Hibernate – fetching strategies examples

By mkyong (https://www.mkyong.com/author/mkyong/) | February 21, 2010 | Viewed : 469,375 times +1,239 pv/w

Hibernate has few fetching strategies to optimize the Hibernate generated select statement, so that it can be as efficient as possible. The fetching strategy is declared in the mapping relationship to define how Hibernate fetch its related collections and entities.

Fetching Strategies

There are four fetching strategies

1. fetch-“join” = Disable the lazy loading, always load all the collections and entities.
2. fetch-“select” (default) = Lazy load all the collections and entities.
3. batch-size=”N” = Fetching up to ‘N’ collections or entities, *Not record*.
4. fetch-“subselect” = Group its collection into a sub select statement.

For detail explanation, you can check on the Hibernate documentation (https://www.hibernate.org/315.html).

Fetching strategies examples

Here’s a “one-to-many relationship” example for the fetching strategies demonstration. A stock is belong to many stock daily records.

Example to declare fetch strategies in XML file

```
...
<hibernate-mapping>
    <class name="com.mkyong.common.Stock" table="stock">
        <set name="stockDailyRecords"  cascade="all" inverse="true"
            table="stock_daily_record" batch-size="10" fetch="select">
            <key>
                <column name="STOCK_ID" not-null="true" />
            </key>
            <one-to-many class="com.mkyong.common.StockDailyRecord" />
        </set>
    </class>
</hibernate-mapping>
```

Example to declare fetch strategies in annotation

```
...
@Entity
@Table(name = "stock", catalog = "mkyong")
public class Stock implements Serializable{
...
    @OneToMany(fetch = FetchType.LAZY, mappedBy = "stock")
    @Cascade(CascadeType.ALL)
    @Fetch(FetchMode.SELECT)
        @BatchSize(size = 10)
    public Set<StockDailyRecord> getStockDailyRecords() {
        return this.stockDailyRecords;
    }
...
}
```

Let explore how fetch strategies affect the Hibernate generated SQL statement.

1. fetch=”select” or @Fetch(FetchMode.SELECT)

This is the default fetching strategy. it enabled the lazy loading of all it’s related collections. Let see the example…

```
//call select from stock
Stock stock = (Stock)session.get(Stock.class, 114);
Set sets = stock.getStockDailyRecords();

//call select from stock_daily_record
for ( Iterator iter = sets.iterator();iter.hasNext(); ) {
      StockDailyRecord sdr = (StockDailyRecord) iter.next();
      System.out.println(sdr.getDailyRecordId());
      System.out.println(sdr.getDate());
}
```

Output

```
Hibernate:
    select ...from mkyong.stock
    where stock0_.STOCK_ID=?

Hibernate:
    select ...from mkyong.stock_daily_record
    where stockdaily0_.STOCK_ID=?
```

Hibernate generated two select statements

1. Select statement to retrieve the Stock records –**session.get(Stock.class, 114)**
2. Select its related collections – **sets.iterator()**

2. fetch=”join” or @Fetch(FetchMode.JOIN)

The “join” fetching strategy will disabled the lazy loading of all it’s related collections. Let see the example…

```
//call select from stock and stock_daily_record
Stock stock = (Stock)session.get(Stock.class, 114);
Set sets = stock.getStockDailyRecords();

//no extra select
for ( Iterator iter = sets.iterator();iter.hasNext(); ) {
      StockDailyRecord sdr = (StockDailyRecord) iter.next();
      System.out.println(sdr.getDailyRecordId());
      System.out.println(sdr.getDate());
}
```

Output

```
Hibernate:
    select ...
    from
        mkyong.stock stock0_
    left outer join
        mkyong.stock_daily_record stockdaily1_
            on stock0_.STOCK_ID=stockdaily1_.STOCK_ID
    where
        stock0_.STOCK_ID=?
```

Hibernate generated only one select statement, it retrieve all its related collections when the Stock is initialized. –**session.get(Stock.class, 114)**

