openSAP

Building Apps with the ABAP RESTful Application Programming Model

Language subtitles (translated by machines)

Week 4 Unit 1

00:00:05 Hola, y bienvenido al curso de openSAP Creación de aplicaciones con el modelo de programación de aplicaciones RESTful ABAP. Semana cuatro, unidad uno, El escenario empresarial.

00:00:19 Me llamo Andre Fischer y soy gestor de productos de SAP Gateway y SAP Cloud Platform, entorno ABAP. Esta es la primera unidad de la semana cuatro.

00:00:34 Esta vez estamos buscando un modelo de negocio en el que queremos aprovechar la lógica de negocio existente. Utilizaremos las tablas existentes /DMO/TRAVEL y /DMO/BOOKING.

00:00:53 Para crear, actualizar y borrar entradas de tabla en nuestra nueva implementación transaccional, reutilizaremos la lógica empresarial existente que se ha implementado en módulos de funciones similares a BAPI que se ocupan de la grabación en memoria intermedia y la validación de los datos introducidos por el usuario final.

00:01:15 Puesto que tenemos que ocuparnos de todo esto, estamos hablando de un escenario no gestionado. No gestionado desde el punto de vista del marco RAP.

00:01:30 La aplicación resultante será similar a la aplicación que hemos creado sobre nuestra implementación gestionada en las semanas dos y tres. Desde una perspectiva de alto nivel, el flujo de desarrollo parece similar a la implementación de un escenario gestionado.

00:01:52 La diferencia se hace más visible cuando observamos con más detalle las definiciones de comportamiento y las implementaciones de comportamiento. La implementación de un escenario no gestionado requiere pasos de implementación de métodos que se ejecutan en la fase de interacción o en la secuencia de grabación

00:02:20 cuando el objeto persiste en el nivel de base de datos. Veamos con más detalle qué es lo que realmente hay detrás del adaptador para el código de aplicación existente.

00:02:31 Estos adaptadores se muestran bastante pequeños en esta diapositiva. Cuando examinemos con más detenimiento la implementación de comportamiento en nuestro escenario no gestionado, descubriremos que aquí es donde tiene lugar la

00:02:49 asignación entre la lógica heredada y el modelo de datos en el objeto empresarial RAP. En realidad, se necesita implementar un código para llamar la lógica empresarial existente en nuestra implementación de comportamiento.

00:03:04 En el fragmento de código del método de creación, verá que los datos que recibimos del framework en la tabla interna, entidades, se asignan a las estructuras de datos legacy_entity_in utilizadas por la lógica empresarial existente de nuestro módulo de funciones /DMO/FLIGHT_TRAVEL_CREATE.

00:03:30 En esta unidad, aprendió cómo el escenario de implementación no gestionado admite los llamados escenarios de campo marrón en los que desea aprovechar el código existente.

00:03:42 Y aprendió cómo este tipo de implementación difiere de los escenarios de las últimas unidades. Ahora estamos al final de la unidad.

00:03:53 Gracias por escuchar y volver a verte en la siguiente unidad. Allí, hablaremos sobre cómo crear el modelo de datos CDS.

Week 4 Unit 2

00:00:05 Hola, y bienvenido al curso de openSAP Creación de aplicaciones con el modelo de programación de aplicaciones RESTful de ABAP. Semana cuatro, unidad dos, creación del modelo de datos CDS.

00:00:18 Me llamo Andre Fischer y soy gestor de productos de SAP Cloud Platform, entorno ABAP y SAP Gateway. Esta es la segunda unidad de la semana cuatro.

00:00:32 El modelo de datos CDS se crea a partir de tablas de base de datos existentes y más adelante también utilizaremos la lógica empresarial existente para crear y actualizar datos.

00:00:46 En esta unidad, empezaremos a modelar nuestro business object. Para ello, crearemos dos vistas de interfaz CDS.

00:00:53 Un nodo raíz para los datos de viaje y un nodo subordinado para los datos de reserva. Comencemos con la demostración.

00:01:04 Al final de esta demostración, habremos creado el árbol de composición de business object para las entidades Viajes y Reserva y utilizaremos la función de vista previa en ADT para verificar los datos que existen en las tablas.

00:01:21 El tema de esta semana es tratar el código existente. Así que comenzamos con una mirada a la lógica de negocio heredada que queremos aprovechar para nuestra implementación de objetos de negocio no

00:01:34 gestionados. Para ello, investigamos el subpaquete /DMO/FLIGHT_LEGACY, que se encuentra en el paquete /DMO/FLIGHT.

