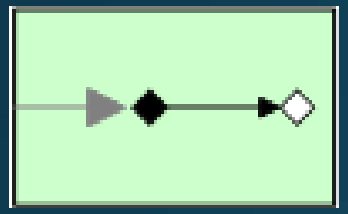


Service Activators

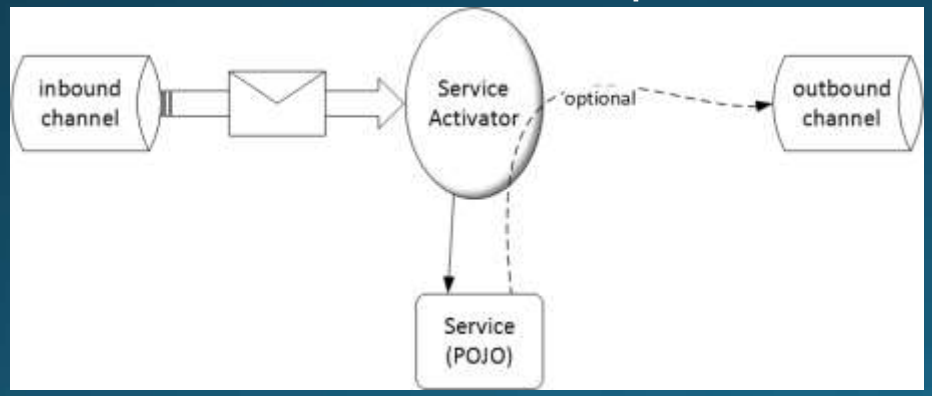
Express Spring Integration

SI Enrichers Review

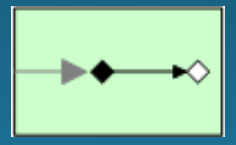
- Enrichers add information or content to an SI message.
  - Enrichers pull messages from a channel, add data or information to the message's header or payload, and post the message to another channel.
  - Enrichers are consider variants of transformers.
  - Spring provides a number of enrichers out of the box.
  - You can also create your own enrichment customization by using additional SI components.

SI Service Activators

- A message endpoint for connecting a Spring object or bean to a message channel.
  - The object or bean serves as a service.
  - The service is triggered (or activated) by the arrival of a message into the channel.
- If a service produces results, the output is sent to an output channel.



- This is the EIP icon for a service activator:



Simple Service Activator

- You have seen simple "printing" service bean in many of the labs.

```
public class ExampleServiceBean {
        public void printShiporder(Object order){
                System.out.println(order);
        }
}
```

- The service activator configuration must specify the message channel that it polls for messages, and the class of the service bean.

```
<int:service-activator id="printing-service-activator"   input-channel="my Channel" ref="serviceBean" />

 <bean id="serviceBean" class="com.intertech.lab4.ExampleServiceBean" />
```

  - Since the service bean has just one method, SI knows what method to call in the bean.
  - If there were more public methods in the service or if you wish to be explicit, the service activator would also need a method attribute.

Service Parameters

- As in the last example, the service (of the service activator) method can optionally have parameters.
  - The argument to a service method could be either a Message or an arbitrary type.

```
public void serviceMethod(Message<Foo> message){
        // service code
}
```

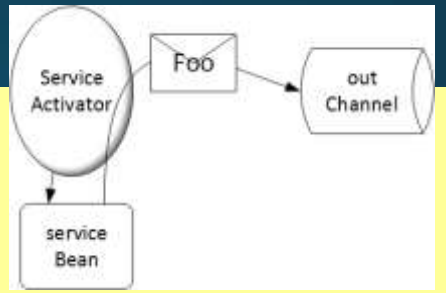
  - If it is an arbitrary type, it is assumed that the argument is the message payload extracted from the triggering message.

```
public void serviceMethod(Foo foo){
        // service code
}
```

- The service method is not required to have any arguments.
  - When passed no arguments, the message serves as a event trigger.
  - When the service method has no arguments, the service activator is referred to as ***event-style Service Activator***.

Result-Returning Service Activator

- Service methods for the service activator do not have to return a result.
  - When they do return a non-null value, the service activator attempts to send a message containing the return value to the configuration designate output-channel.




```
public Foo serviceMethod(Message<Bar> message){
        // service code
}

<int:service-activator id="my-service-activator"   input-channel="inChannel" output-channel="outChannel"
ref="serviceBean" />
```

  - If there is a return value and no "output-channel" Spring checks the triggering message for a replyChannel header and sends the result to that channel.

You are ready to tackle Lab 7

- In Lab 7, you again revisit the XML-to-object shipment order message transformation work from lab 4 & 6.
- A service activator and its associated service bean are used to calculate the total cost of each shipment order and then total up the total revenue generated by all the orders – which it displays in the Console view.