aprendido a utilizar las herramientas de desarrollo ABAP para generar los pools de comportamiento para las entidades de viaje y reserva, cómo implementar los métodos correspondientes y cómo la aplicación mejorada proporciona una mejor experiencia de usuario.

00:24:27 Para más información sobre esta unidad, véase el apéndice de la presente presentación. Gracias por escucharle y verle en la siguiente unidad, donde añadimos la gestión de borradores a nuestro business object.

Week 3 Unit 7

00:00:05 Hola, y bienvenido al curso de openSAP Building Apps con ABAP RESTful Application Programming Model, semana tres, unidad siete, Habilitación del tratamiento de borradores.

00:00:15 Me llamo Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP. En la unidad anterior, aprendió cómo implementar la lógica empresarial para las mejoras de comportamiento de nuestro business object de viaje.

00:00:28 Ahora habilitaremos la gestión de borradores. Las aplicaciones habilitadas para borradores permiten al usuario final almacenar los datos de modificación en el back end y continuar en un momento posterior o desde un

00:00:38 dispositivo diferente, incluso si la aplicación finaliza de forma inesperada. Este tipo de escenario debe admitir una comunicación sin estado y requiere un reemplazo de la versión temporal en memoria de la entidad empresarial que se crea o

00:00:52 edita. Esta versión temporal se conserva en tablas de base de datos separadas y se conoce como datos de borrador.

00:00:58 Los borradores están aislados en su propia persistencia y no influyen en la lógica empresarial existente hasta que se activan. Como principio rector, el borrador se puede considerar una memoria intermedia transaccional persistente.

00:01:10 En una aplicación sin soporte de borrador, no hay interacción con el back end hasta que se realice la grabación. Un inconveniente de este enfoque es que no hay feedback inmediato para el usuario final al introducir datos.

00:01:23 Si, por ejemplo, la conexión se rompe, es posible que se pierdan todos los datos introducidos. La activación de borrador ayuda en este escenario actualizando la instancia de borrador en el back end ya durante la interacción del usuario.

00:01:34 Esto significa que una solicitud correspondiente se envía al back end justo después de abandonar el campo actualizado. Esto permite el feedback temprano de determinaciones y validaciones, incluido el control dinámico de funciones.

00:01:47 La habilitación de borradores ayuda a evitar la pérdida de datos, admite el trabajo continuo y el cambio de dispositivo. Un desafío en la comunicación de estado es el manejo del control de concurrencia.

00:01:59 En las solicitudes que no son de borrador, la única opción es hacer uso del denominado bloqueo optimista utilizando ETag. En las aplicaciones habilitadas para borrador, se ha producido una interacción en curso con el back end, por lo que se pueden fijar bloqueos.

00:02:11 Estos bloqueos exclusivos no están vinculados a la sesión y tienen su propio tiempo de espera. Se denominan bloqueos duraderos.

00:02:18 Después del tiempo de espera, el bloqueo optimista se encarga de identificar los cambios realizados por otros usuarios o a través de diferentes canales. Los bloqueos vencidos se pueden volver a adquirir procesando el denominado currículum vítae en el back end.

00:02:30 El área en la que trabajaremos en esta unidad se resalta en el flujo de desarrollo. La definición de comportamiento permite el tratamiento de borradores.

00:02:37 Esto también se debe proyectar en la proyección y la proyección de comportamiento. Las tablas de borrador son tablas de base de datos regulares que se pueden generar o actualizar con el soporte de las herramientas de desarrollo ABAP.

00:02:47 No se resaltan en este resumen. Si ha seguido el enfoque de usar %tky en su aplicación, la activación de borrador no requiere ajustar el código de aplicación.

00:02:59 Pero en los casos en los que desea una diferenciación entre el tratamiento de borradores y no borradores, debe ajustar su implementación de comportamiento según corresponda.

00:03:06 Como resultado, habremos habilitado la gestión de borradores en la definición de comportamiento base, así como en la proyección de definición de comportamiento. También hemos utilizado las herramientas de desarrollo ABAP para generar las tablas de borrador para cada nodo de nuestro business object.

00:03:21 Ahora pasemos al sistema, habilitemos el tratamiento de borradores y probemos la aplicación. Abra la definición de comportamiento base de nuestro business object de viaje.

00:03:33 Para activar la gestión de borradores para un business object, añada "con borrador" en la sección de cabecera. Como el borrador es gestionado por el framework, necesitamos especificar una tabla de borrador para cada entidad.