1. Select statement to retrieve the Stock records and outer join its related collections.

3. batch-size=”10″ or @BatchSize(size = 10)

This ‘batch size’ fetching strategy is always misunderstanding by many Hibernate developers. Let see the *misunderstand* concept here…

```
Stock stock = (Stock)session.get(Stock.class, 114);
Set sets = stock.getStockDailyRecords();

for ( Iterator iter = sets.iterator();iter.hasNext(); ) {
      StockDailyRecord sdr = (StockDailyRecord) iter.next();
      System.out.println(sdr.getDailyRecordId());
      System.out.println(sdr.getDate());
}
```

What is your expected result, is this per-fetch 10 records from collection? See the output
Output

```
Hibernate:
    select ...from mkyong.stock
    where stock0_.STOCK_ID=?

Hibernate:
    select ...from mkyong.stock_daily_record
    where stockdaily0_.STOCK_ID=?
```

The batch-size did nothing here, it is not how batch-size work. See this statement.

> The batch-size fetching strategy is not define how many records inside in the collections are loaded. Instead, it defines how many collections should be loaded.
>
> — Repeat N times until you remember this statement —

Another example

Let see another example, you want to print out all the stock records and its related stock daily records (collections) one by one.

```
List<Stock> list = session.createQuery("from Stock").list();

for(Stock stock : list){

    Set sets = stock.getStockDailyRecords();

    for ( Iterator iter = sets.iterator();iter.hasNext(); ) {
            StockDailyRecord sdr = (StockDailyRecord) iter.next();
            System.out.println(sdr.getDailyRecordId());
            System.out.println(sdr.getDate());
    }
}
```

No batch-size fetching strategy

Output

```
Hibernate:
    select ...
    from mkyong.stock stock0_

Hibernate:
    select ...
    from mkyong.stock_daily_record stockdaily0_
    where stockdaily0_.STOCK_ID=?

Hibernate:
    select ...
    from mkyong.stock_daily_record stockdaily0_
    where stockdaily0_.STOCK_ID=?

Keep repeat the select statements....depend how many stock records in your table.
```

If you have 20 stock records in the database, the Hibernate's default fetching strategies will generate 20+1 select statements and hit the database.

1. Select statement to retrieve all the Stock records.
2. Select its related collection
3. Select its related collection
4. Select its related collection
….
21. Select its related collection

The generated queries are not efficient and caused a serious performance issue.

Enabled the batch-size='10' fetching strategy

Let see another example with batch-size='10' is enabled.
Output

```
Hibernate:
    select ...
    from mkyong.stock stock0_

Hibernate:
    select ...
    from mkyong.stock_daily_record stockdaily0_
    where
        stockdaily0_.STOCK_ID in (
            ?, ?, ?, ?, ?, ?, ?, ?, ?, ?
        )
```

Now, Hibernate will per-fetch the collections, with a select *in* statement. If you have 20 stock records, it will generate 3 select statements.

1. Select statement to retrieve all the Stock records.
2. Select In statement to per-fetch its related collections (10 collections a time)
3. Select In statement to per-fetch its related collections (next 10 collections a time)

With batch-size enabled, it simplify the select statements from 21 select statements to 3 select statements.

4. fetch="subselect" or @Fetch(FetchMode.SUBSELECT)

This fetching strategy is enable all its related collection in a sub select statement. Let see the same query again..

```
List<Stock> list = session.createQuery("from Stock").list();

for(Stock stock : list){

    Set sets = stock.getStockDailyRecords();

    for ( Iterator iter = sets.iterator();iter.hasNext(); ) {
            StockDailyRecord sdr = (StockDailyRecord) iter.next();
            System.out.println(sdr.getDailyRecordId());
            System.out.println(sdr.getDate());
    }
}
```

Output

```
Hibernate:
    select ...
    from mkyong.stock stock0_

Hibernate:
    select ...
    from
        mkyong.stock_daily_record stockdaily0_
    where
        stockdaily0_.STOCK_ID in (
            select
                stock0_.STOCK_ID
            from
                mkyong.stock stock0_
        )
```

With "subselect" enabled, it will create two select statements.