00:01:49 Los datos se leen de las tablas de base de datos existentes /DMO/TRAVEL y /DMO/BOOKING. Además, utilizaremos módulos de funciones, como /DMO/FLIGHT_TRAVEL_CREATE o /DMO/FLIGHT_TRAVEL_UPDATE o /DMO/FLIGHT_TRAVEL_DELETE, que se utilizan

00:02:34 para crear, actualizar y borrar los datos en una memoria intermedia. También hemos leído el módulo de funciones que lee datos de la memoria intermedia, en lugar del nivel de base de datos, y esto resulta útil al

00:02:59 implementar la lógica empresarial necesaria cuando la lógica empresarial se llama a través de EML, el idioma de manipulación de entidad, en lugar de una IU de SAP Fiori.

00:03:15 Y, por último, existe un módulo de funciones, /DMO/FLIGHT_TRAVEL_SAVE, que se utiliza para volver a escribir los datos de la memoria intermedia en el nivel de base de datos.

00:03:33 Comencemos con la creación de nuestra vista de interfaz raíz para datos de viajes. Se supone que antes ha creado un paquete adecuado, por ejemplo, ZRAP_TRAVEL_U_ y, a continuación, el número de grupo, es decir, este, o

00:03:52 está reutilizando el paquete que ha utilizado en la semana dos y la semana tres. Para crear una definición de datos de una vista CDS, hay un nuevo asistente disponible.

00:04:05 Para aprovechar este nuevo asistente, haga lo siguiente. En su proyecto en la nube ABAP, o en su otro proyecto si está trabajando en un sistema local, seleccione la tabla /DMO/TRAVEL en el Explorador de

00:04:24 proyectos, que se encuentra en el paquete /DMO/FLIGHT_LEGACY. Haga clic con el botón derecho y seleccione Nueva definición de datos.

00:04:38 Esto iniciará un nuevo asistente en ADT que le permitirá crear cómodamente una vista CDS basada en otra tabla u otra vista CDS. Aquí tenemos que modificar el nombre del paquete, por lo que nuestro paquete ZRAP_TRAVEL_U_ su número de grupo.

00:05:02 Y tenemos que introducir el nombre de nuestra vista de interfaz, de modo que sería ZI_RAP_TRAVEL_U, para no gestionado, subrayado y luego su número de grupo.

00:05:19 Y una descripción, como los datos de viaje. Y luego podemos presionar Siguiente.

00:05:29 Seleccionamos una orden de transporte. Y ahora estamos en el nuevo diálogo de definición de datos y aquí seleccionamos la plantilla Definir vista, entidad y pulsamos Finalizar.

00:05:52 Por lo tanto, vemos que se ha creado una nueva vista CDS y la fuente de datos es la tabla de viajes, y todos los campos de la tabla de viajes se han insertado entre llaves.

00:06:09 Así que ahora empezamos añadiendo asociaciones. Y para obtener buenos nombres de campo, veremos cómo se añaden alias, como TravelID o AgencyID y como CustomerID.

00:07:12 Y entonces verá que parte de estas advertencias o flechas desaparecerán. Está previsto ampliar este asistente de modo que también los alias se generen automáticamente en función de los nombres de campo ABAP, pero para

00:07:36 acelerar el desarrollo, hemos preparado un fragmento de código para el código fuente DDL de la entidad raíz y también para la entidad inferior que podemos introducir aquí.

00:07:55 Y vemos que este código contiene anotaciones para los campos que contienen datos de moneda. Esto muestra el marco donde encontrar el campo que contiene el código de moneda.

00:08:15 Además, hemos añadido la @semántica para especificar qué campos contienen datos para createdby, createdat, lastchangedby y lastchangedat. Por lo tanto, esos ámbitos administrativos que conocemos también por el enfoque gestionado.

00:08:38 Por lo tanto, hacemos doble clic en la parte superior para obtener, de nuevo, el Explorador de proyectos en la parte izquierda. Y ahora hacemos lo mismo con las tablas de reservas.

00:08:50 Por lo tanto, haga clic con el botón derecho en la tabla /DMO/BOOKING y seleccione Nueva definición de datos. De nuevo, introducimos el nombre del paquete.

00:09:11 Y un nombre para la vista de interfaz, ZI_RAP_Booking_U_ y, a continuación, el número de grupo y una descripción significativa como los datos de reserva.

00:09:28 Y podemos presionar Siguiente. Haga la orden de transporte, pulse Siguiente.

00:09:35 De nuevo, utilizamos la plantilla Definir entidad de vista y pulsamos Finalizar. De nuevo, utilizamos la codificación preparada que puede utilizar mediante cortar y pegar para añadir varias asociaciones.

00:10:01 Y para el código fuente DDL, también hemos preparado alguna codificación. Y aquí vemos el resultado.

00:10:27 Ahora podemos activar ambas vistas CDS. Pulse Activar.

00:11:02 Ahora podemos verificar la vista previa de datos, de modo que seleccionamos la vista CDS y pulsamos F8. Y vemos todos los datos en la reserva y en la vista CDS de viaje que se lee de las tablas de base de datos subyacentes.