00:04:05 Las tablas de borrador se pueden generar utilizando el soporte de herramientas. Utilice la corrección rápida para generar la tabla de borrador para la entidad Viajes.

00:04:16 Deje los valores predeterminados tal cual, haga clic en Siguiente. Asigne una solicitud de transporte y haga clic en Finalizar.

00:04:26 La tabla de borrador se genera en función del modelo definido. Grabe y active la tabla de borrador.

00:04:35 Tenga en cuenta que cada vez que modifique el modelo, puede volver a utilizar la corrección rápida para regenerar la definición de tabla de borrador. Utilice la corrección rápida para generar el calendario para la entidad de reserva.

00:05:00 Grabe y active la tabla de borrador. Verifique la definición del comportamiento.

00:05:11 Los errores han desaparecido, pero han aparecido mensajes de advertencia nuevos. Las advertencias para las asociaciones indican que están habilitadas para borrador implícitamente, ya que se trata de un business object habilitado para borrador.

00:05:26 Sustituya la definición de asociación para resolver la advertencia. Realice lo mismo para la asociación de viajes de la entidad de reserva.

00:05:44 Para identificar modificaciones en instancias activas en casos en los que el bloqueo duradero ha vencido, se requiere un campo ETag total. Cuando se va a activar una instancia de borrador, la IU de elementos Fiori llama la preparación de acción de determinación de borrador.

00:06:07 Esta llamada tiene lugar en un conjunto de modificaciones OData separado para permitirle grabar los mensajes de estado incluso si la activación falla debido a validaciones fallidas.

00:06:16 Para ejecutar nuestras validaciones durante la preparación, debemos asignarlas a la preparación de acción de determinación de borrador. Grabe y active la definición de comportamiento.

00:06:35 Abra la proyección de definición de comportamiento de nuestro business object de viaje. También es necesario proyectar el borrador de habilitación.

00:06:44 De lo contrario, la previsión se comportaría como si no se hubiera activado ningún borrador para el business object. Añadir borrador de utilización.

00:06:58 Al igual que el ajuste de asociación y la definición de comportamiento base, también necesitamos habilitar el borrador para ambas asociaciones en la proyección. Para OData V2, la IU de elementos Fiori actualmente no admite el tratamiento de ETags.

00:07:21 Por lo tanto, desactivamos el tratamiento de ETag en nuestra proyección. Grabe y active la proyección de definición de comportamiento.

00:07:38 Abra la implementación de comportamiento de la entidad Viajes. Navegue a la implementación get_Approvals.

00:08:00 La introducción del borrador ha añadido dos acciones predefinidas al business object, la preparación de la acción de determinación del borrador y la edición de la acción del borrador.

00:08:10 Desplácese hacia abajo y anule el comentario de las acciones de preparación y edición en ambos lugares. Ahora también se tienen en cuenta en el control de autorización.

00:08:27 Grabe y active la implementación de comportamiento. Lanzamiento, actualizé una IU de algunos elementos.

00:08:39 Lo primero que notamos es un nuevo campo de filtro que permite filtrar el estado de edición. Cree un viaje nuevo.

00:08:50 A diferencia del caso de utilización no borrador, la determinación sin modificar para fijar por defecto el estado ya se ha ejecutado. Al editar el campo, verá inmediatamente el indicador de guardado de borrador en la parte inferior.

00:09:13 La verificación de la tabla de borrador relacionada en el back end le muestra el valor introducido hasta el momento. Proporcione un valor incorrecto para el ID de agencia y corrija los valores para el ID de cliente, así como para la fecha de inicio y de fin.

00:09:49 Haga clic en Grabar. Recibe un mensaje correspondiente del back end, ya que este es el mensaje de estado, se almacena en la base de datos y pertenece al estado de la instancia de business

00:10:04 object. Puede salir de la pantalla, filtrar por su registro y verá los mensajes directamente al abrir la página de objeto sin volver a validar la instancia en el back end.

00:10:29 Corrija el valor y haga clic en Grabar. Ahora se realiza la cancelación de la inseguridad, que calcula el ID de viaje.

00:10:44 Sin necesidad de guardar, puede editar el encabezado del viaje junto con las entidades de reserva. Eso fue para la demostración del sistema.

00:11:11 Ahora estamos al final de esta unidad. Permítanme hacer una breve recapitulación.