1. Select statement to retrieve all the Stock records.
2. Select all its related collections in a sub select query.

Conclusion

The fetching strategies are highly flexible and a very important tweak to optimize the Hibernate query, but if you used it in a wrong place, it will be a total disaster.

Reference

1. http://docs.jboss.org/hibernate/core/3.3/reference/en/html/performance.html (http://docs.jboss.org/hibernate/core/3.3/reference/en/html/performance.html)
2. https://www.hibernate.org/315.html (https://www.hibernate.org/315.html)

Tags : hibernate (https://www.mkyong.com/tag/hibernate/)

Share this article on

Twitter (https://twitter.com/intent/tweet?text=Hibernate - fetching strategies examples&url=https://www.mkyong.com/hibernate/hibernate-fetching-strategies-examples/&via=mkyong) Facebook (https://www.facebook.com/sharer/sharer.php?u=https://www.mkyong.com/hibernate/hibernate-fetching-strategies-examples/) Google+ (https://plus.google.com/share?url=https://www.mkyong.com/hibernate/hibernate-fetching-strategies-examples/)

About the Author

mkyong

Founder of Mkyong.com (http://mkyong.com), love Java and open source stuff. Follow him on Twitter (https://twitter.com/mkyong), or befriend him on Facebook (http://www.facebook.com/java.tutorial) or Google Plus (https://plus.google.com/110948163568945735692?rel=author). If you like my tutorials, consider make a donation to these charities (http://www.mkyong.com/blog/donate-to-charity/).

Related Posts

284k	How to configure the C3P0 connection pool in Hibernate (/hibernate/how-to-configure-the-c3p0-connection-pool-in-hibernate/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=0)
382k	Hibernate – Many-to-Many example (Annotation) (/hibernate/hibernate-many-to-many-relationship-example-annotation/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=1)
143k	Different between cascade and inverse (/hibernate/different-between-cascade-and-inverse/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=2)
833k	Hibernate Query examples (HQL) (/hibernate/hibernate-query-examples-hql/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=3)
157k	Hibernate interceptor example - audit log (/hibernate/hibernate-interceptor-example-audit-log/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=4)
38k	java.lang.NoSuchMethodError: org.objectweb.asm.ClassWriter (/hibernate/java-lang-nosuchmethoderror-org-objectweb-asm-classwriter/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=5)
20k	java.lang.NoSuchMethodError: org.objectweb.asm.ClassWriter.(Z)V (/hibernate/java-lang-nosuchmethoderror-org-objectweb-asm-classwriter-zv/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=6)
14k	Hibernate Error : JavaReflectionManager cannot be cast to MetadataProviderInjector (/hibernate/hibernate-error-javareflectionmanager-cannot-be-cast-to-metadataproviderinjector/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=7)
55k	java.lang.ClassNotFoundException : javassist.util.proxy.MethodFilter (/hibernate/java-lang-classnotfoundexception-javassist-util-proxy-methodfilter/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=8)
10k	Hibernate - Unable to insert if column named is keyword, such as DESC (/hibernate/hibernate-unable-to-insert-if-column-named-is-keyword-such-as-desc/?utm_source=mkyong&utm_medium=author&utm_campaign=related-post&utm_content=9)

Popular Posts

538k

Java 8 forEach examples (/java8/java-8-foreach-examples/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=1)

1.2m

JSON.simple example - Read and write JSON (/java/json-simple-example-read-and-write-json/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=2)

937k

Maven Tutorial (/tutorials/maven-tutorials/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=3)

514k

How to convert InputStream to File in Java (/java/how-to-convert-inputstream-to-file-in-java/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=4)

685k

RESTful Java client with Jersey client (/webservices/jax-rs/restful-java-client-with-jersey-client/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=5)

652k

Tomcat - java.lang.OutOfMemoryError: PermGen space (/tomcat/tomcat-javalangoutofmemoryerror-permgen-space/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=6)

