



Techknow Heights - tkhts.com

[« \(hibernate-criteria-api.jsp\)](#)[Index \(index.jsp\)](#)[Examples \(examples.jsp\)](#)[Videos \(videos.jsp\)](#)[» \(hibernate-caching-strategy.jsp\)](#)

Hibernate Fetching Strategies

DESCRIPTION

- Hibernate uses a fetching strategy to retrieve associated objects if the application needs to navigate the association.
- Fetch strategies can be declared in the O/R mapping metadata, or over-ridden by a particular HQL or Criteria query.

There are four fetching strategies in hibernate.

- **fetch-"select" (default)** - Lazy load all the collections and entities.
- **fetch-"join"** - Disable the lazy loading, always load all the collections and entities.
- **batch-size="N"** - Fetching up to 'N' collections or entities, Not record.
- **fetch-"subselect"** = Group its collection into a sub select statement.

Example To Show Fetch-SELECT Strategies With Annotation

DESCRIPTION

Student.java File

```
view plain copy to clipboard print ?
01. @Entity
02. @Table(name="student")
03. public class Student {
04.     @Id
05.     @GeneratedValue
06.     private int id;
07.     private String studentName;
08.     @OneToMany(mappedBy="student",
```

```

09.         cascade=CascadeType.ALL)
10.     //This annotation is use in one to many relationship
11.
12.     @Fetch(FetchMode.SELECT)
13.     //this annotation declaring fetch mode to select
14.
15.     @BatchSize(size=2) //this define a batch size
16.     private List<Address> address =
17.         new ArrayList<Address>();
18.
19.     public List<Address> getAddress() {
20.         return address;
21.     }
22.     public void setAddress(List<Address> address) {
23.         this.address = address;
24.     }
25.     public String getStudentName() {
26.         return studentName;
27.     }
28.     public void setStudentName(String studentName) {
29.
30.         this.studentName = studentName;
31.     }
32.     public int getId() {
33.         return id;
34.     }
35.     public void setId(int id) {
36.         this.id = id;
37.     }
38. }

```

Address.java File

[view plain](#) [copy to clipboard](#) [print](#) [?](#)

```

01. @Entity
02. public class Address {
03.     @Id
04.     @GeneratedValue
05.     private int Id;
06.     private String city;
07.     private String state;
08.     @ManyToOne //this annotation is use
09.         // in many to one relationship
10.     private Student student;
11.
12.     public Student getStudent() {
13.         return student;
14.     }
15.     public void setStudent(Student student) {
16.         this.student = student;
17.     }
18.     public int getId() {
19.         return Id;
20.     }
21.     public void setId(int id) {
22.         Id = id;
23.     }
24.     public String getCity() {
25.         return city;
26.     }
27.     public void setCity(String city) {
28.         this.city = city;
29.     }

```

```

30.     public String getState() {
31.         return state;
32.     }
33.     public void setState(String state) {
34.         this.state = state;
35.     }
36. }

```

Test.java File

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. public class Test {
02.     public static void main(String[] args)
03.     {
04.
05.         Session session = sessionFactory.openSession();
06.         session.beginTransaction();
07.         Student student=(Student)
08.             session.get(Student.class,1);
09.
10.         System.out.println("Retrieving Data");
11.         System.out.println( student);
12.         System.out.println("Address retrieval initiated");
13.         List<Address> addresses = student.getAddress();
14.         System.out.println("Address retrieval complete");
15.         int i=1;
16.         for (Address address : addresses)
17.         {
18.             System.out.println
19.                 ("#### individual address: " + i);
20.             System.out.println(address);
21.             System.out.println("#### Address complete");
22.             i++;
23.         }
24.         session.getTransaction().commit();
25.         session.close();
26.     }
27. }

```

Output On Console

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. SLF4J: Failed to load class
02.     "org.slf4j.impl.StaticLoggerBinder".
03. SLF4J: Defaulting to no-operation
04.     (NOP) logger implementation
05. SLF4J: See
06.     http://www.slf4j.org/codes.html#StaticLoggerBinder
07.     for further details.
08. Hibernate: select student0_.id as
09.     id0_0_, student0_.studentName
10. as studentN2_0_0_ from student
11.     student0_ where student0_.id=?
12. Retrieving Data
13. com.tkhts.Student@e43c6f
14. Address retrieval initiated.
15. Address retrieval complete.
16. Hibernate: select address0_.student_id as
17.     student4_0_1_, address0_.Id
18. as Id1_, address0_.Id as Id1_0_,
19.     address0_.city as city1_0_,
20.     address0_.state as state1_0_,

