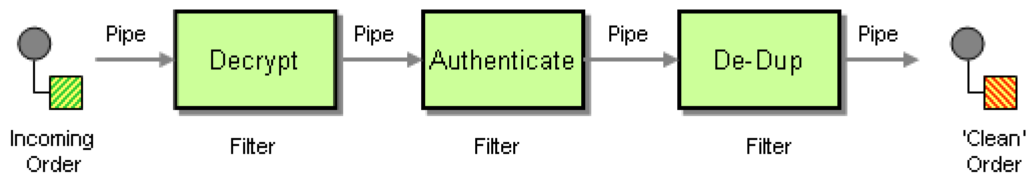
PIPES AND FILTERS

In many enterprise integration scenarios, a single event triggers a sequence of processing steps, each performing a specific function. For example, let's assume a new order arrives in our enterprise in the form of a message. One requirement may be that the message is encrypted to prevent eavesdroppers from spying on a customer's order. A second requirement is that the messages contain authentication information in the form of a digital certificate to ensure that orders are placed only by trusted customers. In addition, duplicate messages could be sent from external parties (remember all the warnings on the popular shopping sites to click the 'Order Now' button only once?). To avoid duplicate shipments and unhappy customers, we need to eliminate duplicate messages before subsequent order processing steps are initiated. To meet these requirements, we need to transform a stream of possibly duplicated, encrypted messages containing extra authentication data into a stream of unique, simple plain-text order messages without the extraneous data fields.

How can we perform complex processing on a message while maintaining independence and flexibility?



Use the *Pipes and Filters* architectural style to divide a larger processing task into a sequence of smaller, independent processing steps (Filters) that are connected by channels (Pipes).

Each filter exposes a very simple interface: it receives messages on the inbound pipe, processes the message, and publishes the results to the outbound pipe. The pipe connects one filter to the next, sending output messages from one filter to the next. Because all component use the same external interface they can be *composed* into different solutions by connecting the components to different pipes. We can add new filters, omit existing ones or rearrange them into a new sequence -- all without having to change the filters themselves. The connection between filter and pipe is sometimes called *port*. In the basic form, each filter component has one input port and one output port.

Source:
http://www.enterpriseintegrationpatterns.com/patterns/messaging/PipesAndFilters.html