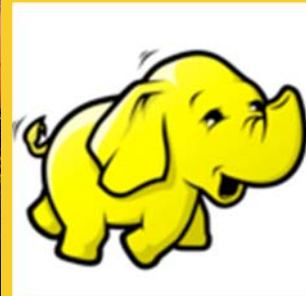




HBase Overview

Originals of slides and source code for examples: <http://www.coreservlets.com/hadoop-tutorial/>
Also see the customized Hadoop training courses (onsite or at public venues) – <http://courses.coreservlets.com/hadoop-training.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live customized Hadoop training (including prep for the Cloudera certification exam), please email info@coreservlets.com

Taught by recognized Hadoop expert who spoke on Hadoop several times at JavaOne, and who uses Hadoop daily in real-world apps. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - JSF 2.2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
 - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services

Contact info@coreservlets.com for details



Agenda

- **Overview**
- **Data Model**
- **Architecture**
- **Resources**

4

HBase

- **Column-Oriented data store, known as “Hadoop Database”**
- **Supports random real-time CRUD operations (unlike HDFS)**
- **Distributed – designed to serve large tables**
 - Billions of rows and millions of columns
- **Runs on a cluster of commodity hardware**
 - Server hardware, not laptop/desktops
- **Open-source, written in Java**
- **Type of “NoSQL” DB**
 - Does not provide a SQL based access
 - Does not adhere to Relational Model for storage

5

HBase

- **Horizontally scalable**
 - Automatic sharding
- **Strongly consistent reads and writes**
- **Automatic fail-over**
- **Simple Java API**
- **Integration with Map/Reduce framework**
- **Thrift, Avro and REST-ful Web-services**

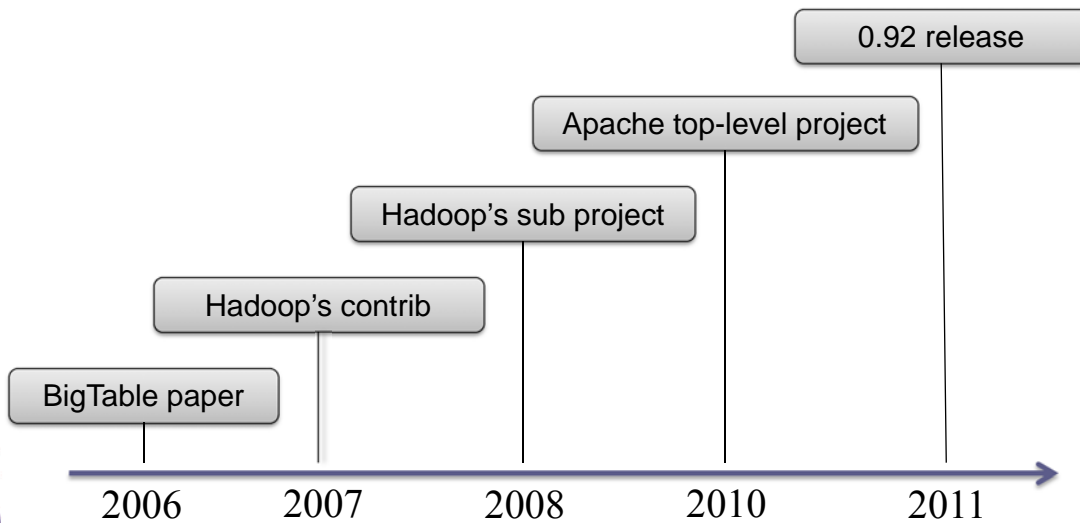
6

HBase

- **Based on Google's Bigtable**
 - <http://labs.google.com/papers/bigtable.html>
- **Just like BigTable is built on top of Google File System (GFS), HBase is implemented on top of HDFS**

7

HBase History



8

Source: Lars, George. [HBase The Definitive Guide](#). O'Reilly Media. 2011

Who Uses HBase?

- **Here is a very limited list of well known names**

- Facebook
- Adobe
- Twitter
- Yahoo!
- Netflix
- Meetup
- Stumbleupon
- You????