874k

log4j.properties example (/logging/log4j-log4j-properties-examples/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=7)

602k

How to loop / iterate a List in Java (/java/how-do-loop-iterate-a-list-in-java/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=8)

582k

Spring + JDBC example (/spring/maven-spring-jdbc-example/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=9)

832k

Jackson 2 - Convert Java Object to / from JSON (/java/jackson-2-convert-java-object-to-from-json/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=10)

564k

Connect to MySQL with JDBC driver (/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=11)

725k

Spring Auto-Wiring Beans with @Autowired annotation (/spring/spring-auto-wiring-beans-with-autowired-annotation/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=12)

582k

How to install JDK on Ubuntu (/java/how-to-install-java-jdk-on-ubuntu-linux/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=13)

583k

How to write to file in Java - FileOutputStream (/java/how-to-write-to-file-in-java-fileoutputstream-example/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=14)

654k

Spring 3 MVC and JSON example (/spring-mvc/spring-3-mvc-and-json-example/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=15)

919k

How to install Maven on Windows (/maven/how-to-install-maven-in-windows/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=16)

777k

Jersey hello world example (/webservices/jax-rs/jersey-hello-world-example/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=17)

533k

Log4j hello world example (/logging/log4j-hello-world-example/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=18)

861k

JAX-WS Hello World Example - RPC Style (/webservices/jax-ws/jax-ws-hello-world-example/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=19)

1.3m

JAX-RS Tutorial (/tutorials/jax-rs-tutorials/?utm_source=mkyong&utm_medium=author&utm_campaign=top-pv&utm_content=20)

Leave a Reply

Join the discussion

Sort by: newest | oldest | most voted

himanshu prasad

Guest

nice explanation.

0 REPLY 1 year 1 month ago

Kisna

Guest

There needs some mentioning about the duplicates that result in different types of collection fetching, for example

@Fetch(FetchMode.SUBSELECT) and SELECT will eliminate duplicates whereas join (which is optimal) does not, you end up using distinct then.

0 REPLY 1 year 10 months ago

sk

Guest

very simple explanation, you are helping me a lot, thanks

0 REPLY 2 years 2 months ago

Tim Bain

Guest

Your explanation of JOIN and SELECT semantics is pretty misleading, since the concept of fetch mode is orthogonal to whether you're eagerly or lazily loading an associated collection. SELECT fetch semantics can be used for either eager or lazy loading (though JOIN can only be used for eager loading), so a better statement of your first two bullets would be: 1. fetch-"join" = Load all the collections and entities in the same query as the parent entity. Note that this can only be used with eager loading. 2. fetch-"select" (default) = Load all the collections and entities in a separate… Read more »

0 REPLY 2 years 4 months ago

Vaibhav

Guest

Thanks. The article is really helpful indeed. Helped me to reduce the number of selects hibernate was performing for joins.

0 REPLY 2 years 9 months ago

Alexandr Melnichnikov

Guest

Thank u, my friend that u and your site exists!! It's very,very,very usefull. For my russian mind that's the best explanation.

0 REPLY 3 years 1 month ago

Timon

Guest

Hello, it would be very helpfull to compare what would happen in each strategies if we want to retrieve all stocks. However its really great exmplanation

0 REPLY 3 years 1 month ago

Alen Jain

Guest

hat's off to you. Specially for demonstrating whats wrong is done by programmers and just after that how it is done in correct manner..

0 REPLY 3 years 2 months ago

Anand

Guest

I used this

@OneToMany(.. fetch=FetchType.LAZY)
@Fetch(FetchMode.JOIN)

I found that, because of FetchMode.JOIN, only one query is fired to load the data and LAZY loading is of no effect here even though I have used it using FetchType.LAZY.

Could you please explain me this behavior?

0 REPLY 3 years 5 months ago

Aravind

Guest

A very nice post. But I'm still trying to make it work.
I'm trying to use @Fetch(FetchMode.SUBSELECT) on OneToMany mapped set collection. But when I run a test program it still generates many queries. What could be the reason?