00:11:14 En esta unidad, ha aprendido sobre las cualidades de una aplicación habilitada para borrador y cómo habilitar la gestión de borradores en nuestro business object de viajes.

00:11:23 Para más información sobre esta unidad, véase el apéndice de la presente presentación. Gracias por escucharle y verle en la siguiente unidad, donde haremos algunos problemas para solucionar los problemas de la aplicación.

Week 3 Unit 8

00:00:05 Hola y bienvenido al curso de openSAP, “Building Apps with the ABAP RESTful Application Programming Model”, semana tres, unidad ocho, "Resolución de problemas de su aplicación SAP Fiori".

00:00:15 Me llamo Volker Drees y soy un experto en productos para el modelo de programación de aplicaciones RESTful ABAP. En la unidad anterior, aprendió cómo añadir la gestión de borradores a nuestro business object de viaje.

00:00:25 Ahora veremos algunas de las diferentes opciones y herramientas de resolución de problemas disponibles. Probablemente, la herramienta más importante que necesita para solucionar problemas en su implementación de business object es el ABAP Debugger.

00:00:38 Poner un breakpoint en, por ejemplo, parte de su implementación de método le permite ejecutar el programa paso a paso y verificar el contenido de las variables relacionadas.

00:00:49 El análisis de errores en tiempo de ejecución se produce cada vez que se lanza un dump breve. Esto puede ocurrir, por ejemplo, si un método no devuelve el resultado esperado y el framework no puede continuar.

00:01:00 El log de errores de gateway muestra los errores que se han producido durante el procesamiento de llamadas OData. Está integrado en el lector de feed ADT.

00:01:08 La mayoría de los navegadores comunes admiten herramientas de desarrollador que normalmente se pueden llamar con el botón F12. Estas herramientas le ayudan a analizar las llamadas de red HTTP realizadas por su aplicación.

00:01:19 Comprender las llamadas y sus respuestas suele ayudar a identificar y resolver problemas. En el lado CDS, hay varias herramientas que le ayudan a definir y analizar su modelo de datos CDS.

00:01:31 La más común es, sin duda, la vista previa de datos que funciona también para las tablas de base de datos. Además de las capacidades de vista previa pura, ofrece muchas opciones para, por ejemplo, filtrar y clasificar sus datos.

00:01:42 Para consultas más complejas, contiene una consola SQL. En el contexto de CDS, también puede seguir asociaciones definidas en su vista CDS.

00:01:52 El log de diccionario entra en juego cada vez que falla una activación de una o varias vistas CDS y se necesita información más detallada sobre el error. La propagación de anotaciones y las vistas de anotaciones activas le ayudan a analizar las anotaciones utilizadas finalmente, así como a comprender de dónde

00:02:10 se pueden heredar. Ahora pasemos al sistema y veamos algunas de las herramientas de resolución de problemas.

00:02:16 Comencemos con el ABAP Debugger. Abra la implementación de comportamiento para la entidad de viaje.

00:02:33 Navegue a la implementación de acción aceptTravel y coloque un breakpoint en la sentencia de modificación. Ejecute la aplicación de elementos Fiori y ponga un registro en borrador.

00:03:06 Haga clic en el botón Aceptar viaje. El breakpoint es un acierto.

00:03:16 Cambie a la perspectiva de depuración. Haga doble clic en la tabla de importación de claves.

00:03:30 El depurador muestra el contenido de toda la tabla. Como se esperaba, el indicador %is_borrador se fija a medida que realizamos esta operación en un registro de borrador.

00:03:42 Tenga en cuenta la utilización de la clave de transacción porcentual, que representa la clave transaccional, incluido el indicador de borrador. Utilice el paso sobre, F6, para ejecutar la sentencia de modificación EML.

00:04:01 Pase el cursor por encima de la estructura "fallida" para validar que no se haya proporcionado ninguna clave fallida. Lo mismo es válido para la estructura notificada.

00:04:16 Ejecute la sentencia EML read. Haga doble clic en la tabla de viajes para ver los valores obtenidos.

00:04:28 Como se esperaba, el estado del viaje se fijó en A. Ejecute la sentencia value para rellenar el resultado. Pase el cursor por encima de la tabla de resultados y desglose.

00:04:45 Esto permite un análisis sencillo del contenido de tablas anidadas incluso complejas. Haga clic en retomar, F8.