9

When To Use HBase

- **Not suitable for every problem**
 - Compared to RDBMs has VERY simple and limited API
- **Good for large amounts of data**
 - 100s of millions or billions of rows
 - If data is too small all the records will end up on a single node leaving the rest of the cluster idle

10

When To Use HBase

- **Have to have enough hardware!!**
 - At the minimum 5 nodes
 - There are multiple management daemon processes: Namenode, HBaseMaster, Zookeeper, etc....
 - HDFS won't do well on anything under 5 nodes anyway; particularly with a block replication of 3
 - HBase is memory and CPU intensive
- **Carefully evaluate HBase for mixed work loads**
 - Client Request vs. Batch processing (Map/Reduce)
 - SLAs on client requests would need evaluation
 - HBase has intermittent but large IO access
 - May affect response latency!!!

11

When to Use HBase

- **Two well-known use cases**
 - Lots and lots of data (already mentioned)
 - Large amount of clients/requests (usually cause a lot of data)
- **Great for single random selects and range scans by key**
- **Great for variable schema**
 - Rows may drastically differ
 - If your schema has many columns and most of them are null

12

When NOT to Use HBase

- **Bad for traditional RDBMs retrieval**
 - Transactional applications
 - Relational Analytics
 - 'group by', 'join', and 'where column like', etc....
- **Currently bad for text-based search access**
 - There is work being done in this arena
 - HBasene: <https://github.com/akkumar/hbasene/wiki>
 - HBASE-3529: 100% integration of HBase and Lucene based on HBase' coprocessors
 - Some projects provide solution that use HBase
 - Lily=HBase+Solr <http://www.lilyproject.org>

13

HBase Data Model

- **Data is stored in Tables**
- **Tables contain rows**
 - Rows are referenced by a unique key
 - Key is an array of bytes – good news
 - Anything can be a key: string, long and your own serialized data structures
- **Rows made of columns which are grouped in column families**
- **Data is stored in cells**
 - Identified by row x column-family x column
 - Cell's content is also an array of bytes

14

HBase Families

- **Rows are grouped into families**
 - Labeled as “family:column”
 - Example “user:first_name”
 - A way to organize your data
 - Various features are applied to families
 - Compression
 - In-memory option
 - Stored together - in a file called HFile/StoreFile
- **Family definitions are static**
 - Created with table, should be rarely added and changed
 - Limited to small number of families
 - unlike columns that you can have millions of

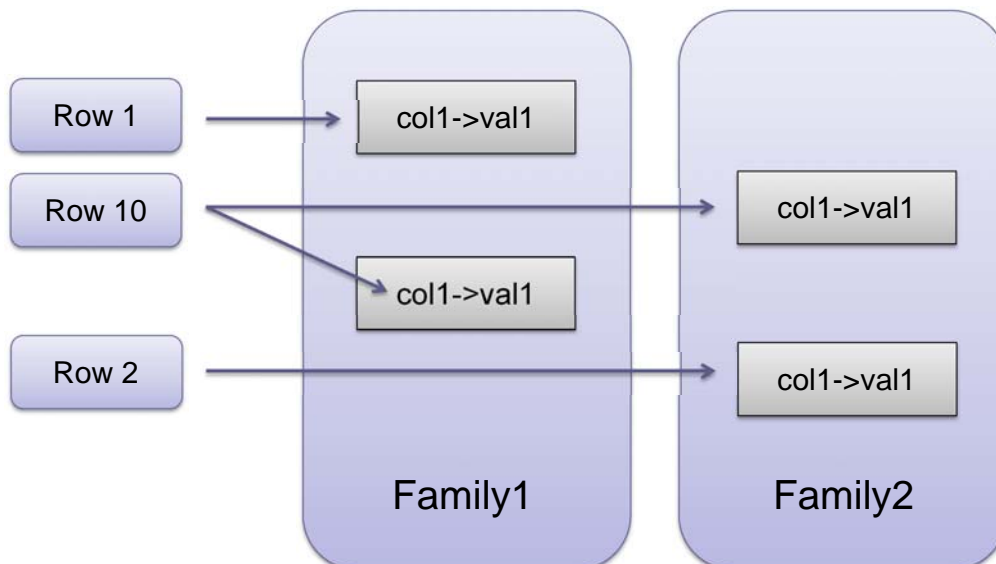
15

HBase Families

- **Family name must be composed of printable characters**
 - Not bytes, unlike keys and values
- **Think of family:column as a tag for a cell value and NOT as a spreadsheet**
- **Columns on the other hand are NOT static**
 - Create new columns at run-time
 - Can scale to millions for a family