00:11:28 Ahora que hemos activado correctamente ambas vistas de interfaz CDS, añadimos la asociación de composición que apunta desde la entidad Viajes a la entidad Reserva y la asociación que remite de la entidad secundaria Reserva a la entidad superior Viajes, que en este caso también es la

00:11:53 entidad raíz. Por lo tanto, en la entidad raíz, agregaremos la asociación del tipo composición que apunta a la entidad Reserva y aquí obtendremos un mensaje

00:12:08 de error que indica que la entidad Reserva aún no tiene una asociación a un superior. Por lo tanto, tendremos que añadir esto también.

00:12:21 Por lo tanto, también tenemos que publicar la asociación. Abrimos la vista de interfaz para los datos de reserva y aquí agregamos una asociación que apunta a la entidad principal.

00:13:00 Y esto también tiene que ser publicado. Ahora podemos utilizar, de nuevo, la activación en masa y activar ambas vistas CDS.

00:13:36 Seleccione Activar. Y vemos que los objetos se están activando, pero también vemos que hay una advertencia de que no se encuentra ninguna entidad raíz en la estructura

00:14:01 de BO, y si miramos detenidamente vemos que me olvidé de añadir la palabra clave "raíz". Y sólo entonces, si vuelvo a activar ambas cosas, desaparecerá.

00:14:29 En esta unidad, ha aprendido qué es un árbol de composición de business object y cómo lo define. Ahora estamos al final de esta unidad.

00:14:38 Gracias por escuchar y verlo en la siguiente unidad. Allí hablaremos sobre la definición e implementación del comportamiento del business object.

Week 4 Unit 3

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

00:00:20 object. Me llamo Andre Fischer y soy gestor de productos de SAP Gateway y SAP Cloud Platform, entorno ABAP.

00:00:29 Esta es la tercera unidad de la semana cuatro. En esta unidad, crearemos la definición de comportamiento para nuestro business object, que es un business object

00:00:46 no gestionado. Y en la definición de comportamiento, tenemos que especificar que no se gestiona.

00:00:55 Y en la implementación de comportamiento, a su vez, tenemos que implementar varios métodos, crear, actualizar y borrar, así como un método de lectura que leerá de la memoria intermedia y un método de bloqueo que aprovechará

00:01:11 los mecanismos de bloqueo de nuestra lógica empresarial heredada. Si miramos el flujo de desarrollo, estamos en las secciones resaltadas en amarillo que agregan e

00:01:24 implementan el comportamiento de nuestro business object. Comencemos con una demostración.

00:01:31 Al final de la demostración, habremos creado y definido la definición de comportamiento de nuestro business object para las entidades de viaje y reserva.

00:01:43 Además, implementaremos el comportamiento de business object en la clase de implementación de comportamiento. Además, crearemos una clase de test y usaremos el lenguaje de manipulación de entidades para probar nuestro

00:02:00 business object justo después de haber implementado su funcionalidad. En esta unidad, agregaremos comportamiento a nuestro business object ZI_RAP_Travel_U_ group number.

00:02:16 Al hacer clic con el botón derecho en la vista raíz de nuestro business object, se abre el diálogo Nueva definición de comportamiento.

00:02:34 Aquí debemos cambiar el tipo de implementación de gestionado a no gestionado. El nombre de la definición de comportamiento no se debe modificar y no se puede modificar, por lo que es el mismo

00:02:52 que la vista de interfaz raíz. La descripción también se puede dejar sin cambios.

00:03:00 Pulsamos Siguiente, seleccionamos una orden de transporte y Finalizamos. El asistente generará un archivo BDEF con las opciones predeterminadas que debemos ajustar.

00:03:16 En lugar de utilizar una clase de implementación de comportamiento para toda la definición de comportamiento, utilizaremos clases de implementación de comportamiento separadas para cada entidad de nuestro business object.

00:03:31 Por lo tanto, hacemos la declaración. La implementación en la clase zbp_i_rap_trav_u_1234 es unívoca.

00:03:43 Así que justo después de definir el comportamiento para la declaración de nuestra entidad de Travel. Agregamos un alias, Viajes.

00:04:04 Eliminamos la codificación sugerida para la numeración tardía porque no queremos utilizar la numeración tardía y descomentamos el maestro de bloqueos de codificación.

00:04:20 También anulamos el comentario para el maestro de etag y el nombre de campo, podemos seleccionar completando el código, a través de L, Lastchangeedat.

00:04:36 Dejamos las parametrizaciones para las operaciones estándar, creamos, actualizamos y eliminamos, así como el código que define que la asociación _Booking es una opción de creación.

00:04:56 De este modo, solo puede crear entidades de reserva a través de la entidad principal Viajes. Para la reserva, también agregamos un alias.

