

External System wants to communicate with our systems through many ways Ex:

* JSON input
* XML input
* File input
* Command Line input or Curl commands
* Web-Service input
* JMS Message etc…

**How Integrating systems (Integration Framework and ESB) evolved**

1. Initially we stored our data as file.
2. Instead of storing data in files, SQL databases came to rescue.
3. Relational DBs are costlier operation because of transaction, so NoSQL databases are evolved in some use cases.
4. Then for distributed access, Synchronous remote procedure calls or asynchronous messaging is used to communicate via several technologies such as RMI, SOAP Web Services, REST or JMS.
5. **Most of the time, all these applications and products built on decades have to communicate with each other to work together perfectly which was solved by integration framework and ESB**
6. Of course, you could reinvent the wheel for each problem, write some spaghetti code and let the applications interact and work together. Unfortunately, your management will not like the long-term perspective of this solution. Thus, it is a good idea to forget about reinventing the wheel each time you have a problem.

**HOST to Host 🡪Client / Server Model 🡪 SOA 🡪 Cloud 🡪 Big Data**

**Why Integration Framework and ESB**

Data exchanges between companies increase a lot. The number of applications which must be integrated increases, too. The interfaces use different technologies, protocols and data formats. Nevertheless, the integration of these applications shall be modeled in a standardized way, realized efficiently and supported by automatic tests.

**Practical Approach to Integration Framework and ESB**

2 integration frameworks mostly used are

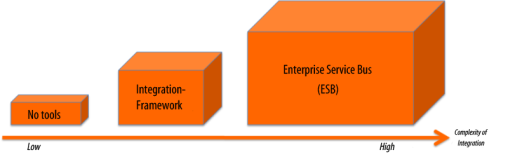
1. Spring Integration,
2. Apache Camel. They implement the well-known Enteprise Integration Patterns and offer a standardized, domain-specific language to integrate applications.

4 ESB solutions mostly used are

1. Mule ESB
2. Apache ServiceMix
3. JBOSS Fuse ESB
4. Open ESB
5. Tibco

ESB generally contains a) Integration Framework 2) Business Process Flow 3) Monitoring Support 4) Service Registry and Discover etc…

**Integrating Systems Practical Approach**



1. The custom development approach (No tools) is suitable for small applications because it is the simplest and fastest to use. In such developments, the source code is simply implemented via the Java standard APIs. For example, code that reads and processes data from a file or sends a JMS message can quickly be generated and easily maintained without an integration framework. In addition, small utility libraries such as the Apache Commons IO or the Spring JMS Template can be used. For small integration tasks, these are, in most cases, preferable to an integration framework.
2. For more complex integration scenarios, an ESB can be used instead of an integration framework. ESBs are significantly more powerful, but also more complex than integration frameworks. Furthermore, ESBs offer extended features such as a registry, a rules engine, BPM and Business Activity Monitoring. If these features are required in addition to the actual integration, using an ESB is advisable because it will allow the entire integration process to be handled within one stack. Combining several small frameworks to build a custom ESB is usually unnecessary and can introduce numerous extra pitfalls

<http://www.h-online.com/developer/features/Free-integration-frameworks-on-the-Java-platform-1672712.html%3Fpage=3>