Бесплатная электронная книга

УЧУСЬ spring-integration

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-integration

Около

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

глава 1: Начало работы с пружинной интеграцией

замечания

В этом разделе представлен обзор того, что такое интеграция с весной, и почему разработчик может захотеть его использовать.

Следует также упомянуть любые крупные темы в рамках весенней интеграции, а также ссылки на связанные темы. Поскольку Документация для Spring-интеграции является новой, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	Дата выхода
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

Установка или настройка

Лучший способ начать использовать Spring-Integration в вашем проекте - это система управления зависимостями, например, gradle.

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

```
}
```

Ниже приведен очень простой пример использования конечных точек сообщений шлюза - активатора .

```
//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!"));
    }
}
```

Он отобразит строку `Hi, Spring Integration!` в консоли.

Конечно, Spring Integration также предоставляет конфигурацию в стиле xml. В приведенном выше примере вы можете написать следующий файл конфигурации 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>
```

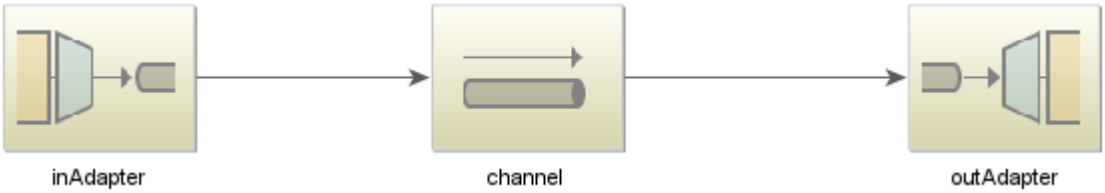
Чтобы запустить приложение с помощью файла конфигурации xml, вы должны изменить `new AnnotationConfigApplicationContext(Application.class)` код `new AnnotationConfigApplicationContext(Application.class)` в классе `Application` на `new ClassPathXmlApplicationContext("classpath:getstarted.xml")` класс `new ClassPathXmlApplicationContext("classpath:getstarted.xml")` . И запустите это приложение еще раз, вы увидите тот же результат.

Общий адаптер входящего и исходящего каналов

Адаптер канала является одной из конечных точек сообщения в Spring Integration. Он используется для однонаправленного потока сообщений. Существует два типа адаптера канала:

Входящий адаптер : входная сторона канала. Слушайте или активно читайте сообщение.

Исходящий адаптер : выходная сторона канала. Отправьте сообщение в класс Java или внешнюю систему или протокол.



- Исходный код.

```
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
```

```
    }
}
```

- Файл конфигурации в стиле 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>
```

- Поток сообщений
    - `inAdapter` : адаптер входящего канала. Invoke `Application$MessageProducer.produce` каждый раз в секунду (`<int:poller fixed-rate="1000"/>`) и отправляет возвращенную строку как сообщение каналу `channel` .
    - `channel` : канал для передачи сообщения.
    - `outAdapter` : адаптер исходящего канала. Когда сообщение достигнет канала `channel` , этот адаптер получит сообщение, а затем отправит его в метод `Application$MessageConsumer.consume` который печатает сообщение на консоли.
    - Таким образом, вы можете заметить, что эта случайная строка выбора будет отображаться на консоли каждые 1 секунду.

Простой эхо-эхо-запрос с функцией Spring-Integration-Stream

Код Java:

```
public class StdioApplication {
    public static void main(String[] args) {
        new
```

```
ClassPathXmlApplicationContext("classpath:spring/integration/stackoverflow/stdio/stdio.xml");
    }
}
```

Файл конфигурации 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>
```

Это пример эха. Когда вы запустите это Java-приложение, вы можете ввести некоторую строку, а затем она будет отображаться на консоли.

Прочитайте Начало работы с пружинной интеграцией онлайн: https://riptutorial.com/ru/spring-integration/topic/6985/начало-работы-с-пружинной-интеграцией

глава 2: Интеграция Jdbc

Examples

Входящий адаптер Jdbc - настройка xml

В официальном справочном документе говорится:

> Основной функцией входящего адаптера канала является выполнение SQL SELECT-запроса и преобразование результирующего набора в виде сообщения. Полезная нагрузка сообщения - это весь набор результатов, выраженный как список, а типы элементов в списке зависят от используемой стратегии сопоставления строк. Стратегия по умолчанию - это общий картер, который возвращает карту только для каждой строки результата запроса.

- Исходный код

```
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");
    }
}
```

- xml-файл конфигурации

```
<?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);
```

- Резюме:
  - `jdbcInbound` : адаптер входящего канала Jdbc. Он выполняет SQL `SELECT * FROM BOOKS` и конвертирует результирующий набор в `List<Book>` через bean `bookRowMapper` . Наконец, он отправляет этот список книг на канал `channel` .
  - `channel` : передача сообщения
  - `outbound` : общий адаптер исходящих сообщений. см. Общий адаптер входящего и исходящего каналов

Адаптер исходящих каналов Jdbc - настройка xml

В « Весеннем интеграционном справочном документе» говорится:

> Исходящий адаптер канала является обратным для входящего: его роль заключается в обработке сообщения и использовании его для выполнения SQL-запроса. Полезная нагрузка сообщения и заголовки доступны по умолчанию в качестве входных параметров для запроса ...

- Код 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");
    }
}
```

- Файл конфигурации 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
);
```

- Вы можете наблюдать таблицу `BOOKS` , и вы можете увидеть, что записи вставлены. Или вы можете написать `int-jdbc:inbound-channel-adapter` для подсчета таблицы `BOOKS` и вы можете обнаружить, что количество счетчиков постоянно растет.

- Резюме:

    - `inbound` : общий входящий адаптер, используемый для получения объекта `Book` качестве полезной нагрузки сообщения и отправки его на канал `channel` .
    - `channel` : используется для передачи сообщения.
    - `jdbcOutbound` : а адаптер JDBC исходящего, он получает сообщение с `Book` типа , а затем подготовить параметр запроса `:title` и `:price` через `sqlParameterSource` боб с использованием SPEL как `payload.title` и `payload.price` , чтобы получить название и цены сформировать полезную нагрузку сообщения.

Прочитайте Интеграция Jdbc онлайн: https://riptutorial.com/ru/spring-integration/topic/8140/интеграция-jdbc

кредиты

S. No	Главы	Contributors
1	Начало работы с пружинной интеграцией	Community, Maxi Wu, walsh
2	Интеграция Jdbc	walsh