Couchbase

CD220 Developing Couchbase NoSQL Applications

Enroll Now

Learn hands-on development of scalable NoSQL applications

CD220 Developing Couchbase NoSQL Applications - This course provides 3 days of progressive hands-on flight time building NoSQL web and enterprise applications. Use of an engaging real-world data model and codebase enables a practical approach to NoSQL application development.

Through instructor-led discussion, demonstrations and intensive programming labs, participants will develop an actual Couchbase NoSQL app that reflects typical use-cases for scalable NoSQL databases operating across the cloud, leveraging Couchbase Server's capabilities, flexibility and performance.

Objectives: The course leverages the Couchbase Client Java SDK, Gson/JSON, and the CrudRepository<T> pattern. We start with a high level perspective on NoSQL applications, Couchbase Server, installing, and starting Couchbase Server; basic administration overview. From there, each lab exercise builds progressive skills in developing NoSQL applications that make use of Couchbase Server's scalability and performance.

Each student will practice and be able to perform all the basic NoSQL application development tasks that are required of a modern web application in the real world. Creating Buckets and usage of View/Index, Map/Reduce, and document oriented data will be gained through more than a dozen lab exercises which challenge the student to build real solutions with this NoSQL technology.

The client side Java SDK will be used extensively, including but not limited to: The Write Path, The Read Path, add, replace, CAS, set, and get, writing a Bucket utility with ClusterManager, writing a ClientFactory, data modeling with a document database, denormalization, CrudRespositories and Services, and View API with index creation and use.

The course will include use of the Couchbase Query Language: N1QL, and an introduction to Elastic Search with Couchbase. Students will learn the top ten current developer issues and best practices.

Audience: This is a hands-on developer training for people who want to get real work done with NoSQL database applications, working with Couchbase Server and the Client SDK. While the labs will be based on the Java SDK, the concepts, techniques and methods are applicable to any of the available SDKs (.NET, PHP, Ruby, Python, C, Node.js). Expect to spend more time writing solutions and less time watching a presentation. If you have a background in programming, want to write software and are technically savvy this is the right course for you.

Pre-requisites: The course is designed for participants with experience writing software in some modern language, such as Java, or C/C++, C#/.Net, or Python, Ruby, PHP, etc. Familiarity with basic programming fundamentals and database essentials is assumed. Experience with web development technologies (ASP, JSP, Perl, HTML, JavaScript) may be helpful.

Participation: since the class covers concepts at a rapid pace and because lab assignments are additive and progressive, it is important for students to devote their full attention to the training. If students miss one of the earlier labs or fail to complete it this may prevent or reduce ability to complete subsequent labs.

[*continued on next page...*]

Terms: Enroll worldwide 24/7 at training.couchbase.com via valid credit card for immediate confirmation. Enterprise customers may purchase Pre-Paid Class credits (PPC) to be redeemed by authorized Customer personnel for available seat(s) in any public class session(s). Customer provides name and email of contact authorized to approve usage of PPC credits. PPC credits must be used within one year for enrollment in open-enrollment training session(s) available from Couchbase. To reach Couchbase Enterprise Sales: please email training@couchbase.com. Additional offerings, terms and conditions at training.couchbase.com

Hardware and Systems: classroom will usually be equipped with PC type workstations (which may be laptops) with minimum of 4GB RAM, or Apple Mac computers, with minimum of 4GB RAM. Labs will be run on virtual machine instances, provided.

The current version of this class is focused on the Java SDK. The concepts are the same across all available SDKs. Labs and lab solutions will be provided in Java. We provide Eclipse as the IDE for all labs. If you want to use another IDE, you can. We will use Gson for JSON transformations, as well as JUnit. For those who wish to use Maven rather than the available JARs, an internet connection is required.

Couchbase Server 2.5 Enterprise Edition currently supports the following Operating Systems: Ubuntu Linux, Red Hat Linux, Windows, Mac OS X

Outline: a ***partial list*** of concepts, topics and labs, subject to changes and updates, may include:

I. Introduction to Couchbase:
   a. What is it?
   b. What Couchbase is not.
   c. Where Couchbase server fits in the Couchbase ecosystem
   d. Mobile Development: Couchbase Lite & Sync Gateway
   e. Administration (Ops) overview.

