Intertech

Introduction to Spring Integration

By: Jim White

Instructors Who Consult. | **Consultants** Who Teach.

Do you remember some of those crazy kitchen utensil commercials during the 70's and 80's? One of my favorites was the Veg-O-Matic. The slogan for that tool was that "it slices, it dices"…and of course who can forget "but wait there's more!"

I have been working on some Enterprise application integration projects recently. We are using the Spring Integration framework to accomplish our work. This project within the Spring Framework portfolio amazes me. It reminds me of the Veg-O-Matic. It slices and dices data across the network and application spectrum like nobody's business. Each time I think I have seen all that the Spring Integration framework can do, I realize…"but wait there is more." It's an incredibly versatile tool.

Online Training – Right Here

I believe the Spring Integration project is a valuable tool worthy of inclusion in any Java developer's toolbox. Web mashups, cloud computing, RESTful and SOAP Web services, and big data are just some parts of the modern application portfolio that often require moving, transforming, combining, separating, filtering, and otherwise slicing and dicing many types, sizes and shapes of data in and out of application environments. In a space that often breeds home grown, complex and sloppy solutions, Spring Integration brings clarity and simplicity – all guided by well-founded patterns. If you haven't yet had a

chance to learn Spring Integration, you are in luck. This white paper walks you through a series of guided tutorials and hands-on learning labs that give you a good base by which to learn Spring Integration

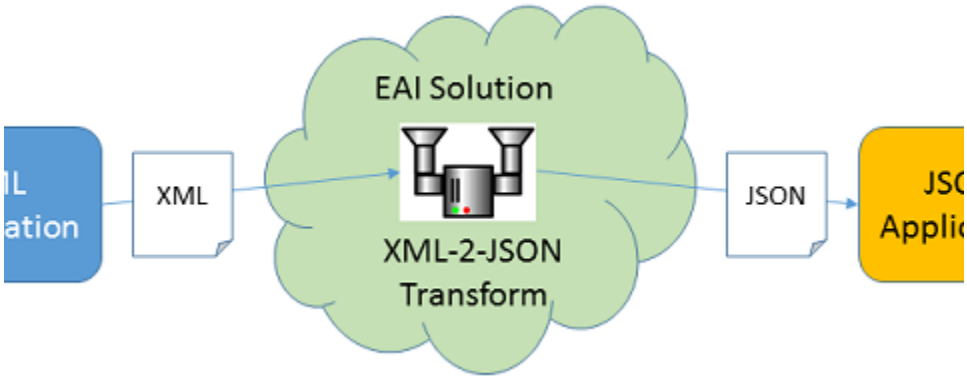
What is Enterprise Application Integration

Before we begin talking about Spring Integration, let's make sure you understand the term Enterprise Application Integration (EAI). EAI is the middleware that allows applications across an Enterprise to communicate and work together. Gartner Group is credited with defining EAI as "unrestricted sharing of data and business processes among any connected application or data sources in the enterprise." Admittedly, EAI can sound like many things and seem like a nebulous term, but that is the reality of the situation.

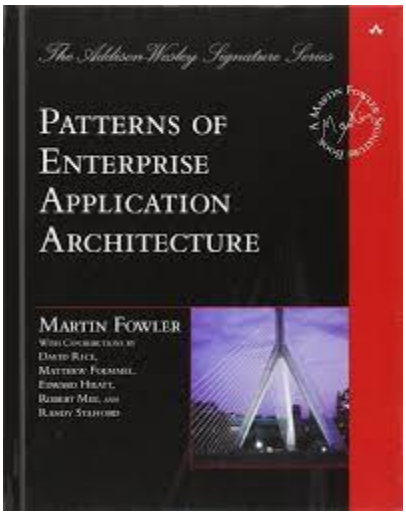
EAI is anything that allows applications to work together and share data and processes. While not always the case, the applications share data and processes that they probably weren't originally designed to share and in ways that may not have even have been conceivable at the time of their creation. Often, the applications run on different boxes, atop different operating systems, and distributed across the network (maybe even the Internet). The applications may have data stored and organized in very different forms. The applications may have been written in different programming languages.

