EBook Gratis

APRENDIZAJE spring-integration

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-integration

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: spring-integration

It is an unofficial and free spring-integration ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official spring-integration.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Comenzando con la integración de primavera

Observaciones

Esta sección proporciona una descripción general de qué es la integración de Spring y por qué un desarrollador puede querer usarla.

También debe mencionar cualquier tema importante dentro de la integración de primavera y vincularse con los temas relacionados. Como la Documentación para la integración de Spring es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Versiones

Versión	Fecha de lanzamiento
4.3.x	2016-11-07
4.2.x	2016-11-07
4.1.x	2016-07-25
4.0.x	2016-07-26
3.0.x	2015-10-27
2.2.x	2016-01-27
2.1.x	2013-06-10
2.0.x	2013-04-11
1.0.x	2010-04-16

Examples

Instalación o configuración

La mejor manera de comenzar a utilizar Spring-Integration en su proyecto es con un sistema de administración de dependencias, como gradle.

```
dependencies {
    compile 'org.springframework.integration:spring-integration-core:4.3.5.RELEASE'
}
```

A continuación se muestra un ejemplo muy simple que utiliza los puntos finales de mensaje de activador de servicio , puerta de enlace .

```
//these annotations will enable Spring integration and scan for components
@Configuration
@EnableIntegration
@IntegrationComponentScan
public class Application {
    //a channel has two ends, this Messaging Gateway is acting as input from one side of
inChannel
    @MessagingGateway
    interface Greeting {
        @Gateway(requestChannel = "inChannel")
        String greet(String name);
    }

    @Component
    static class HelloMessageProvider {
        //a service activator act as a handler when message is received from inChannel, in
this example, it is acting as the handler on the output side of inChannel
        @ServiceActivator(inputChannel = "inChannel")
        public String sayHello(String name) {
            return "Hi, " + name;
        }
    }

    @Bean
    MessageChannel inChannel() {
        return new DirectChannel();
    }

    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(Application.class);
        Greeting greeting = context.getBean(Greeting.class);
        //greeting.greet() send a message to the channel, which trigger service activitor to
process the incoming message
        System.out.println(greeting.greet("Spring Integration!"));
    }
}
```

Se mostrará la cadena `Hi, Spring Integration!` en la consola

Por supuesto, Spring Integration también proporciona una configuración de estilo xml. Para el ejemplo anterior, puede escribir el siguiente archivo de configuración XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:int="http://www.springframework.org/schema/integration"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/integration
        http://www.springframework.org/schema/integration/spring-integration.xsd">
    <int:gateway default-request-channel="inChannel"
                 service-
interface="spring.integration.stackoverflow.getstarted.Application$Greeting"/>
    <int:channel id="inChannel"/>
```

```
    <int:service-activator input-channel="inChannel" method="sayHello">
        <bean
class="spring.integration.stackoverflow.getstarted.Application$HelloMessageProvider"/>
    </int:service-activator>
</beans>
```

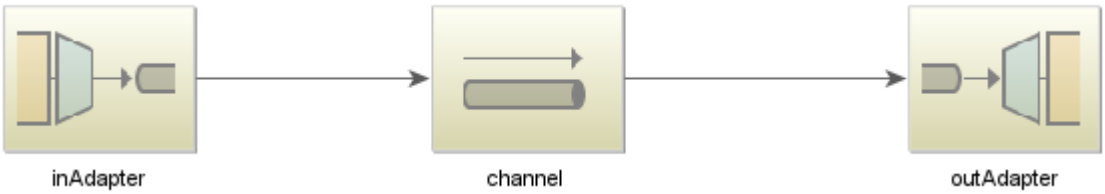
Para ejecutar la aplicación utilizando el archivo de configuración xml, debe cambiar el código `new AnnotationConfigApplicationContext(Application.class)` en la clase de `Application` a `new ClassPathXmlApplicationContext("classpath:getstarted.xml")` . Y ejecuta esta aplicación otra vez, puedes ver el mismo resultado.

Adaptador de canal entrante y saliente genérico

El adaptador de canal es uno de los puntos finales del mensaje en Spring Integration. Se utiliza para el flujo de mensajes unidireccionales. Hay dos tipos de adaptador de canal:

Adaptador de entrada: lado de entrada del canal. Escucha o lee activamente el mensaje.

Adaptador de salida: lado de salida del canal. Enviar mensaje a la clase Java o al sistema o protocolo externo.



- Código fuente.