0 REPLY 3 years 9 months ago

hibernate-lazy-loading-eager-loading | mauroprogram's Blog (http://mauroprogramsite.wordpress.com/2013/09/24/hibernate-lazy-loading-eager-loading/)

[…] http://www.mkyong.com/hibernate/hibernate-fetching-strategies-examples/ (http://www.mkyong.com/hibernate/hibernate-fetching-strategies-examples/) […]

0 REPLY 3 years 9 months ago

Jitendra (http://jitendra.singh12@gmail.com)

Guest

Nice Explanation.

I have a question here with some different situation here.

Suppose i a have parent object/entity as a Stock and its child collection as a StockDailyRecord. In StockDailyRecord i have a property like activeRecord (Value could be Y/N). Now i want to perform an eager loading for parent.child collection's which are active(setted Y).

Stock.StockDailyRecord.activeRecord = 'Y'.

0 REPLY 3 years 9 months ago

krishna

Guest

Sorry, I forget to add how I am mapping..

I am doing one-to-many mapping by using fetching strategy as "select" and I got the following error while trying to get the mapped object.

org.hibernate.LazyInitializationException: failed to lazily initialize a collection of role: could not initialize proxy – no Session

0 REPLY 3 years 10 months ago

krishna

Guest

I am using this but lazy load is not working. Do I need to do any more configurations to make lazy load workable.?

Thanks,
Krishna

0 REPLY 3 years 10 months ago

Diyyana Kishore Babu

Guest

@ Diyyana Kishore Babu.
Really very good explanation, but one small correction for ?fetch=?select?? strategy.
You mentioned as
?Hibernate generated two select statements
1. Select statement to retrieve the Stock records -session.get(Stock.class, 114)
2. Select its related collections ? sets.iterator()?.

Actually it generate 1(stock) + n(stock_daily_record) for select fetch.

Let?s say stock_daily_record have 10 records it generate 10 select statements and one select statement for stock table.

So Hibernate generate 11 (1+10) select statements for ?select? fetch strategy.

0 REPLY 4 years 1 month ago

Diyyana Kishore Babu

Guest

Really very good explanation, but one small correction for ?fetch=?select?? strategy.
You mentioned as
?Hibernate generated two select statements
1. Select statement to retrieve the Stock records -session.get(Stock.class, 114)
2. Select its related collections ? sets.iterator()?.

Actually it generate 1(stock) + n(stock_daily_record) for select fetch.

Let?s say stock_daily_record have 10 records it generate 10 select statements and one select statement for stock table.

So Hibernate generate 11 (1+10) select statements for 'select' fetch strategy.

0 REPLY 4 years 1 month ago

geelong

Guest

how to define how many records inside in the collections are loaded.

0 REPLY 4 years 1 month ago

Maharsh

Guest

Excellent explanation of fetching strategy with wonderful example !! Thanks.

0 REPLY 4 years 7 months ago

prashanth

Guest

very nice examples easy to understand……..

0 REPLY 4 years 9 months ago

Manish

Guest

Thanks Mkyong for posting very nice article and in a simple way also!!

What is difference between lazy=true and fetch=select as both are responsible to trigger lazy initialization for associated collection.

0 REPLY 4 years 9 months ago

SIddhant

Guest

What is meaning of FetchMode.LAZY and FetchMode.EAGER ..?

0 REPLY 4 years 11 months ago

mperk (http://www.mehmetperk.com)

Guest

I try this example. But I have error
Exception in thread “main” org.hibernate.LazyInitializationException: could not initialize proxy – no Session

0 REPLY 5 years 1 day ago

Cristian Ortiz

Guest

Hey Guys there is a possible of doing a OneToMany relationship using criteria and pagination methods(setMaxResult and setFirstResult) and completely LAZY loading the Many relationship[IS Loaded Eagerly on Annotations] thanks a lot.

0 REPLY 5 years 10 days ago

amit

Guest

very nice article.
Thanks for posting this one.