16

Rows Composed Of Cells Stored In Families:Columns



17

HBase Timestamps

- **Cells' values are versioned**
 - For each cell multiple versions are kept
 - 3 by default
 - Another dimension to identify your data
 - Either explicitly timestamped by region server or provided by the client
 - Versions are stored in decreasing timestamp order
 - Read the latest first – optimization to read the current value
- **You can specify how many versions are kept**
 - More on this later....

18

HBase Cells

- **Value = Table+RowKey+Family+Column+Timestamp**
- **Programming language style:**

```
Table  → SortedMap<
          RowKey, List<
Family  → SortedMap<
          Column, List<
          Value, Timestamp
          >
          >
          >
          >
```

Cells

19

HBase Row Keys

- **Rows are sorted lexicographically by key**
 - Compared on a binary level from left to right
 - For example keys 1,2,3,10,15 will get sorted as
 - 1, 10, 15, 2, 3
- **Somewhat similar to Relational DB primary index**
 - Always unique
 - Some but minimal secondary indexes support

20

HBase Cells

An example - Logical representation of how values are stored

Row Key	Time stamp	Name Family		Address Family	
		first_name	last_name	number	address
row1	t1	<u>Bob</u>	<u>Smith</u>		
	t5			10	First Lane
	t10			30	Other Lane
	t15			<u>7</u>	<u>Last Street</u>
row2	t20	<u>Mary</u>	Tompson		
	t22			77	One Street
	t30		<u>Thompson</u>		

21

Source: Lars, George. [HBase The Definitive Guide](#). O'Reilly Media. 2011

HBase Cells

- **Can ask for**
 - Most recent value (default)
 - Specific timestamp
 - Multiple values such as range of timestamps
 - More on this later....

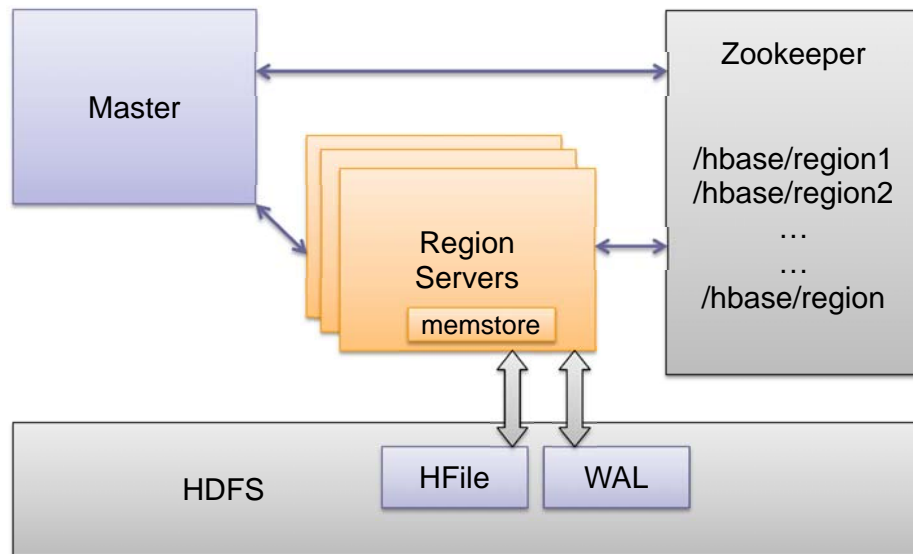
22

HBase Architecture

- **Table is made of regions**
- **Region – a range of rows stored together**
 - Single shard, used for scaling
 - Dynamically split as they become too big and merged if too small
- **Region Server- serves one or more regions**
 - A region is served by only 1 Region Server
- **Master Server – daemon responsible for managing HBase cluster, aka Region Servers**
- **HBase stores its data into HDFS**
 - relies on HDFS's high availability and fault-tolerance features
- **HBase utilizes Zookeeper for distributed coordination**