```
public class Application {
    static class MessageProducer {
        public String produce() {
            String[] array = {"first line!", "second line!", "third line!"};
            return array[new Random().nextInt(3)];
        }
    }

    static class MessageConsumer {
        public void consume(String message) {
            System.out.println(message);
        }
    }

    public static void main(String[] args) {
        new
ClassPathXmlApplicationContext("classpath:spring/integration/stackoverflow/ioadapter/ioadapter.xm

    }
}
```

- Archivo de configuración de estilo XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:int="http://www.springframework.org/schema/integration"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/integration
http://www.springframework.org/schema/integration/spring-integration.xsd">
    <int:channel id="channel"/>
    <int:inbound-channel-adapter id="inAdapter" channel="channel" method="produce">
        <bean
class="spring.integration.stackoverflow.ioadapter.Application$MessageProducer"/>
        <int:poller fixed-rate="1000"/>
    </int:inbound-channel-adapter>
    <int:outbound-channel-adapter id="outAdapter" channel="channel" method="consume">
        <bean
class="spring.integration.stackoverflow.ioadapter.Application$MessageConsumer"/>
    </int:outbound-channel-adapter>
</beans>
```

- Flujo de mensajes
    - `inAdapter` : un adaptador de canal entrante. Invoque el método de la `Application$MessageProducer.produce` cada 1 segundo (`<int:poller fixed-rate="1000"/>`) y envíe la cadena devuelta como mensaje al canal del `channel` .
    - `channel` : canal para transferir mensaje.
    - `outAdapter` : un adaptador de canal de salida. Una vez que se haya recibido el mensaje en el canal del `channel` , este adaptador recibirá el mensaje y luego lo enviará al método `Application$MessageConsumer.consume` que imprime el mensaje en la consola.
    - Por lo tanto, puede observar que estas cadenas de selección aleatoria se mostrarán en la consola cada 1 segundo.

Ejemplo simple de eco con Spring-Integration-Stream

Código de Java:

```
public class StdioApplication {
    public static void main(String[] args) {
        new
ClassPathXmlApplicationContext("classpath:spring/integration/stackoverflow/stdio/stdio.xml");
    }
}
```

Archivo de configuración xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
       xmlns:int="http://www.springframework.org/schema/integration"
       xmlns:int-stream="http://www.springframework.org/schema/integration/stream"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/integration/stream
       http://www.springframework.org/schema/integration/stream/spring-integration-stream.xsd
       http://www.springframework.org/schema/integration
       http://www.springframework.org/schema/integration/spring-integration.xsd">
    <int:channel id="channel"/>
    <int-stream:stdin-channel-adapter id="stdin" channel="channel">
        <int:poller fixed-rate="1000"/>
    </int-stream:stdin-channel-adapter>
    <int-stream:stdout-channel-adapter id="stdout" channel="channel"/>
</beans>
```

Este es un ejemplo de eco. Cuando ejecute esta aplicación Java, puede ingresar alguna cadena y luego se mostrará en la consola.

Lea Comenzando con la integración de primavera en línea: https://riptutorial.com/es/spring-integration/topic/6985/comenzando-con-la-integracion-de-primavera

Capítulo 2: Integración Jdbc

Examples

Adaptador de entrada Jdbc - configuración xml

En el documento de referencia oficial , dice:

> La función principal de un adaptador de canal entrante es ejecutar una consulta SQL SELECT y convertir el conjunto de resultados como un mensaje. La carga útil del mensaje es el conjunto de resultados completo, expresado como una lista, y los tipos de elementos en la lista dependen de la estrategia de mapeo de filas que se utiliza. La estrategia predeterminada es un asignador genérico que simplemente devuelve un Mapa para cada fila en el resultado de la consulta.

- Código fuente