Thanks,
Amit

0 REPLY 5 years 1 month ago

Anon

Guest

“The fetching strategies are highly flexible and a very important tweak to optimize the Hibernate query, but if you used it in a wrong place, it will be a total disaster.”

— why are we all dancing around “disaster” again? We all know how JDBC is going to behave. Is it really that hard to read a result set or use a JdbcTemplate or something simple like iBatis? What does hibernate give us that justifies flirting with disaster?

0 REPLY 5 years 1 month ago

belatrix

Guest

great article, I only have single doubt , how to restrict elements in collections directly on query level ? because the fetchmode=join causes all elements in collection to be fetched and if I apply a Restrictions on associated entities’ properties, It gives me, property not found exception

0 REPLY 5 years 1 month ago

Vijay

Guest

I have no words to thank you! Just the explanation I was looking for. Great work ..Kudos to you!

0 REPLY 5 years 2 months ago

Suresh Siva

Guest

Very sharp and crisp explanation…very neat….Thanks…

0 REPLY 5 years 2 months ago

Adarsh

Guest

It was very helpful.
Got exactly what i was looking for. GREAT

0 REPLY 5 years 3 months ago

HibLearner

Guest

Suppose I have an Order table with 50 columns and I have an Order class with 50 properties and mapped with each other in hibernate file. I just want to show order list to the user which may have at most 3 to 4 columns like order_id, date, placed_by and order_amount. My problem starts here… when I load rows through the class, all 50 columns will be loaded along with the required just 4 columns which may impact the performance significantly. And sometime I want to load one (1) order with say 20 columns and sometime want to load a… Read more »

0 REPLY 5 years 4 months ago

bekbote

Guest

```
I have tried this scenario with Criteria.setProjection(Projections.property(...))
```

0 REPLY 5 years 1 month ago

Load More Comments (https://www.mkyong.com/hibernate/hibernate-fetching-strategies-examples/#!parentId=77857)

Mkyong
53,944 likes

Like Page Sign Up

Be the first of your friends to like this

Rising Posts (100k-500k pv)

+7.3k	Java 8 Streams filter examples (/java8/java-8-streams-filter-examples/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=0)
+6.3k	Java - Convert String to int (/java/java-convert-string-to-int/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=1)
+5.2k	Java 8 Lambda : Comparator example (/java8/java-8-lambda-comparator-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=2)
+4.9k	Java enum example (/java/java-enum-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=3)
+3.6k	How to set java_home on Windows 10? (/java/how-to-set-java_home-on-windows-10/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=4)
+3.2k	Java 8 – Stream Collectors groupingBy examples (/java8/java-8-collectors-groupingby-and-mapping-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=5)
+3.1k	How to access JSON object in JavaScript (/javascript/how-to-access-json-object-in-javascript/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=6)
+2.9k	Spring @PropertySource example (/spring/spring-propertysources-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=7)
+2.6k	Spring @Value default value (/spring3/spring-value-default-value/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=8)
+2.6k	Java 8 - Filter a Map examples (/java8/java-8-filter-a-map-examples/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=9)
+2.5k	RESTful Java client with Apache HttpClient (/webservices/jax-rs/restful-java-client-with-apache-httpclient/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=100k-500k&utm_content=10)

Rising Posts (10k-99k pv)

+55.4k	Spring Boot + Spring Data JPA + Oracle example (/spring-boot/spring-boot-spring-data-jpa-oracle-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=0)
+22.7k	Spring Boot + Spring Data MongoDB example (/spring-boot/spring-boot-spring-data-mongodb-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=1)
+17.4k	Spring Boot + Spring Security + Thymeleaf example (/spring-boot/spring-boot-spring-security-thymeleaf-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=2)
+15.5k	Spring Boot Profiles example (/spring-boot/spring-boot-profiles-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=3)
+10.5k	Spring Boot - Show Hibernate SQL query (/spring-boot/spring-boot-show-hibernate-sql-query/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=4)
+10.2k	Spring Data - Add custom method to Repository (/spring-data/spring-data-add-custom-method-to-repository/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=5)