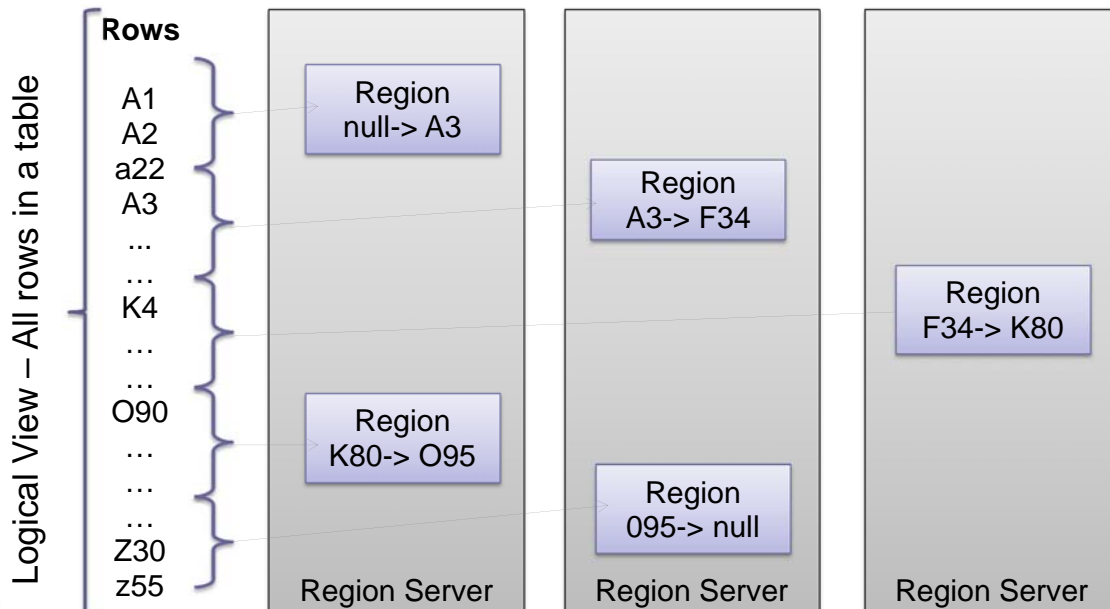
23

HBase Components



24

Rows Distribution Between Region Servers



25

HBase Regions

- **Region is a range of keys**
 - start key → stop key (ex. k3cod → odiekd)
 - start key inclusive and stop key exclusive
- **Addition of data**
 - At first there is only 1 region
 - Addition of data will eventually exceed the configured maximum → the region is split
 - Default is 256MB
 - The region is split into 2 regions at the middle key
- **Regions per server depend on hardware specs, with today's hardware it's common to have:**
 - 10 to 1000 regions per Region Server
 - Managing as much as 1GB to 2 GB per region

26

HBase Regions

- **Splitting data into regions allows**
 - Fast recovery when a region fails
 - Load balancing when a server is overloaded
 - May be moved between servers
 - Splitting is fast
 - Reads from an original file while asynchronous process performs a split
 - All of these happen automatically without user's involvement

27

Data Storage

- **Data is stored in files called HFiles/StoreFiles**
 - Usually saved in HDFS
- **HFile is basically a key-value map**
 - Keys are sorted lexicographically
- **When data is added it's written to a log called Write Ahead Log (WAL) and is also stored in memory (memstore)**
- **Flush: when in-memory data exceeds maximum value it is flushed to an HFile**
 - Data persisted to HFile can then be removed from WAL
 - Region Server continues serving read-writes during the flush operations, writing values to the WAL and memstore

28

Data Storage

- **Recall that HDFS doesn't support updates to an existing file therefore HFiles are immutable**
 - Cannot remove key-values out of HFile(s)
 - Over time more and more HFiles are created
- **Delete marker is saved to indicate that a record was removed**
 - These markers are used to filter the data - to “hide” the deleted records
 - At runtime, data is merged between the content of the HFile and WAL

29

Data Storage

- **To control the number of HFiles and to keep cluster well balanced HBase periodically performs data compactions**
 - Minor Compaction: Smaller HFiles are merged into larger HFiles (n-way merge)
 - Fast - Data is already sorted within files
 - Delete markers are not applied
 - Major Compaction:
 - For each region merges all the files within a column-family into a single file
 - Scan all the entries and apply all the deletes as necessary

