Spring – 2014
CmpE – 226 – 01

Kaustubh Deshmukh
20, May, 2014

# Mongo Hands-on Assignment

**Assignment Summary:**

**Technology used:** Java-J2EE (JAX-RS)
**Build tool used:** Maven
**Frameworks used:** Jersey (RESTful service development), Bootstrap (CSS), Jackson (JSON handling), JSTL (JSP design)
**Server used:** Apache Tomcat v7.0
**GitHub URL:** https://github.com/KaustubhDeshmukh/CmpE-226-Mongo-Assignment
**HomePage URL:** http://<ServerName>:<HTTP-Port>/CmpE-226-Mongo-Assignment/home - (After deployment)
**Prerequisite to run:** Mongo running as a service or as an instance

**Assignment Q&A:**

**Q:** What are the advantages of using Mongo over traditional RDBMS for this application?

**A:** Mongo has following positives for the developed application and its domain:

- **Flexible Schema:**
  As Mongo does not enforce any constraints over structure of documents, contents of collection and document nesting; we can modify the product attributes very easily and without any side effects.
- **Scalability with No-Joins:**
  After initial "Mongos" (Mongo shard) configuration setup, MongoDB automates the data sharding and hence it is very agile in scaling out. We can create shards from different collections and tune it in application specific way and as mongo does not have any "Join" concept, it can scale very easily.
- **High availability:**
  Mongo also provides automated replication with minimum configuration and when combined with sharding it becomes ideal persistence solution for applications that needs to be highly available and with agile scalability.
- **Dynamic-queries with built-in map-reduce:**
  Mongo inherently supports the map-reduce and dynamic querying and that can be used for small mining tasks for quicker prediction, forecasts and preemptive actions for any domain specific issues such as inventory issues, offer-exploitation.

**Q:** What are the dis-advantages of using Mongo over traditional RDBMS for this application?

**A:** Mongo has following negatives for the developed application and its domain:

- **Absence of transactions:**
  As a trade-off to scalability, Mongo has sacrificed the transactional support to the group of operations. This jeopardizes consistency across different documents and over different collections. Application has to ensure that multiple transactions are being performed in atomic way and when fire-and forget approach is combined with replication it becomes mandatory for application to wait **programmatically** for the completion of the transaction. For ex: Inventory management with order processing.
- **Difficulty in object mapping and use of ORM frameworks:**
  Due to schema-less approach and absence of joins, mapping the documents and their relationships in Objects becomes slightly difficult, as there is no explicit way of describing the hierarchy. Also the use of ORM framework is quite susceptible to breakage due to no schema enforcement of document storage. In reality, it contradicts the Mongo virtues.
- **Comparatively large data-size of the collection:**
  As every document has field names associated with its values, it becomes redundant across all the similar objects having same schema. When collection grows, redundant memory usage becomes significantly notorious and results in large memory mapped files. For ex: Product attributes will be redundantly stored with their values.
- **Larger Applications:**
  Due to absence of joins and transaction, application has to compensate for these features programmatically. This results in the large applications and hence the increase in bugs and maintenance cost. Actually this is the trade-off of using Mongo.