00:05:14 Y añadimos una declaración para la clase de implementación para la entidad Reserva. De nuevo, eliminamos la sentencia para la numeración tardía y anulamos el comentario del bloqueo dependiente

00:05:37 de , por lo que depende de la asociación _Travel. Y el maestro de etag también se reemplaza, de modo que también depende de _Travel.

00:06:08 En este punto, recibiremos un mensaje de error, no se ha definido ningún comportamiento para la asociación _Travel de la entidad ZI_RAP_Booking.

00:06:19 Por lo tanto, ahora debemos añadir una declaración para la asociación entre llaves. Añadimos ahora el comportamiento de aquellos campos que deberían ser obligatorios o de solo lectura.

00:06:58 Y hacemos lo mismo con la entidad de reserva. También eliminamos el comportamiento creado para la entidad Reserva porque queremos garantizar que las

00:07:19 entidades de reserva solo se creen mediante asociación, es decir, a través de la entidad superior Viajes. Como paso final, podemos especificar la asignación entre los nombres de campo de vista CDS y los nombres de campo

00:07:35 que utiliza nuestra lógica empresarial existente. Esto significa que los módulos de funciones que utilizamos para crear, actualizar y borrar datos de

00:07:51 vuelo. O sustituimos las etiquetas hash por el número de grupo y hacemos lo mismo para la reserva.

00:08:18 Entonces, ¿por qué lo hacemos? Al acceder al código heredado, el desarrollador normalmente tendría que utilizar "correspondiente con

00:08:27 asignación" en muchos lugares para asignar la entrada de tipos derivados, como la tabla de tipos para crear o la tabla de tipos para la importación de acciones, a los tipos existentes y viceversa.

00:08:40 Para asignar resultados existentes a tipos derivados de salida, como tabla de tipo para resultado de acción, tabla de tipo para resultado de lectura o fallida.

00:08:52 En los escenarios existentes que se basan en BAPIs, se utilizan las llamadas estructuras de control o estructuras x.

00:09:00 Estas estructuras tienen el mismo nombre de campo, además de que la única excepción es el campo clave, como la estructura normal, pero tienen el tipo char1 para indicar a la BAPI cuál de los campos transferidos contiene un

00:09:20 valor, mediante verdadero y falso. Este tipo tiene la misma función que la estructura %CONTROL en los tipos derivados de la entrada, que

00:09:31 utiliza el framework RAP para indicar a qué campos de la estructura principal se accede mediante una operación, actualización, lectura, etc.

00:09:42 Pero dado que el tipo de datos en la estructura %CONTROL del framework RAP difiere del tipo de datos utilizado por las BAPI, también debemos realizar una asignación aquí.

00:09:53 La solución ahora es introducir una asignación declarativa central dentro de la definición de comportamiento en lugar de muchas sentencias individuales correspondientes en el código.

00:10:04 Esto mejora la capacidad de mantenimiento del código fuente de la aplicación. Por lo tanto, lo que queda ahora es crear dos estructuras para ambas entidades, para la estructura de control.

00:10:19 Por lo tanto, la asignación, ya hemos definido. Por lo tanto, vamos a crear las dos estructuras que faltan.

00:10:45 Estructura de control y una descripción. Primero para viajes.

00:10:59 Y la reemplazamos por una codificación preparada. Podemos activarlo.

00:11:13 Y creamos una segunda estructura para la reserva. Aquí también tenemos los mismos nombres de campo, pero el tipo xsdbooleano.

00:11:47 Y cuando verificamos la codificación de nuestra definición de comportamiento, vemos que los errores han desaparecido porque hemos creado esas estructuras de control y lo único que queda es que tenemos que crear las

00:12:02 clases de implementación de comportamiento, que haremos en un segundo en el siguiente paso. Como pequeña recapitulación, ¿qué hemos conseguido?

00:12:12 Por lo tanto, hemos creado un comportamiento de implementación no gestionado, una definición de comportamiento para el tipo de implementación no gestionado.

00:12:23 Hemos especificado los nombres para las clases de implementación de comportamiento, por separado para cada entidad.

00:12:31 Hemos definido las operaciones estándar, crear, actualizar, borrar, que deberían implementarse posteriormente en la clase de implementación de comportamiento.

00:12:42 La asociación habilitada para crear que apunta de Viajes a Reserva. Hemos especificado campos, que son obligatorios y de solo lectura, que veremos más adelante en la IU.

00:12:56 Y hemos definido una asignación central para la lógica empresarial existente, incluida una estructura de control.

00:13:07 Para crear las clases, podemos utilizar una corrección rápida, podemos seleccionar el nombre de la clase y pulsar CTRL + 1.

00:13:17 Esto crearía la clase de implementación de comportamiento, pero dado que la codificación para el beta aún no está activa, primero tenemos que activarla.

00:13:30 Y luego podemos pulsar, nuevamente, CTRL + 1 y crear la clase de implementación de comportamiento para el viaje. Aquí puede ver desde la firma, un método de creación que se llamará al crear una entidad y bórrelo si se va a

00:14:03 borrar una entidad. Empezaremos por implementar el método Create.