00:05:10 Para rechazar el viaje, haga clic en Rechazar viaje en la aplicación. Active el registro de borrador haciendo clic en Grabar y vuelva a hacer clic en el botón Aceptar viaje.

00:05:29 Ahora podemos ver que la implementación del método se llama con la misma clave, pero con el indicador is_eprint fijado en 0, lo que significa que es un registro activo.

00:05:44 La clave de transacción contiene el valor correspondiente. Como puede ver, la implementación puede gestionar ambos tipos de registros: registros de borrador, así como registros activos.

00:05:57 Por eso recomendamos encarecidamente que se utilice la clave transaccional incluso en aplicaciones que no sean de borrador, ya que hace que un borrador sea mucho más fácil.

00:06:06 Elimine el breakpoint y haga clic en hoja de vida. Ponga algunas instancias de viaje en modo de edición.

00:06:38 En la implementación de comportamiento, abra el método get_functions y coloque un breakpoint en la primera sentencia. Realice una consulta en la IU de elementos Fiori.

00:07:01 La tabla de claves ahora contiene todos los registros activos de la primera página. La implementación utiliza la sentencia de lectura EML en masa para obtener el status de viaje para registros con una llamada.

00:07:35 La tabla de resultados se rellena con los datos correspondientes. Conserve el breakpoint y reanude la ejecución con F8.

00:07:47 Como puede ver, el breakpoint vuelve a aparecer, esta vez para todos los registros de borrador de la primera página. Elimine el breakpoint y reanude la ejecución.

00:08:05 Se llama get_functions para la consulta porque la IU solicita indicadores de control para activar o desactivar las acciones que hemos definido. Estos indicadores se fusionan con el control de autorización.

00:08:16 Tanto el control de características como el control de autorización se fusionan en los llamados consejos del consumidor. Ahora, vuelva a la perspectiva ABAP.

00:08:31 Es posible que ya haya notado la información del elemento ABAP que estoy utilizando, se puede llamar haciendo clic en F2 en, por ejemplo, una variable. La información del elemento también se puede abrir como una vista permanente haciendo clic en Ventana, Mostrar vista, Otro y, a continuación, en

00:09:05 Información de elemento ABAP. Asegúrese de que el enlace con la opción de selección está habilitado y que la vista se actualiza automáticamente al hacer clic en un elemento.

00:09:23 Esto es muy útil ya que le muestra rápidamente la declaración del elemento relacionado. El Explorador de relaciones es otra herramienta útil para visualizar la estructura de artefactos de desarrollo.

00:09:39 Se puede utilizar para definiciones de comportamiento y para proyecciones, y muestra las entidades del objeto empresarial, así como acciones, determinaciones y validaciones.

00:09:52 El Explorador de relaciones se puede abrir de forma similar a la información del elemento ABAP, mediante Ventana, Mostrar vista, Otro y, a continuación, Explorador de relaciones.

00:10:06 De nuevo, asegúrese de habilitar el enlace con el editor para que la vista se actualice con el contenido del editor. Abra la definición de comportamiento base, haga clic con el botón derecho en, por ejemplo, la determinación setInicialStatus, que le permite navegar

00:10:38 directamente a la declaración y la definición de comportamiento, a la declaración del método en la clase de implementación, así como a la implementación de método real.

00:10:58 Pero el Explorador de relaciones no solo es para business objects, admite muchos tipos de objeto, por ejemplo, clases ABAP. Seleccione la clase de implementación de comportamiento y cambie a Objetos utilizados.

00:11:14 Como puede ver, se muestran todos los objetos utilizados de la clase ABAP. La mayoría de los navegadores comunes admiten herramientas de desarrollador que normalmente se pueden llamar con el botón F12.

00:11:35 Los utilizamos para supervisar las llamadas HTTP enviadas por el cliente de elementos Fiori al back end. Las herramientas de desarrollador le permiten ver toda la solicitud HTTP y la respuesta con la cabecera y la información del cuerpo.

00:11:48 Inicie las herramientas de desarrollador desde la aplicación de elementos Fiori y vuelva a cargar la página. Si desea filtrar por llamadas OData, introduzca el filtro "odata".

00:12:15 En nuestro caso, la primera entrada de la lista es el archivo de metadatos. El código HTTP 304 significa que no se ha modificado, por lo que el fichero se ha tomado del caché.

