Exploring Asynchronous Integration

7

Overview

Asynchronous integration occurs when a number of processes integrate but do not lock for the time of the transaction on which they are integrated. In general, a caller makes a request of a server and then goes away and does its own thing. When the server finishes its part of the process, it sends the results back to the caller via a **callback**. This involves the caller and the server keeping information about each other in what is usually referred to as a **session**. It involves overhead on both sides, and as such, in large scale systems it requires careful design and usage.

In a typical scenario, asynchronous integration is appropriate where a request process takes a long period of time or if a client application can continue processing without waiting for a response. Asynchronous communication is often utilized for the Enterprise Application Integration (EAI). For

example, processing an order fulfillment system often relies on asynchronous processing of incoming orders. Once the order is processed, a status confirmation is sent to the user or application that placed the order.

A classic example of an asynchronous service is when one orders a ticket for an airline. Typically the scenario starts with someone making a request, and then the system goes off and checks availability of the requested resource. When ordering an airplane seat, it assigns that seat to you and blocks it off for all other sessions.

The first asynchronous session occurs when the ticket query is made. In a typical browser session you get a 'waiting' screen, but what is happening here is that the HTTP session is being kept alive while the query for seats is being made. Once the query is complete, the server finishes the process, downloads the new data, and closes the HTTP session. (This isn't a perfect example because in a true asynchronous case, the user could go and browse other Web sites with the same browser and get redirected to the seat availability screen when the server is ready, but the HTTP protocol doesn't allow for this.)

At this point, the session is being maintained on the server, usually with an expiration time of two minutes. Should the user not complete the transaction in this timeframe, the session dies, and the resources are released. The next step of the session is that you confirm you want the seats and inform the server. In this case the session stays alive while the server finalizes the booking and blocks off the seats for you.

A third asynchronous session takes place for payment, with a call to another server to validate the given credit card information. This calls back to the booking server with a positive or a negative and based on those results the booking server calls back to the browser with the results of the successful booking or a request for an alternative payment.

In this manner two or more remote systems can interoperate in a more efficient way by hiding internal system complexity and only exposing high-level services forming an asynchronous Service Oriented Architecture (SOA).

Using Asynchronous Integration

Asynchronous integration, as seen in the given example, is generally used when transactions have a long lifetime or require resources that are shared between many users to be locked (like seats on an airplane) for the life of the transaction. Typical scenarios for asynchronous integration are

- In a peer to peer process (as opposed to a client server process). For example when making a stock trade, either selling or buying, an asynchronous implementation is necessary. Generally one requests a trade from the server, and the server then tries to find someone in the market that is requesting the resource that is being bought or sold. Negotiation may take place based on pricing rules (such as Bid/Ask automatic values based on lot size) and a price is met, the trade is executed and each party informed. This process can take several microseconds (in the case of a market trade) or years (in the case of a stop/limit trade).
- When notifications of the status of a transaction are required at fixed times or when based on certain external events occurring. A good example of this is an online auction. A fixed time update is required at the end of the auction; external events occur when others bid on the auction and you need to know what they are.
- When it takes a lot of time for a process to complete. For example if the call to a server requires a bulk update to a database that can take a long time. An asynchronous update informs the caller in this case when the transaction is complete. When using a synchronous transaction, the session would have to stay open for the lifetime of the process, or the caller would have to continually ping the server for the results.

There are several technologies that can be used to realize asynchronous integration, which are explored in more detail in the following chapters.