00:14:15 Y utilice la codificación preparada. ¿Y qué vemos aquí?

00:14:24 De modo que obtenemos datos de entrada de esas entidades para que los datos en el payload que se utilizan para crear nuevos elementos de viaje se asignen utilizando esta asignación central y la estructura de control, de

00:14:45 modo que los campos que forman parte del payload, a esta estructura legacy_entity_in , que es de tipo estructura, que acepta nuestra API heredada, /DMO/FLIGHT_TRAVEL_CREATE.

00:15:02 Y si la creación se ha realizado con éxito, no existe ningún mensaje típico con BAPIs. A continuación, añadimos el CDS y el ID de viaje a esta estructura de viajes dinámica.

00:15:21 Y si algo sale mal, la estructura fallida y la notificada se rellenarán para que el error se pueda visualizar en la IU.

00:15:34 A continuación, tenemos que navegar desde la clase de programa de control local, que es responsable de actualizar los datos en la memoria intermedia hasta la clase de ahorrador local.

00:15:49 Aquí implementamos el método de grabación en el que llamaríamos a la API heredada, que persiste los datos de la memoria intermedia, que conservan esos módulos de funciones hasta la base de datos.

00:16:06 Y estamos activando nuestros cambios. Entonces, ¿cómo vamos a probar si nuestra implementación funciona?

00:16:15 Por lo tanto, podríamos proceder como hicimos la última vez, publicar el servicio y luego utilizar la funcionalidad de vista previa.

00:16:24 Aquí voy a mostrar una forma diferente. Primero podemos empezar a implementar una prueba, que siempre es una buena práctica, para que probemos nuestra

00:16:37 implementación a través de una prueba de unidad. Así que navegamos a las clases de prueba de pestaña y allí insertamos los datos preparados.

00:16:52 Por lo tanto, aquí tenemos que sustituir todos los sucesos de los cuatro hashes por nuestro número de grupo. ¿Y qué hace esta codificación?

00:17:14 Por lo tanto, estamos utilizando el marco de prueba doble CDS y creamos un viaje, un solo viaje y, para ello, estamos utilizando EML, por lo que el lenguaje de manipulación de entidades, por lo tanto, modifica las

00:17:32 entidades, Viajes, y luego estamos comprobando que las estructuras de devolución para el fallido y el notificado que indicarían un error están vacías.

00:17:44 Si ese es el caso, enviamos los datos a las tablas de framework de test dobles y, a continuación, podemos volver a seleccionar los datos de la vista de interfaz CDS.

00:18:01 Y podemos verificar si los datos se han creado correctamente. Y aquí vemos la configuración de la clase, lo que crea el framework de test doble para las dos vistas de interfaz

00:18:15 para viajes y reservas. Ahora hacemos clic con el botón derecho en la clase y seleccionamos Ejecutar como, Test funcional, y vemos que

00:18:29 la prueba fue exitosa. Lo que queda ahora es que tenemos que implementar todos los demás métodos, por ejemplo, el método de eliminación.

00:18:40 De este modo, el método de borrado, solo obtiene claves y, a continuación, llamamos al módulo de funciones que borra un viaje.

00:18:52 O podemos actualizar los datos. Una vez más, obtenemos datos como entidades.

00:19:01 Los datos vuelven a asignarse a nuestra asignación central, según nuestra asignación central, y aquí también utilizamos esta asignación para la estructura de control.

00:19:15 A continuación, podemos utilizar el módulo de funciones de actualización de viajes. También hemos implementado el método de bloqueo, que realizaría un bloqueo para evitar, cuando se realiza una

00:19:47 actualización, que se produzcan dos actualizaciones al mismo tiempo. Y tenemos un método de lectura que leería datos de la memoria intermedia en lugar de de la base de datos.

00:20:05 Por lo tanto, esto es necesario, por ejemplo, al realizar actualizaciones, porque estamos utilizando ETags aquí. Y tenemos un método de creación por asociación de reservas, este método se ejecuta cuando se crea una

00:20:27 reserva a través de la entidad superior Travel. Y por último, pero no por ello menos importante, existe una codificación para la lectura mediante la reserva de

00:20:39 asociación, similar al método de lectura para la entidad Travel. De este modo se leerían los datos de inscripción de la memoria intermedia.

00:20:48 En esta unidad, ha aprendido a definir el comportamiento de un business object y cómo implementarlo. También aprendió cómo puede implementar una clase de prueba para probar su implementación de comportamiento.

00:21:07 Justo después de haberlo desarrollado. Ahora estamos al final de esta unidad.

00:21:13 Gracias por escuchar y verle en la siguiente unidad. Aquí hablaremos sobre la creación de la proyección de business object.