Enterprise Integration Patterns

While it might seem that EAI defines a disparate field of middleware software, it turns out that there are some pretty common and well-understood needs within this space. For example, applications that need to communicate with each other, often speak in different data formats and there is a need to transform the data from one application format to another. My application uses and speaks XML while your application uses and speaks JSON. Before you can communicate with and make use of the services and data that my application provides, the EAI solution has to transform my data to JSON for your application to understand.



There are even well known solutions to many of the common needs in the EAI world. When you have a common solution to a commonly occurring problem you have what we call a ***design pattern***. Within EAI, there is a relatively famous and well-known collection of design patterns. They are the Enterprise Integration Patterns as defined by Gregor Hohpe and Bobby Woolf in their book by the same name: Enterprise Integration Patterns (Addison-Wesley, 2003). Hohpe and Woolf are to EAI design patterns what Gamma et. al are to general design patterns.



In fact Hohpe and Woolf not only documented the design patterns around EAI, they also developed an EAI vocabulary and graphical notation to be able to more easily discuss,

document, and design EAI applications. If you are into EAI, Hohpe and Woolf are the "rock stars" of the community and a must read.

Spring Integration (SI) is EIP Built on Spring Framework Foundation

Why are the Enterprise Integration Patterns (EIP) important to the discussion about Spring Integration framework? Because Spring Integration is physical embodiment of the EIP in Java and on top of the Spring Framework. As the Spring Integration documentation states: "Developers who have read that book should be immediately comfortable with the Spring Integration concepts and terminology." Spring Integration provides a lego-like collection of EIP building blocks that allow you to build your EAI solutions in an easy to think about, easy to assemble fashion.

Of course, Spring Integration is also built on top of the Spring Framework. This allows you to leverage the dependency injected, loosely coupled, easily extended, POJO driven capabilities of Spring in your EAI solution and even incorporate other Spring technologies (AOP, security, etc.) where necessary.

Learning Spring Integration

So what do you need know in order to learn Spring Integration? Prerequisites for learning Spring Integration and following along with the rest of my training series include knowing Java and the basics of the Spring Framework. Specifically with regard to knowing Spring, you need to understand Spring containers, dependency injection, and how to define and wire Spring beans by XML and annotations. Knowledge or at least familiarity with Enterprise Integration Patterns is helpful, but not required (however by the end of the training, you will know many of the EIP if you don't already know them).

What do you need to run the exercises that will be provided in this training series? All the labs will be written in downloadable Eclipse projects. So if you have a recent version of Java (Java 8, but Java 6 or 7 is fine), and Eclipse (I am using the Kepler train release but any Eclipse 4 or better version of Eclipse will do) you are ready. The contents of the

training will look at Spring Integration 4 (and therefore also use Spring Framework version 4 at its base.

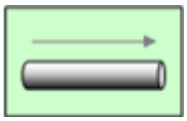
In the training, we are going to look at many of the common components (and Enterprise integration patterns) you find in EAI solutions. Specifically, we will look at Spring Integration's

- Message Channels
- Adapters
- Gateways
- Filters
- Transformers
- Enrichers
- Service Activators
- Routers

Spring Integration is a big project and there is no way to cover it all. The training should give you enough to get a sense of SI's power and capabilities while helping you see a place for it in your projects.

For each of the above components, this document provides a brief overview, a link to a training video that dives deeper into the component, and links to labs associated with each section.

PART 1 - UNDERSTANDING CHANNELS



In Part 1 of this training, I introduce you at a high level to the main components of a Spring Integration application: message endpoints, messages, and channels. Then, you dig into message channels. Channels are an integral part of any Spring Integration application. They are what tie together the various components of a Spring Integration EAI application. Watch this presentation video first before you start on the lab.