30

HBase Master

- **Responsible for managing regions and their locations**
 - Assigns regions to region servers
 - Re-balanced to accommodate workloads
 - Recovers if a region server becomes unavailable
 - Uses Zookeeper – distributed coordination service
- **Doesn't actually store or read data**
 - Clients communicate directly with Region Servers
 - Usually lightly loaded
- **Responsible for schema management and changes**
 - Adding/Removing tables and column families

31

HBase and Zookeeper

- HBase uses Zookeeper extensively for region assignment



“Zookeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services” - zookeeper.apache.org

- HBase can manage Zookeeper daemons for you or you can install/manage them separately
- Learn More at <http://zookeeper.apache.org>

32

HBase and Zookeeper

- Zookeeper crash course
 - Very simple file-like API, written in Java
 - Operations on directories and files (called Znodes)
 - CRUD ZNodes and register for updates
 - Supports PERSISTENT and EPHERMAL Znodes
 - Clients connect with a session to Zookeeper
 - Session is maintained via heartbeat, if client fails to report then the session is expired and all the EPHERMAL nodes are deleted
 - Clients listening for updates will be notified of the deleted nodes as well as new nodes

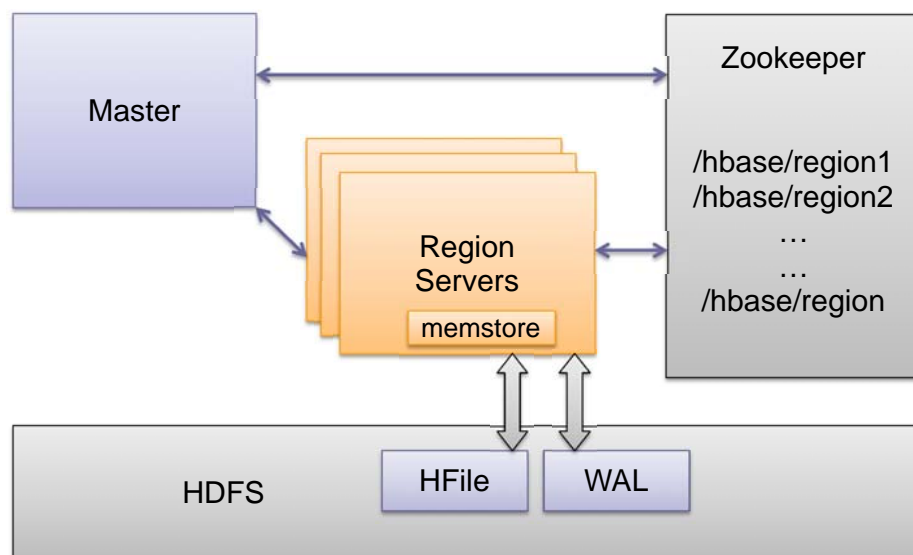
33

HBase and Zookeeper

- **Each Region Server creates an ephemeral node**
 - Master monitors these nodes to discover available region servers
 - Master also tracks these nodes for server failures
- **Uses Zookeeper to make sure that only 1 master is registered**
- **HBase cannot exist without Zookeeper**