```

```

21.         address0_.student_id as
22.         student4_1_0_ from Address address0_ where
23.         address0_.student_id=?
24. ##### individual address: 1
25. com.tkhts.Address@1981537
26. ##### Address complete
27. ##### individual address: 2
28. com.tkhts.Address@198d2cc
29. ##### Address complete

```

Example To Show Fetch-JOIN Strategies With Annotation

DESCRIPTION

Student.java File

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. @Entity
02. @Table(name="student")
03. public class Student {
04.     @Id
05.     @GeneratedValue
06.     private int id;
07.     private String studentName;
08.     @OneToMany(mappedBy="student",
09.         cascade=CascadeType.ALL)
10.     //This annotation is use in one to many relationship
11.
12.     @Fetch(FetchMode.JOIN)
13.         //this annotation declaring
14.         //fetch mode to JOIN
15.     @BatchSize(size=2) //this define a batch size
16.     private List<Address> address =
17.         new ArrayList<Address>();
18.
19.     public List<Address> getAddress() {
20.         return address;
21.     }
22.     public void setAddress(List<Address> address) {
23.         this.address = address;
24.     }
25.     public String getStudentName() {
26.         return studentName;
27.     }
28.     public void setStudentName(String studentName) {
29.
30.         this.studentName = studentName;
31.     }
32.     public int getId() {
33.         return id;
34.     }
35.     public void setId(int id) {
36.         this.id = id;
37.     }
38. }

```

Test.java File

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. public class Test {
02.     public static void main(String[] args)
03.     {
04.
05.         Session session = sessionFactory.openSession();
06.         session.beginTransaction();
07.         Student student=(Student)
08.             session.get(Student.class,1);
09.
10.         System.out.println("Retrieving Data");
11.         System.out.println( student);
12.         System.out.println("Address retrieval initiated");
13.         List<Address> addresses = student.getAddress();
14.         System.out.println("Address retrieval complete");
15.         int i=1;
16.         for (Address address : addresses)
17.         {
18.             System.out.println
19.                 ("#### individual address: " + i);
20.             System.out.println(address);
21.             System.out.println("#### Address complete");
22.             i++;
23.         }
24.         session.getTransaction().commit();
25.         session.close();
26.     }
27. }

```

Output On Console

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. SLF4J: Failed to load class
02.     "org.slf4j.impl.StaticLoggerBinder".
03. SLF4J: Defaulting to no-operation
04.     (NOP) logger implementation
05. SLF4J: See
06.     http://www.slf4j.org/codes.html#StaticLoggerBinder
07.         for further details.
08. Hibernate: select student0_.id as id0_1_,
09.             student0_.studentName
10.             as studentN2_0_1_, address1_.student_id
11.             as student4_0_3_,
12.             address1_.Id as Id3_, address1_.Id as Id1_0_,
13.             address1_.city
14.             as city1_0_, address1_.state as state1_0_,
15.             address1_.student_id
16.             as student4_1_0_ from student
17.             student0_ left outer join
18.             Address address1_ on
19.             student0_.id=address1_.student_id
20.             where student0_.id=?
21. Retrieving Data
22. com.tkhts.Student@1eb6e46
23. Address retrieval initiated.
24. Address retrieval complete.
25. #### individual address: 1
26. com.tkhts.Address@209b8c
27. #### Address complete
28. #### individual address: 2
29. com.tkhts.Address@26b13c
30. #### Address complete

```

Example To Show batch-size="2" Strategies With Annotation

DESCRIPTION

Student.java File

[view plain](#) [copy to clipboard](#) [print](#) ?