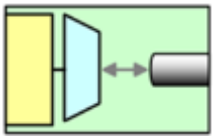
Click Here or on the image above to be taken to the training video

Once you have watched the presentation, you are ready to tackle the lab. Here is the first lab and associated code files.

- Lab1-Understanding Channels
- Lab1 code files

To view the slide deck from the above tutorial, **click here**.

PART 2 – ADAPTERS



In part two of this tutorial series on Spring Integration, I explore adapters. Adapters are messaging endpoints. In particular, they are the message endpoint that bridges your Spring Integration system to external systems and services (like JMS queues, databases, mail systems, etc.).

The presentation video, at the link below, will introduce you to adapters - in particular showing you examples of JMS adapters in Spring Integration.

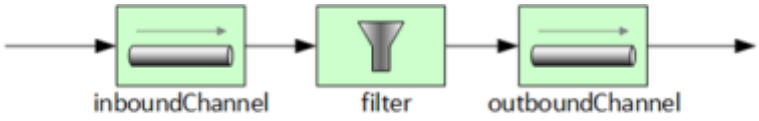
Click Here or on the image above to be taken to the training video

After watching the video, apply your knowledge via Lab 2, which has you explore Spring Integration File Adapters.

- Lab2-Adapters
- Lab2 code files

To view the slide deck from the above tutorial, **click here**.

PART 3 – FILTERS



So now you have an understanding of the channel and endpoint basics from Part 1 of our Spring Integration tutorial. And you have studied a particular type of endpoint - the adapter in Part 2 in our Spring Integration tutorial series. Time to look at another message endpoint - the filter. Filters sit between message channels.

Filters allow, on the basis of a message's content or metadata (in the message header), a message to pass from one channel to the next or reject and discard the message from the system - that is, not allowing the rejected message into the next channel. The filter is a "yea or nay" component determining which messages flow through and which messages do not.

In this third part of the tutorial series, you explore the types and configuration of filters provided by Spring Integration and how to create your own custom filter using a Spring Integration MessageSelector interface.

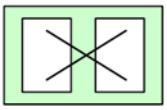
Click Here or on the image above to be taken to the training video

After watching the video, you are ready to take on the challenges of Lab 3 using Spring Integration Filters.

- Lab 3-Filters
- Lab 3 code files

To view the slide deck from the above tutorial, **click here**.

PART 4 – TRANSFORMERS



We continue our Spring Integration tutorial and exploration of Spring Integration (SI) message endpoints today with a look at transformers. Transformers take a message from a channel and create a new message containing converted payload or message structure. This allows the provider of information and the consumer of information to communicate via SI without having to settle on a common format. XML can be transformed to JSON, JSON transformed to Java objects, etc.

Part 4 of 8 in our Spring Integration Tutorial looks at SI transformers - how to use the built-in transformers and how to create a custom transformer using POJOs.

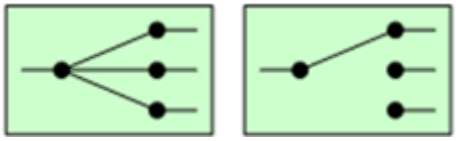
Click Here or on the image above to be taken to the training video

After watching the video, you are ready to take on the challenges of Lab 4 using Spring Integration Transformers.

- Lab 4-Transformers
- Lab 4 code files

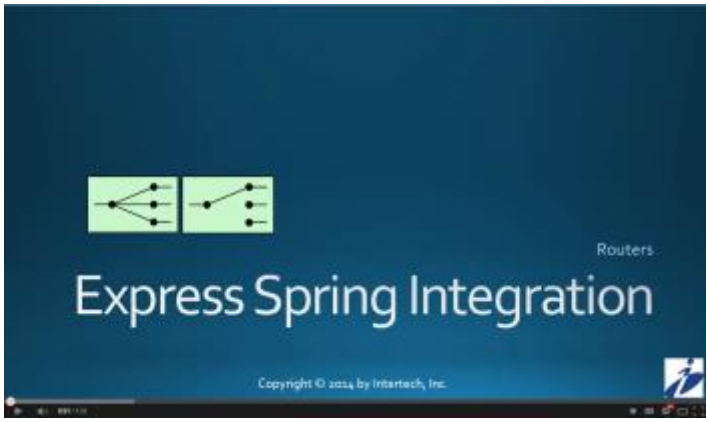
To view the slide deck from the above tutorial, **click here**.