```
public class Application {
   static class Book {
      String title;
      double price;

      Book(String title, double price) {
          this.title = title;
          this.price = price;
      }

      double getPrice() {
          return price;
      }

      String getTitle() {
          return title;
      }

      @Override
      public String toString() {
          return String.format("{title: %s, price: %s}", title, price);
      }
    }

    static class Consumer {
      public void consume(List<Book> books) {
          books.stream().forEach(System.out::println);
      }
    }
```

```
    static class BookRowMapper implements RowMapper<Book> {

      @Override
      public Book mapRow(ResultSet rs, int rowNum) throws SQLException {
          String title = rs.getString("TITLE");
          double price = rs.getDouble("PRICE");
          return new Book(title, price);
      }
    }

    public static void main(String[] args) {
      new ClassPathXmlApplicationContext(
              "classpath:spring/integration/stackoverflow/jdbc/jdbc.xml");
    }
}
```

- archivo de configuración xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jdbc="http://www.springframework.org/schema/jdbc"
       xmlns:int="http://www.springframework.org/schema/integration"
       xmlns:int-jdbc="http://www.springframework.org/schema/integration/jdbc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/integration
http://www.springframework.org/schema/integration/spring-integration.xsd
       http://www.springframework.org/schema/integration/jdbc
http://www.springframework.org/schema/integration/jdbc/spring-integration-jdbc.xsd
       http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc.xsd">
    <jdbc:embedded-database id="dataSource" type="H2">
        <jdbc:script
location="classpath:spring/integration/stackoverflow/jdbc/schema.sql"/>
    </jdbc:embedded-database>
    <bean id="bookRowMapper"
          class="spring.integration.stackoverflow.jdbc.Application$BookRowMapper"/>

    <int:channel id="channel"/>

    <int-jdbc:inbound-channel-adapter id="jdbcInbound"
                                      channel="channel"
                                      data-source="dataSource"
                                      query="SELECT * FROM BOOKS"
                                      row-mapper="bookRowMapper">
        <int:poller fixed-rate="1000"/>
    </int-jdbc:inbound-channel-adapter>

    <int:outbound-channel-adapter id="outbound" channel="channel" method="consume">
        <bean class="spring.integration.stackoverflow.jdbc.Application$Consumer"/>
    </int:outbound-channel-adapter>
</beans>
```

- schema.sql

```
CREATE TABLE BOOKS (
```

```
  TITLE VARCHAR(20) NOT NULL,
  PRICE DOUBLE      NOT NULL
);

INSERT INTO BOOKS(TITLE, PRICE) VALUES('book1', 10);
INSERT INTO BOOKS(TITLE, PRICE) VALUES('book2', 20);
```

- Resumen:
  - `jdbcInbound` : un adaptador de canal entrante Jdbc. Ejecuta SQL `SELECT * FROM BOOKS` y convierte el conjunto de resultados a `List<Book>` través del bean `bookRowMapper` . Finalmente, envía esta lista de libros al `channel` .
  - `channel` : transferir el mensaje
  - `outbound` : un adaptador de salida genérico. ver Adaptador de canal entrante y saliente genérico

Adaptador de canal de salida Jdbc - configuración xml

En el Documento de Referencia de Integración de Spring , dice:

> El adaptador de canal de salida es el inverso de la entrada: su función es manejar un mensaje y usarlo para ejecutar una consulta SQL. El mensaje de carga útil y los encabezados están disponibles de forma predeterminada como parámetros de entrada para la consulta ...