+6.3k	Spring Boot Tutorials (/tutorials/spring-boot-tutorials/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=6)
+4.8k	Java 8 Streams map() examples (/java8/java-8-streams-map-examples/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=7)
+4.7k	Apache POI – Reading and Writing Excel file in Java (/java/apache-poi-reading-and-writing-excel-file-in-java/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=8)
+4.5k	Spring Boot Hello World Example - JSP (/spring-boot/spring-boot-hello-world-example-jsp/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=9)
+4k	Java 8 Tutorials (/tutorials/java-8-tutorials/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=10k-100k&utm_content=10)

Rising Posts (<10k pv)

+7.3k	Spring Boot Test - How to stop DEBUG logs (/spring-boot/spring-boot-test-how-to-stop-debug-logs/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=0)
+6.4k	Apache Solr Hello World Example (/solr/apache-solr-hello-world-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=1)
+6k	JPA optimistic lock exception in Java Development (/jpa/jpa-optimistic-lock-exception-in-java-development/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=2)
+4.6k	Java Swing - JOptionPane showInputDialog example (/swing/java-swing-joptionpane-showinputdialog-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=3)
+4.2k	Java Swing - JFileChooser example (/swing/java-swing-jfilechooser-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=4)
+3.1k	JavaFX Animated Ball Example (/javafx/javafx-animated-ball-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=5)
+2.1k	Java Swing - JOptionPane showOptionDialog example (/swing/java-swing-joptionpane-showoptiondialog-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=6)
+1.6k	Java Swing – Keep dialog window up (/swing/java-swing-keep-dialog-window-up/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=7)
+889	JaCoCo Java Code Coverage + Maven example (/maven/jacoco-java-code-coverage-maven-example/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=8)
+777	Java AWT - Drawing rectangle, line and circle (/awt/java-awt-drawing-rectangle-line-and-circle/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=9)
+736	How to tell Maven to use Java 8 (/maven/how-to-tell-maven-to-use-java-8/?utm_source=mkyong&utm_medium=sidebar&utm_campaign=0-10k&utm_content=10)

Favorites Links

Android Getting Started (http://developer.android.com/training/index.html)
Google App Engine – Java (https://cloud.google.com/appengine/docs/java/)
Spring 2.5.x Documentation (http://docs.spring.io/spring/docs/2.5.x/reference/index.html)
Spring 3.2.x Documentation (http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/)
Spring 4.1.x Documentation (http://docs.spring.io/spring/docs/4.1.x/spring-framework-reference/html/)
Java EE 5 Tutorial (http://docs.oracle.com/javaee/5/tutorial/doc/docinfo.html)
Java EE 6 Tutorial (http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html)
Java EE 7 Tutorial (https://docs.oracle.com/javaee/7/tutorial/index.html)
Java 6 API (http://docs.oracle.com/javase/6/docs/api/overview-summary.html)
Java 7 API (http://docs.oracle.com/javase/7/docs/api/overview-summary.html)
Java 8 API (http://docs.oracle.com/javase/8/docs/api/overview-summary.html)
Oracle J2SE Tutorials (http://docs.oracle.com/javase/tutorial/index.html)
JSF Home Page (https://javaserverfaces.java.net/)
JSP Home Page (https://jsp.java.net/)
Maven Central Repository (http://search.maven.org/)
Gradle User Guide (https://docs.gradle.org/current/userguide/userguide.html)
Hibernate ORM (http://hibernate.org/orm/)
JAX-WS Home Page (https://jax-ws.java.net/)
JAX-RS Home Page (Jersey) (https://jax-ws.java.net/)
Tomcat 8 Documentation (http://tomcat.apache.org/tomcat-8.0-doc/index.html)

Partners

JAX London (https://jaxlondon.com/?utm_source=mkyong.com&utm_medium=referral&utm_campaign=mediapartner)
Java Code Geeks (http://www.javacodegeeks.com/)