PART 5 – ROUTERS



In this installment of our Spring Integration tutorial, we examine routers. Routers distribute messages to one or more message channels. Some routers (content routers) examine the message payload or headers in order to select a particular destination message channel. Other routers (recipient list routers) simply distribute the message to all listed message channels.

Here is part 5 of 8 in our Spring Integration Tutorial explaining routers and providing a few examples of SI routers.



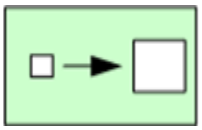
Click Here or on the image above to be taken to the training video

After watching the video, try your hand at using routers per Lab 5.

- Lab 5-Routers
- Lab 5 code files

To view the slide deck from the above tutorial, **click here**.

PART 6 – ENRICHERS



Spring Integration enrichers are really just a special type of transformer. Enrichers take a message from a channel and enhance it by adding information to its header or payload. Thus, an enricher is a transformer that only adds information to the message rather than alter it in other ways.

Learn more about SI enrichers in part 6 of 8 in the Intertech Spring Integration tutorial series.

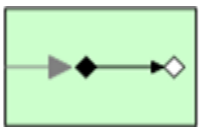
Click Here or on the image above to be taken to the training video

After watching the video, get some practical hands on with enrichers in Lab 6.

- Lab6-Enrichers
- Lab 6 code files

To view the slide deck from the above tutorial, **click here**.

PART 7 - SERVICE ACTIVATORS



As we near the end of this eight-part tutorial series on Spring Integration (SI), the topic of this seventh installment is on service activators. The name of this SI message endpoint aptly defines what it does. A service activator is an SI component that triggers (or activates) a Spring-managed service object or bean. A service activator polls a message channel looking for messages. On the arrival of a message, it calls the processing method of the service bean (which is typically just a POJO).

The service's processing method is passed the message or the payload of the message based on the parameter type to the processing method. In fact, the service's processing method can be passed no data. In which case, the service activator is considered an event-style component that triggers processing just on the mere arrival of the message. However, the processing method often uses the message data (its payload or header information) to perform work. The service's processing method may also optionally return a value or message. The output, when returned, can be sent to an SI output channel.

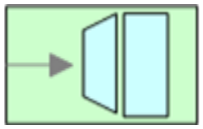
Click Here or on the image above to be taken to the training video

Once done with the video session, get some hands on experience through Lab 7.

- Lab7-ServiceActivators
- Lab 7 code files

To view the slide deck from the above tutorial, **click here**.

PART 8 – GATEWAYS



In this final part on my Spring Integration tutorial series (SI), we take a look at gateways. Gateways are a means of loosely coupling other application components from the SI API or other messaging API. The gateway serves as a façade to a SI system.

Gateways are defined by an interface. The gateway can either by synchronous (causing the application to block and wait for the SI system to respond) or asynchronous (allowing the application to do other work while a long running SI system processes). Learn how to code and configure gateways in this tutorial.

Click Here or on the image above to be taken to the training video

Lab 8 provides you hands-on experience with a synchronous and asynchronous gateway.

- Lab8-Gateways
- Lab 8 Code Files

To view the slide deck from the above tutorial, **click here**.

Wrap Up

Here at Intertech we offer helpful training courses and consulting services to make effective use of Spring Integration.

Our Training Course

- https://www.intertech.com/Training/Java/Frameworks/Spring

Our Consulting Practice

- https://www.intertech.com/Consulting/Spring-Framework-Consulting