Week 4 Unit 4

00:00:05 Hola y bienvenido al curso de openSAP "Creación de aplicaciones con el modelo de programación de aplicaciones RESTful ABAP", semana cuatro, unidad cuatro, "Creación de la proyección de business

00:00:17 object". Me llamo Andre Fischer y soy gestor de productos para SAP Cloud Platform, entorno ABAP y SAP Gateway.

00:00:25 Esta es la cuarta unidad de la semana cuatro. En esta unidad, crearemos las proyecciones del modelo de datos y la definición de comportamiento de

00:00:38 nuestro business object. Puede crear varias proyecciones del mismo business object, de modo que pueda reutilizar la

00:00:46 implementación de sus business objects para varias aplicaciones en las que algunas de ellas solo ofrezcan un subconjunto de funcionalidades.

00:00:56 Si miramos el flujo del dev, estamos en la capa de proyección resaltada en amarillo. Comencemos con la demostración.

00:01:07 Al final de esta demostración, habremos creado y definido la proyección del modelo de datos de business object para las entidades de viaje y reserva, y habrá definido también la proyección de

00:01:20 comportamiento de business object para ambas entidades. La capa de proyección es la primera capa en el flujo de desarrollo del modelo de programación RESTful

00:01:31 ABAP que es específico de servicio. La capa de proyección contiene un ajuste preciso específico de servicio que no pertenece a la capa de

00:01:41 modelo de datos general, por ejemplo, anotaciones de IU, ayudas para entradas, cálculos o valores predeterminados.

00:01:49 La capa de proyección también permite exponer un business object en un servicio OData para una IU Fiori y para una API web estable.

00:02:00 Dado que las API web no necesitan anotaciones de IU, no se implementan en la proyección correspondiente. Tendremos una mirada a las API web en la semana cinco.

00:02:14 Con las proyecciones, también puede gestionar un acceso basado en roles. Mientras que un servicio permitiría crear, actualizar y borrar datos, es posible que otra

00:02:26 proyección solo permita aprobar o rechazar una solicitud. Comencemos añadiendo una capa de proyección en nuestras vistas de interfaz CDS.

00:02:36 Podemos hacer clic con el botón derecho en las vistas de interfaz CDS para viajes y reservas, y utilizar el nuevo asistente para crear vistas de proyección que tengan nuestras vistas CDS de

00:03:01 interfaz como fuente de datos. Así que presionamos Siguiente.

00:03:09 Seleccione una orden de transporte. Aquí seleccionamos la plantilla Definir vista de proyección.

00:03:26 Y pulse Finalizar. Aquí tenemos que añadir la palabra clave raíz.

00:03:40 Y empezamos a añadir varias anotaciones. Comenzamos a hacer que la vista CDS pueda buscarse.

00:03:56 Y establezca el ID de agencia. Y el CustomerID los elementos de búsqueda predeterminados.

00:04:07 A continuación, añadimos anotaciones de ayuda para entradas para AgencyID y CustomerID, así como para CurrencyCode.

00:04:29 También tenemos que actualizar información adicional para la entidad de reserva de que esto se basa en una composición, apunta a un hijo de composición.

00:04:45 Recibimos un mensaje de error, ya que la vista de proyección para la reserva aún no existe. Ahora creamos una vista de proyección para la entidad de reserva, para la entidad secundaria.

00:05:17 Pulse Siguiente a la orden de transporte y, de nuevo, seleccionamos el modelo Definir vista de proyección.

00:05:32 También hacemos que esta vista CDS se pueda buscar. Y añada anotaciones para el elemento de búsqueda predeterminado para TravelID e BookingID.

00:05:47 A continuación, agregamos definiciones de ayuda para entradas para varios campos, como CustomerID. El ID de transportista, que se llama AirlineID en la vista CDS transportista.

00:06:08 Tenga en cuenta que el nombre del elemento es diferente. Para el ID de conexión, tenemos vinculaciones adicionales para que, si seleccionamos una

00:06:23 determinada conexión, los otros campos de la IU también se actualicen automáticamente. Finalmente, añadimos una ayuda para búsqueda para CurrencyCode y tenemos que añadir información a la

00:06:40 asociación de viajes que señala, que se redirige a una entidad superior. Ahora podemos activar tanto la vista del proyecto para viajes como para los datos de reserva, y

00:07:04 seleccionamos esta opción, activar todo. Parte de la capa de proyección, en caso de que desee implementar el servicio para el consumo basado en

00:07:19 IU, son las extensiones de metadatos. Las ampliaciones de metadatos contienen anotaciones específicas de IU y se han inventado para separar

00:07:28 anotaciones basadas en IU de las vistas de proyección. Por lo tanto, hacemos clic con el botón derecho en nuestra proyección recién creada para viajes y

00:07:42 seleccionamos Nueva extensión de metadatos. El nombre, seleccionamos el mismo nombre que la vista de proyección.