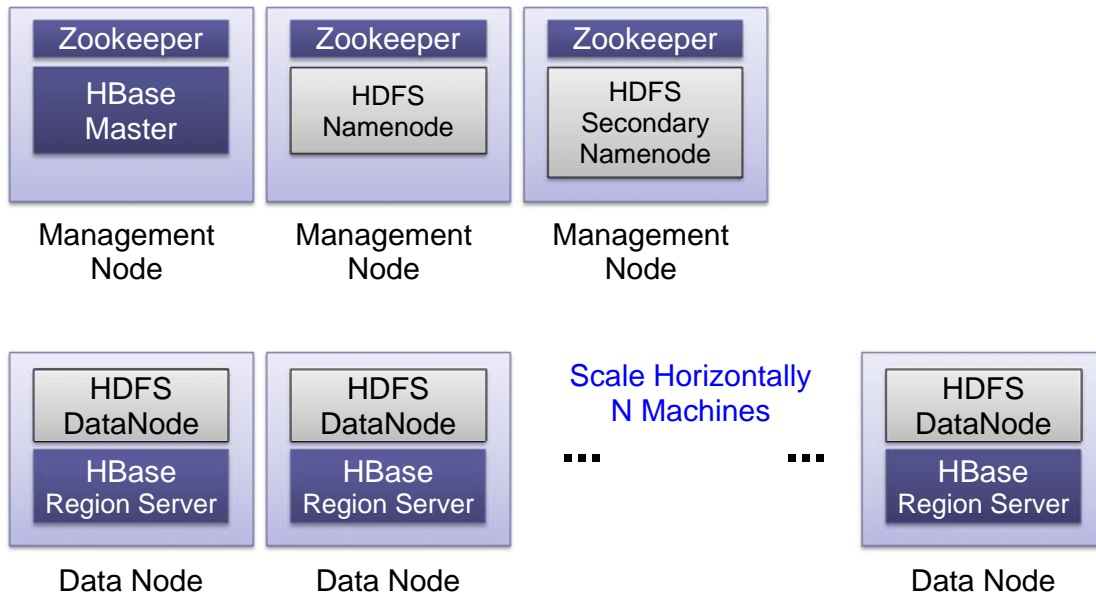
34

HBase Components



35

HBase Deployment



36

HBase Access

- **HBase Shell**
- **Native Java API**
 - Fastest and very capable options
- **Avro Server**
 - Apache Avro is also a cross-language schema compiler
 - <http://avro.apache.org>
 - Requires running Avro Server
- **HBql**
 - SQL like syntax for HBase
 - <http://www.hbql.com>

37

HBase Access

- **PyHBase**
 - python client for HBase Avro interface
 - <https://github.com/hammer/pyhbase>
- **AsyncHBase**
 - asynchronous, non-blocking, thread-safe, HBase client
 - <https://github.com/stumbleupon/asynchbase>
- **JPA/JPO access to HBase via DataNucleus**
 - <http://www.datanucleus.org>
- **HBase-DSL**
 - Java Library that helps you build queries
 - <https://github.com/nearinfinity/hbase-dsl>

38

HBase Access

- **Native API is not the only option**
 - REST Server
 - Complete client and admin APIs
 - Requires a REST gateway server
 - Supports many formats: text, xml, json, protocol buffers, raw binary
 - Thrift
 - Apache Thrift is a cross-language schema compiler
 - <http://thrift.apache.org>
 - Requires running Thrift Server

39

Resources: Books

- **HBase: The Definitive Guide by Lars George**

- Publication Date: September 20, 2011



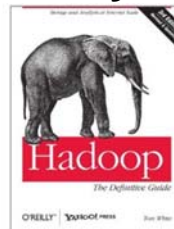
- **Apache HBase Reference Guide**

- Comes packaged with HBase
- <http://hbase.apache.org/book/book.html>



- **Hadoop: The Definitive Guide by Tom White**

- Publication Date: May 22, 2012
- Chapter about HBase



40

Resources

- **Home Page**

- <http://hbase.apache.org>

- **Mailing Lists**

- <http://hbase.apache.org/mail-lists.html>
- Subscribe to User List

- **Wiki**

- <http://wiki.apache.org/hadoop/Hbase>

- **Videos and Presentations**

- <http://hbase.apache.org/book.html#other.info>

41



Wrap-Up

Customized Java EE Training: <http://courses.coreservlets.com/>

Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **Presented**
 - HBase Overview
 - HBase Architecture
- **Learned about**
 - Data Model
 - Available Resources



Questions?

More info:

<http://www.coreservlets.com/hadoop-tutorial/> – Hadoop programming tutorial

<http://courses.coreservlets.com/hadoop-training.html> – Customized Hadoop training courses, at public venues or onsite at *your* organization

<http://courses.coreservlets.com/Course-Materials/java.html> – General Java programming tutorial

<http://www.coreservlets.com/java-8-tutorial/> – Java 8 tutorial

<http://www.coreservlets.com/JSF-Tutorial/jsf2/> – JSF 2.2 tutorial

<http://www.coreservlets.com/JSF-Tutorial/primefaces/> – PrimeFaces tutorial

<http://coreservlets.com/> – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training

Customized Java EE Training: <http://courses.coreservlets.com/>

Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.