- Código Java

```
public class OutboundApplication {
    static class Book {
        String title;
        double price;

        Book(String title, double price) {
            this.title = title;
            this.price = price;
        }

        public double getPrice() {
            return price;
        }

        public String getTitle() {
            return title;
        }

    }
```

```
    static class Producer {
        public Book produce() {
            return IntStream.range(0, 3)
                    .mapToObj(i -> new Book("book" + i, i * 10))
                    .collect(Collectors.toList())
                    .get(new Random().nextInt(3));
        }
    }

    public static void main(String[] args) {
        new ClassPathXmlApplicationContext(
                "classpath:spring/integration/stackoverflow/jdbc/jdbc-outbound.xml");
    }
}
```

- archivo de configuración xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jdbc="http://www.springframework.org/schema/jdbc"
       xmlns:int="http://www.springframework.org/schema/integration"
       xmlns:int-jdbc="http://www.springframework.org/schema/integration/jdbc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/jdbc
       http://www.springframework.org/schema/jdbc/spring-jdbc.xsd
       http://www.springframework.org/schema/integration
       http://www.springframework.org/schema/integration/spring-integration.xsd
       http://www.springframework.org/schema/integration/jdbc
       http://www.springframework.org/schema/integration/jdbc/spring-integration-
jdbc.xsd">
    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="url" value="jdbc:h2:tcp://localhost/~/booksystem"/>
        <property name="username" value="sa"/>
        <property name="password" value=""/>
        <property name="driverClassName" value="org.h2.Driver"/>
    </bean>
    <jdbc:initialize-database>
        <jdbc:script
location="classpath:spring/integration/stackoverflow/jdbc/schema.sql"/>
    </jdbc:initialize-database>
    <int:channel id="channel"/>
    <int:inbound-channel-adapter channel="channel" method="produce" >
        <bean
class="spring.integration.stackoverflow.jdbc.OutboundApplication$Producer"/>
        <int:poller fixed-rate="1000"/>
    </int:inbound-channel-adapter>
    <int-jdbc:outbound-channel-adapter id="jdbcOutbound"
                                       channel="channel"
                                       data-source="dataSource"
                                       sql-parameter-source-factory="sqlParameterSource"
                                       query="INSERT INTO BOOKS(TITLE, PRICE)
VALUES(:title, :price)"/>
    <bean id="sqlParameterSource"
class="org.springframework.integration.jdbc.ExpressionEvaluatingSqlParameterSourceFactory">

        <property name="parameterExpressions">
```

```
            <map>
                <entry key="title" value="payload.title"/>
                <entry key="price" value="payload.price"/>
            </map>
        </property>
    </bean>
</beans>
```

- schema.sql

```
DROP TABLE IF EXISTS BOOKS;
CREATE TABLE BOOKS (
  TITLE VARCHAR(20) NOT NULL,
  PRICE DOUBLE      NOT NULL
);
```

- Puede observar la tabla `BOOKS` y puede ver cómo se insertan los registros. O puede escribir un `int-jdbc:inbound-channel-adapter` para contar la tabla `BOOKS` y puede encontrar que el número de cuenta está creciendo continuamente.

- Resumen:

    - `inbound` : un adaptador de entrada genérico utilizado para obtener el objeto `Book` como carga útil del mensaje y enviarlo al `channel` .
    - `channel` : se utiliza para transferir el mensaje.
    - `jdbcOutbound` : un adaptador de salida JDBC, que recibe el mensaje con el `Book` tipo y luego preparar el parámetro de consulta `:title` y `:price` través `sqlParameterSource` bean mediante SpEL como `payload.title` y `payload.price` para obtener el título y el precio formar la carga útil del mensaje.

Lea Integración Jdbc en línea: https://riptutorial.com/es/spring-integration/topic/8140/integracion-jdbc

Creditos

S. No	Capítulos	Contributors
1	Comenzando con la integración de primavera	Community, Maxi Wu, walsh
2	Integración Jdbc	walsh