00:07:55 Siguiente. A continuación, seleccionamos Anotar vista.

00:08:08 Como capa de metadatos, elegimos #CORE. Agregamos anotaciones para la cabecera y para la presentaciónVariant, que define el orden de

00:08:23 clasificación predeterminado. Y agregamos varias anotaciones, por lo que ocultaremos los campos administrativos y agregaremos

00:08:34 anotación de identificación de lineItem para que los campos se muestren en la lista y en la página de objeto.

00:08:43 De modo que todavía vemos un error que indica que falta la anotación "Metadata.AllowExtensions" en la vista de proyección que se va a anotar.

00:08:54 Aquí simplemente podemos pulsar Ctrl+1. Y utilice la solución rápida de que esta anotación se añade ahora a la vista base.

00:09:09 Lo activamos de nuevo. Puede volver a nuestra vista de extensión de metadatos y activarla.

00:09:22 Ahora podemos hacer lo mismo para la vista de proyección de reserva, así que haga clic con el botón derecho en ella y seleccione Nueva extensión de metadatos.

00:09:32 Aquí también elegimos el mismo nombre. Pulse Siguiente y Finalizar.

00:09:47 Pulsando Finalizar, seleccionamos la misma plantilla que en el último paso. Seleccionamos la anotación #CORE.

00:10:08 Añada anotaciones para la cabecera. Y añada invitaciones de nuevo para una faceta y ocultaremos, por ejemplo, el campo clave.

00:10:22 De nuevo, recibiremos un mensaje de error que indica que la vista de previsión para la reserva no permite que se amplíe con ampliaciones de metadatos.

00:10:36 De nuevo, seleccionamos la corrección rápida, activamos la vista de extensión de metadatos y ahora podemos activar la extensión de metadatos.

00:10:52 El último objeto que tenemos que crear en la capa de proyección es una proyección para nuestra definición de comportamiento.

00:11:00 Para ello, hacemos clic con el botón derecho en la definición de datos para viajar desde la capa de proyección y seleccionamos Nueva definición de comportamiento.

00:11:13 Por lo tanto, todo está lleno, simplemente tenemos que presionar Siguiente. Esta vez, la proyección de tiempo de implementación está preseleccionada.

00:11:32 Y lo que tenemos que hacer es añadir un alias. Y añadimos la información que, además, queremos utilizar para ambas entidades.

00:12:20 Como recapitulación, hemos creado una proyección de modelo de datos, hemos añadido anotaciones de búsqueda, hemos permitido el uso de extensiones de metadatos, hemos añadido anotaciones de ayuda de

00:12:37 valor y hemos especificado que la asociación de reserva es una redirección a un hijo de composición. Luego, en la extensión de metadatos, hemos añadido anotaciones para cabeceras, para variante de

00:12:55 presentación y para facetas que son necesarias para que la IU de elementos de SAP Fiori se visualice correctamente.

00:13:05 Y, por último, hemos creado una proyección para nuestra definición de comportamiento y hemos especificado que queremos utilizar la etiqueta electrónica y las operaciones estándar, crear,

00:13:17 actualizar y borrar, así como la asociación habilitada para crear. En esta unidad, aprendió cómo definir la proyección del modelo de datos del business object y la

00:13:28 proyección del comportamiento del business object para las entidades de viaje y reserva. Ahora estamos al final de la unidad.

00:13:35 Gracias por escucharle y verle en la siguiente unidad. Aquí hablaremos sobre cómo crear y previsualizar el servicio de IU OData.

Week 4 Unit 5

00:00:05 Hola y bienvenido al curso de openSAP, "Building Apps with the ABAP RESTful Application Programming Model", semana cuatro, unidad cinco, "Creación y vista previa del servicio de IU de OData".

00:00:19 Me llamo Andre Fischer y soy un gestor de productos para SAP Cloud Platform, entorno ABAP y SAP Gateway.

00:00:26 Esta es la quinta y también la última unidad de la semana cuatro. En esta unidad crearemos la definición de servicio y la vinculación de servicio para nuestro business

00:00:39 object no gestionado. Con la funcionalidad de vista previa también podremos probar el servicio con una IU de elementos

00:00:49 de SAP Fiori. Si miramos el flujo de desarrollo, ahora estamos en aquellas secciones en las que se define el alcance

00:00:57 de nuestro servicio, lo que significa qué entidades forman parte del servicio y en las que este servicio está vinculado a un protocolo.

00:01:06 Comencemos con una demostración. Al final de la demostración habrá creado la definición de servicio a partir de los datos, la

00:01:14 proyección de modelo OData y habrá creado una vinculación de servicio basada en esta definición de servicio.

00:01:20 Y, por último, podrá probar su servicio con la aplicación de vista previa de elementos de SAP Fiori.

00:01:28 En esta unidad publicaremos nuestro business object como servicio OData y lo probaremos con la vista previa de los elementos de SAP Fiori.

