Patrón Publicar-Suscribir

Contexto: hay una serie de productores y consumidores independientes de datos que debe interactuar. El número exacto y la naturaleza de los productores y consumidores de datos no están predeterminados ni son fijos, ni tampoco los datos que comparten.

Problema: ¿Cómo podemos crear mecanismos de integración que apoyen la capacidad de transmitir mensajes entre los productores y consumidores de tal manera que ¿desconocen la identidad del otro, o incluso potencialmente su existencia?

Solución: en el patrón de publicación-suscripción, resumido en la tabla 13.8, los componentes interactúan a través de mensajes anunciados o eventos. Los componentes pueden suscribirse a un conjunto de eventos. Es el trabajo de la infraestructura de ejecución de publicación-suscripción asegúrese de que cada evento publicado se envíe a todos los suscriptores de ese evento.

Por tanto, la forma principal de conector en estos patrones es un bus de eventos. Quien publica los componentes colocan eventos en el autobús anunciándolos; el conector luego transmite esos eventos a los componentes del suscriptor que han registrado un interés en esos eventos. Cualquier componente puede ser tanto un editor como un suscriptor.

Publish-subscribe agrega una capa de indirección entre remitentes y receptores.Esto tiene un efecto negativo sobre la latencia y la escalabilidad potencial, dependiendo de cómo se implementa. Normalmente, uno no querría utilizar publicación-suscripción en un sistema que tenía plazos difíciles en tiempo real que cumplir, ya que introduce incertidumbre en tiempos de entrega de mensajes.

Además, el patrón de publicación-suscripción sufre porque proporciona menos control sobreordenar mensajes, y la entrega de mensajes no está garantizada (porque el remitente no puede saber si un receptor está escuchando). Esto puede hacer que el patrón de publicación-suscripción sea inapropiado para interacciones complejas donde el estado compartido es crítico.

Tabla 13.8 Solución de patrón de publicación-suscripción

Descripción general	Los componentes publican y se suscriben a eventos. Cuando un evento es anunciado por un componente, la infraestructura del conector envía el evento a todos los suscriptores registrados.
Elementos	Cualquier componente de C&C con al menos un puerto de publicación o suscripción. Las preocupaciones incluyen qué eventos se publican y a los que se suscribe, y la granularidad de los eventos. El conector de publicación-suscripción, que anunciará y escuchará roles para componentes que desean publicar y suscribirse a eventos.
Relaciones	La relación de adjunto asocia componentes con el conector de publicación y suscripción prescribiendo qué componentes anuncian eventos y qué componentes están registrados para recibir eventos.
Restricciones	Todos los componentes están conectados a un distribuidor de eventos que puede visto como un bus (conector) o un componente. Publicar puertos se adjuntan para anunciar roles y los puertos de suscripción se adjuntan a escuchar roles. Las restricciones pueden restringir qué

	componentes pueden escuchar qué eventos, si un componente puede escuchar sus propios eventos, y cuántos conectores de publicación-suscripción pueden existir dentro de un sistema. Un componente puede ser tanto un editor como un suscriptor, al tener puertos de ambos tipos.
Debilidades	Por lo general, aumenta la latencia y tiene un efecto negativo en la escalabilidad y previsibilidad del tiempo de entrega del mensaje. Se reduce el control sobre el orden de los mensajes y la entrega de mensajes no garantizado.

Hay algunos refinamientos específicos de este patrón que son de uso común.

Es mejor pensar en el modelo computacional para el patrón de publicación-suscripción como un sistema de procesos u objetos independientes, que reaccionan a los eventos generados por su entorno, y que a su vez provocan reacciones en otros componentes como un efecto secundario de sus anuncios de eventos. Un ejemplo de publicación-suscripción El patrón, implementado en la parte superior de la plataforma Eclipse, se muestra en la Figura 13.12.

Ejemplos típicos de sistemas que emplean el patrón de publicación-suscripción son el seguimiento:

■ Interfaces gráficas de usuario, en las que las acciones de entrada de bajo nivel de un usuario son tratados como eventos que se enrutan a los controladores de entrada apropiados

■ Aplicaciones basadas en MVC, en las que se notifica a los componentes de la vista cuando el estado de un objeto modelo cambia

■ Sistemas de planificación de recursos empresariales (ERP), que integran muchos componentes, cada uno de los cuales solo está interesado en un subconjunto de eventos del sistema.