II. Starting Couchbase Server
   a. Cluster architecture overview.
   b. View Cluster:
   c. Create Data Buckets (two types, Memcached or Couchbase):
   d. Connecting to the Cluster and the Bucket(s).
   e. Document Basics:
   f. Couchbase Java Client SDK Overview:

III. Developing Applications with Couchbase Server:
   a. Not reviewing Couchbase Lite, that is another course.
   b. Choosing your SDK: For our course, Java SDK
   c. Choosing your IDE: in our course, Eclipse.
   d. Putting dependencies in place.
   e. Overview of CrudRepository<T>, etc.
   f. Overview of Using Gson to migrate POJOs <~> Couchbase.
   g. View API Overview (Query only works with Views).
   h. Understanding single data center replication.
   i. Understanding Cross Data Center Replication (XDCR)

IV. Couchbase Query Language: N1QL (pronounced "Nickel")
   a. In Developer Preview 2 (as of February 2014).
   b. Beyond "Key/Value" access patterns.
   c. SQL-like syntax
   d. Tightly integrated with Couchbase Server.

V. Introduction to Elastic Search (ES):
   a. Data Modeling with ES
   b. Querying and Indexing
   c. Couchbase plug-in for ES
   d. Configuration needed
   e. Starting ES
   f. Setup XDCR between CB and ES

VI. "Top 10" Developer Issues (as of February 2014):
   a. Too many client connections.
   b. Not using Config Cache.
   c. Don't use views with large (close to 1MB & larger) documents.
   d. Error (timeout) handling.
   e. Writing good views.
   f. When to have more than one bucket.
   g. Randomizing node list prior to connection.
   h. Why do I need to reuse an object, I did not have to with memcached object.
   i. Do I need to use Replica Read to scale reads

Partial List of Labs:

Lab 1. Install/Start Couchbase Server
Lab 2. Configure IDE with JARs/Maven
Lab 3. Use ClusterManager to create a Couchbase Named Bucket for the PlayList application
Lab 4a. Create a CouchbaseClientFactory for PlayList application
Lab 4b. Refactor CouchbaseClientFactory to handle many clients (one per named bucket)
Lab 5. Create a User document using JSON and save the document to Couchbase Server with set(…)
Lab 6a. Create a UserRepository interface which extends CrudRepository<T>, and implement the save(…) method
Lab 6b. Write an overloaded save(Iterable<S> entities) method
Lab 6c. Write a refactored save(…) using the add(…) method
Lab 7. Create the UserRepositoryImpl update(…) methods
Lab 8. Create the UserRepositoryImpl exists(String key) method
Lab 9. Create the UserRepositoryImpl User findOne(String key) method
Lab 10. Create the UserRepositoryImpl delete(String key), delete(User u), and delete(Iterable<User> users) methods
Lab 11. Create a View/Index for users, by userId for use by the remaining CrudRepository methods: count(), findAll() and deleteAll()
Lab 12. Create the long count() method using the Query API types: .Query, .View and ViewResponse
Lab 13. Create the Iterable<User> findAll() method using the types: Paginator, as well as ViewResponse and ViewRow
Lab 14. Create the deleteAll() method using the Paginator
Lab 15. Data Modeling: De-normalization Basics and Documents with Json
Lab 16. Data Modeling: PlayList, Song, SongInfo, and PlayListService which <<HAS-A>> SongRepository
Lab 17. Create the SongRepositoryImpl methods
Lab 18. Create the PlayList class
Lab 19. Create the PlayListService class
Lab 20. Create an AuthenticationService which <<HAS-A>> UserRepository, and implement the authenticate(Credentials cred) method
Lab 21. Create the PlayListController class <<HAS-A>> AuthenticationService, and <<HAS-A>> PlayListService;
Lab 22. Instantiate a User who is already in the system, and authenticate them, and then allow that authenticated user to test all the methods available in the PlayListService

Please Note: Course content, topics, labs, software versions, pre-requisites, duration, start and/or end times, etc., are subject to change without prior notice, at sole discretion of Couchbase, from time to time as we continuously update and iteratively enhance our hands-on enterprise training offerings. Check online course description at training.couchbase.com for further information, and to find schedule of upcoming classes in a city nearest to you. Additional offerings, terms and conditions at training.couchbase.com