00:01:38 Para ello, primero crearemos una definición de servicio que enumere todas las entidades CDS del modelo de datos que deben exponerse.

00:01:48 En nuestro caso, expondremos las vistas de proyección para viajes y reservas, y publicaremos todas las vistas CDS que se utilizan como ayuda para entradas.

00:01:59 Aunque esto no es absolutamente necesario, ya que todas las entidades que se utilizan como ayuda para entradas son accesibles por defecto, se trata de una buena práctica, ya que, de lo contrario, las

00:02:12 entidades se publican utilizando sus nombres técnicos. Añadir estos alias a la definición de servicio nos permite definir alias significativos.

00:02:21 En función de la definición de servicio, se creará una vinculación de servicio que vincula nuestro servicio a un protocolo específico.

00:02:29 Empezamos haciendo clic con el botón derecho en la vista de proyección de los datos de viaje. Aquí seleccionamos Nueva definición de servicio.

00:02:49 Como nombre seleccionamos ZUI_RAP_Travel_U y, a continuación, nuestro número de grupo. Para una API web, habríamos utilizado la API en lugar de la IU para el nombre de la definición de

00:03:12 servicio. Y añadimos Viajes como una descripción significativa para nuestra definición de servicio.

00:03:22 Siguiente. Seleccione nuestra orden de transporte.

00:03:29 Y seleccione la plantilla Definir servicio. Aquí sustituimos el código por lo siguiente.

00:03:45 Como hemos dicho, publicaremos las vistas de proyección para viajes y reservas, así como todas las vistas CDS que se utilizan como ayuda para entradas.

00:03:57 Y activar nuestros cambios. Hacimos clic con el botón derecho en la definición de servicio y seleccionamos Nueva vinculación de

00:04:13 servicio. Como nombre, seleccionamos ZUI_RAP_Travel_U_02 y, a continuación, nuestro número de grupo.

00:04:22 02 denota el protocolo OData que utilizaremos para publicar nuestro servicio. Y, una vez más, agregamos descripciones significativas como Viajes.

00:04:36 Pulsamos Siguiente, seleccionamos una solicitud de transporte y seleccionamos Finalizar. En esta pantalla, tenemos que activar el punto final del servicio local.

00:04:57 Por lo tanto, si seleccionamos Activar, este proceso de activación tarda un rato. Una vez activado, vemos todos los conjuntos de entidades que forman parte de nuestro servicio.

00:05:18 Y vemos la URL de servicio, así que si hace clic en esta URL, se abrirá una ventana del navegador y podríamos probar el servicio o verificar el documento de metadatos.

00:05:28 Pero aquí simplemente hacemos doble clic en Viajes. Esto abrirá una vista previa de Fiori.

00:05:44 Si no se actualiza directamente, es posible que tenga que utilizar la funcionalidad de actualización de su navegador. Si presionamos Ir, veremos los datos que se encuentran en nuestras

00:05:58 tablas antiguas. Y podemos intentar probar nuestro servicio, por ejemplo, podemos crear una nueva entrada.

00:06:06 Algo que ya probamos de antemano utilizando nuestra clase de prueba. Por lo tanto, seleccionamos una agencia, un cliente y una fecha de inicio.

00:06:26 Si, por ejemplo, intentamos introducir una fecha de fin anterior a la fecha de inicio, recibiremos un mensaje de error significativo.

00:06:38 Así pues, desde el marco heredado. Ahora seleccionamos una fecha de fin posterior a la fecha de inicio, ingresamos una tasa de reserva y

00:06:52 una moneda como el euro y un precio total. Introduzca "Primer test con IU".

00:07:02 y podemos pulsar Guardar y luego, por ejemplo, empezar a crear reservas, fecha de reserva, podemos seleccionar un cliente.

00:07:19 Y aquí elegimos el número de vuelo, porque esto, debido a las vinculaciones adicionales, rellenará todos los demás campos automáticamente y presionamos Guardar.

00:07:33 Bien, existe una fecha de reserva no válida. Por lo tanto, tendríamos que comprobarlo.

00:07:54 Ahora, después de solucionarlo, vemos todos los datos. Y como en la variante de presentación utilizamos el orden de clasificación predeterminado, nuestra

00:08:09 entrada recién creada aparece en la parte superior de la lista. Con esto, hemos probado con éxito nuestro servicio, por lo que lo publicamos y probamos con la

00:08:20 funcionalidad de vista previa de Fiori. En esta unidad, aprendió cómo crear la definición de servicio a partir de la proyección del modelo de

00:08:31 datos y cómo crear la vinculación de servicio y probar el servicio. Ahora estamos al final de la unidad.

00:08:38 Gracias por escucharte y verte la semana que viene. El tema de la próxima semana será el consumo de servicios y las API web.

www.sap.com/contactsap

THE BEST RUN SAP®