■ Entornos de programación extensibles, en los que se coordinan herramientas a través de eventos

■ Listas de correo, donde un conjunto de suscriptores puede registrar interés en Temas

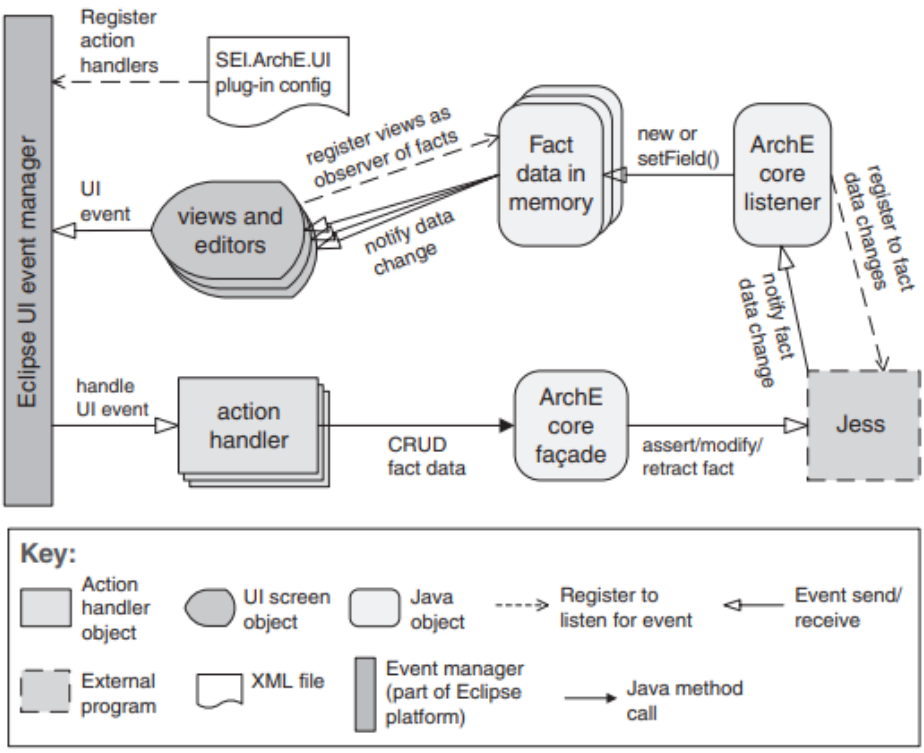

FIGURE 13.12 A typical publish-subscribe pattern realization

■ Redes sociales, donde se notifica a los "amigos" cuando se producen cambios en un sitio web de la persona

El patrón de publicación-suscripción se utiliza para enviar eventos y mensajes a un conjunto desconocido de destinatarios. Debido a que el conjunto de destinatarios del evento es desconocido para el productor de eventos, la exactitud del productor no puede, en general, depender de esos destinatarios. Por lo tanto, se pueden agregar nuevos destinatarios sin modificar el productor.

Hacer que los componentes ignoren la identidad de los demás resulta en una fácil modificación del sistema (agregar o eliminar productores y consumidores de datos), pero a costa del rendimiento en tiempo de ejecución, porque la infraestructura de publicación-suscripción es una especie de indirección, que agrega latencia. Además, si la publicación-suscripción conector falla completamente, este es un punto único de falla para todo el sistema.

El patrón de publicación-suscripción puede adoptar varias formas:

■ La **publicación-suscripción basada en listas** es una realización del patrón en el que cada editor mantiene una lista de suscripción: una lista de suscriptores que tienen registró interés en recibir el evento. Esta versión del patrón es menos desacoplado que otros, como veremos a continuación, y por lo tanto no proporcionar tanta modificabilidad, pero puede ser bastante eficiente en términos de sobrecarga de tiempo de ejecución. Además, si los componentes se distribuyen, no hay un solo punto de falla.

■ **La publicación-suscripción basada en difusión** difiere de la suscripción a publicación basada en listas en que los editores tienen menos (o ningún) conocimiento de los suscriptores.

Los editores simplemente publican eventos, que luego se transmiten. Suscriptores (o en un sistema distribuido, servicios que actúan en nombre de los suscriptores) examinar cada evento a medida que llega y determinar si el evento publicado es de interés. Esta versión tiene el potencial de ser muy ineficaz si hay muchos mensajes y la mayoría de los mensajes no son de interés para un abonado.

■ La **publicación-suscripción basada en contenido** se distingue de las dos anteriores. variantes, que en general se clasifican como "basadas en temas". Los temas son eventos o mensajes predefinidos, y un componente se suscribe a todos los eventos dentro del tema. El contenido, por otro lado, es mucho más general. Cada evento está asociado con un conjunto de atributos y se entrega a un suscriptor solo si esos atributos coinciden con los patrones definidos por el suscriptor.

En la práctica, el patrón de publicación-suscripción se realiza típicamente de alguna forma de middleware orientado a mensajes, donde el middleware se realiza como un intermediario, gestionar las conexiones y canales de información entre productores y consumidores. Este middleware es a menudo responsable de la transformación de mensajes (o protocolos de mensajes), además de enrutar y, a veces, almacenar los mensajes. Por lo tanto, el patrón de publicación-suscripción hereda las fortalezas y debilidades del patrón del corredor.