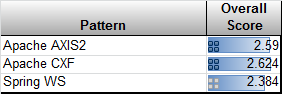
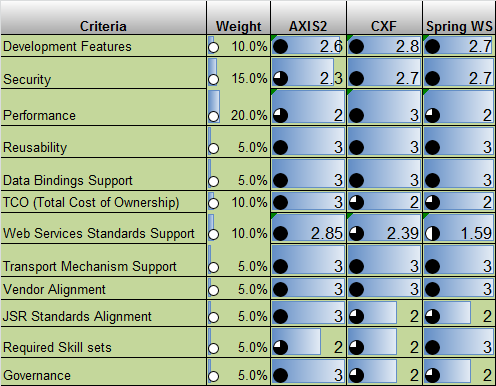
1. **Interface to EIE**: There are numerous solutions which can be adopted to provide an external interface to EIE through which it can interact with our Cache solution. For e.g. we can write a Web Service which will perfume CRUD operations on Cache DB. Then we can provide wsdl to client through which they can generate stubs to connect to our exposed WS. But this mode of communication is very heavy. So we may use REST which is light weight. But REST has several implementations like Jersey, Apache Client. But we need to integrate with Spring as well as it provides high level of decoupling. So there is another solution regarding Apache CXF.

In view of this we have drawn several comparisons and here is our analysis:

[](http://1.bp.blogspot.com/-jwbLq6Bx2-8/UYjba7QsZyI/AAAAAAAABF4/uFUzbR5idw0/s1600/overall.png)

[](http://2.bp.blogspot.com/-2B8XnUfrdJk/UYjbREATpEI/AAAAAAAABFw/TKFGNY175jY/s1600/fw-eval.png)

Conclusion:

•Apache AXIS2 is relatively most used framework but Apache CXF scores over other Web Services Framework comparatively considering ease of development, current industry trend, performance, overall scorecard and other.

•Though Spring Core Framework is an established technology, Spring Web Services is still evolving in comparison with Apache CXF and CXF has wider support of standards and proven framework from performance perspective.

•Hence, Apache CXF is the recommended framework and clearly the most preferred our opinion.

1. **References**:

<http://cxf.apache.org/docs/features.html>

<http://stackoverflow.com/questions/16783851/what-is-benefit-of-developing-webservice-using-apache-cxf-over-normal-jax-ws-wit>

<http://architects.dzone.com/articles/apache-cxf-vs-apache-axis-vs>

**Purging Strategy**

* For the purging strategy we looked into Quartz, but Cassandra itself provides the inbuilt mechanism of TTL while inserting column data.
* Timezone impact across multiple data centers is also self-managed by Cassandra.
* Removes additional overhead of scheduling any external purging framework.
* TTL can be specified at a more lowest level to a particular column

**Build tool Gradle**

* Focused on JAVA projects
* Build scripts are code
* Move from XML to Groovy ( no longer xmls, few lines of groovy does the job)
* Supports Maven central, Local Maven repository as well as local jars as dependencies
* Plugin support (project report, dependency report, find-bugs report, sonar report, junit test case reports…all of these comes as plugin)
* Gradle Wrapper (For deploying on machines which doesn’t have gradle installed. We can embed gradle wrapper in our source code management and create build)

**Datastax Driver**

This Driver we will be using for connecting with the Cassandra DB through java code.

DataStax Driver is based on new binary protocol. This protocol is the custom one and offers the following features over the other driver such as Hector:

* Asynchronous: each connection can handle more than one active request at the same time. In practice, this means that client libraries will only need to maintain a relatively low amount of open connections to a given Cassandra node to achieve good performance. This particularly matters with Cassandra where a client usually wants to keep connection to all (or at least a good part of) the nodes of the Cluster and so having a low number of per-node connections helps scaling to large clusters.  
  Technically, this is achieved by giving each messages a *stream ID*, and by having responses to a request preserve the request’s stream ID. Clients can thus send multiple requests with different stream IDs on the same connection (i.e. without waiting for the response to a request to send the next one) while still being able to associate each received response to the right request, even if said responses comes in a different order than the one in which requests were submitted. That asynchronicity is of course optional in the sense that a client library can still choose to use the protocol in a synchronous way if that is simpler.
* Server notifications: the protocol allows clients to register for certain types of events notifications. The currently supported events are cluster topology changes (a node join the cluster, is removed, or move), status changes (a node is detected up/down) and schema changes (the schema has been modified). When one of those events occurs, the server will push a notification to the registered clients. This allows those clients to maintain a state of the Cassandra cluster up to date without having to poll the cluster regularly. Obviously, more type of notifications might be added in the future, opening up a number of interesting possibilities.
* Optional compression: messages of the protocol can be optionally compressed.