00:12:29 Hacer clic en la entrada le permite ver la información de la solicitud, así como la respuesta. La ficha Vista previa le permite profundizar en el contenido.

00:12:49 Esto ayuda a comprender los metadatos enviados por el back end. La segunda entrada es el fichero de anotaciones, que proporciona las anotaciones basadas en vocabulario para nuestro servicio.

00:13:11 Haga clic en el botón Ir para realizar una consulta. Verá que se ha iniciado una llamada $batch correspondiente.

00:13:29 La llamada Get dentro de $batch incluye la información de paginación reflejada como $skip y $top. El orden de clasificación se especifica con la opción de consulta $orderby.

00:13:40 Además, encontrará un filtro y una sentencia SELECT. $inlinecount proporciona el número de registros que coinciden con los criterios de filtro.

00:13:51 La apertura de la página de objeto de un viaje desencadena otras llamadas $batch. Encontrará una solicitud para la cabecera del viaje, así como una solicitud para obtener la lista de reservas.

00:14:08 Haga clic en una acción, ya sea aceptar o rechazar mostrará los detalles de la solicitud. La respuesta contiene el registro de viaje con el valor actualizado.

00:14:19 Esto se debe a que hemos definido $auto-como respuesta en la definición de comportamiento y la implementación de comportamiento proporciona los datos correspondientes.

00:14:29 La activación de un borrador muestra la orquestación, que primero consiste en una llamada de preparación; esto desencadena la acción de determinación de borrador preparar con sus determinaciones y validaciones asociadas.

00:14:53 La acción de preparación puede, por definición, no fallar ya que se supone que debe verificar el registro de borrador y recopilar mensajes. En el mismo $batch, pero en un conjunto de modificaciones diferente, encontrará la acción de activar borrador.

00:15:11 La tercera llamada es una llamada Get que obtiene la instancia activa. En las herramientas de desarrollo ABAP, abra la vista de proyección de viajes.

00:15:33 Pulse F8 para iniciar la presentación preliminar, que muestra las primeras 100 filas de la vista CDS. Hacer clic con el botón derecho en una entrada le permite, por ejemplo, seguir una asociación.

00:15:52 Seleccione Inscripción y verá la lista de reservas asociadas. Puede seguir desglosando y navegar hasta el cliente, el país, etc.

00:16:14 Los breadcrumbs de la parte superior le permiten retroceder y elegir una asociación diferente. Vuelva al nodo Viajes.

00:16:32 Haga clic con el botón derecho en AgencyID y seleccione valores distintos. Esto muestra los valores distintos y su distribución.

00:16:51 Haga clic con el botón derecho en un ID de agencia y seleccione "Filtro rápido". Esto añade un filtro para AgencyID con el valor seleccionado.

00:17:08 "Copiar todas las filas como sentencia de valor ABAP" le permite añadir fácilmente un constructor de valor en su código ABAP con los registros seleccionados.

00:17:56 La consola SQL le proporciona total flexibilidad a la hora de definir la sentencia SQL. Anotaciones activas : haga clic con el botón derecho en el proyecto de viaje en la vista CDS y seleccione Anotaciones activas.

00:18:28 Al hacer doble clic en la ficha, se expande a pantalla completa. Esta vista muestra las anotaciones, válidas para su vista CDS y de dónde se originan.

00:18:42 Como ejemplo, puede ver las acciones definidas para la lista, así como para la página de objeto que están asociadas con el estado de viaje. Esta vista es muy útil, especialmente en escenarios complejos en los que las anotaciones se pueden heredar en varias vistas CDS.

00:19:04 Ése era el caso de la demostración del sistema, ahora estamos al final de esta unidad. Permítanme hacer una breve recapitulación.

00:19:10 En esta unidad, ha aprendido sobre la resolución de problemas de su implementación ABAP, la verificación de las llamadas de red de su aplicación y cómo las herramientas de desarrollo ABAP le ayudan a analizar y definir su modelo de datos CDS.

00:19:24 Para más información sobre esta unidad, véase el apéndice de la presente presentación. Ahora estamos al final de la semana tres del curso de openSAP, "Creación de aplicaciones con el modelo de programación de aplicaciones RESTful ABAP".

00:19:37 Gracias por escucharle y verle la semana que viene, donde creamos una aplicación basada en el código existente, haciendo uso del escenario no gestionado.

www.sap.com/contactsap

THE BEST RUN SAP®