```
01. @Entity
02. @Table(name="student")
03. public class Student {
04.     @Id
05.     @GeneratedValue
06.     private int id;
07.     private String studentName;
08.     @OneToMany(mappedBy="student",
09.         cascade=CascadeType.ALL)
10.     //This annotation is use in one to many relationship
11.
12.     @BatchSize(size=2) //this define a batch size
13.     private List<Address> address =
14.         new ArrayList<Address>();
15.
16.     public List<Address> getAddress() {
17.         return address;
18.     }
19.     public void setAddress(List<Address> address) {
20.         this.address = address;
21.     }
22.     public String getStudentName() {
23.         return studentName;
24.     }
25.     public void setStudentName(String studentName) {
26.
27.         this.studentName = studentName;
28.     }
29.     public int getId() {
30.         return id;
31.     }
32.     public void setId(int id) {
33.         this.id = id;
34.     }
35. }
```

Test.java File

[view plain](#) [copy to clipboard](#) [print](#) ?

```
01. public class Test {
02.     public static void main(String[] args)
03.     {
04.
05.         Session session = sessionFactory.openSession();
06.         session.beginTransaction();
07.         Student student=(Student)
08.             session.get(Student.class,1);
09.
10.         System.out.println("Retrieving Data");
11.         System.out.println( student);
12.     }
13. }
```

```

12.         System.out.println("Address retrieval initiated");
13.         List<Address> addresses = student.getAddress();
14.         System.out.println("Address retrieval complete");
15.         int i=1;
16.         for (Address address : addresses)
17.         {
18.             System.out.println
19.                 ("#### individual address: " + i);
20.             System.out.println(address);
21.             System.out.println("#### Address complete");
22.             i++;
23.         }
24.         session.getTransaction().commit();
25.         session.close();
26.     }
27. }

```

Output On Console

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. SLF4J: Failed to load class
02.     "org.slf4j.impl.StaticLoggerBinder".
03. SLF4J: Defaulting to no-operation
04.     (NOP) logger implementation
05. SLF4J: See
06.     http://www.slf4j.org/codes.html#StaticLoggerBinder
07.     for further details.
08. Hibernate: select student0_.id as
09.     id0_0_, student0_.studentName
10.     as studentN2_0_0_ from student
11.     student0_ where student0_.id=?
12. Retrieving Data
13. com.tkhts.Student@1d69131
14. Address retrieval initiated.
15. Address retrieval complete.
16. Hibernate: select address0_.student_id
17.     as student4_0_1_, address0_.Id
18.     as Id1_, address0_.Id as Id1_0_,
19.     address0_.city as city1_0_,
20.     address0_.state as state1_0_, address0_.student_id as
21.     student4_1_0_ from Address address0_
22.     where address0_.student_id=?
23. #### individual address: 1
24. com.tkhts.Address@743d2c
25. #### Address complete
26. #### individual address: 2
27. com.tkhts.Address@15c1703
28. #### Address complete

```

Note : In above example of batch-size fetch strategy the 'batch-size=2' indicate that per-fetch 2 records from collection. 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.

Example To Show Fetch-SUBSELECT Strategies With Annotation

DESCRIPTION

Student.java File

view plain copy to clipboard print ?

```

01. @Entity
02. @Table(name="student")
03. public class Student {
04.     @Id
05.     @GeneratedValue
06.     private int id;
07.     private String studentName;
08.     @OneToMany(mappedBy="student",
09.         cascade=CascadeType.ALL)
10.     //This annotation is use in one to many relationship
11.
12.     @Fetch(FetchMode.SUBSELECT)
13.     //this annotation declaring fetch mode to JOIN
14.
15.     @BatchSize(size=2) //this define a batch size
16.     private List<Address> address =
17.     new ArrayList<Address>();
18.
19.     public List<Address> getAddress() {
20.         return address;
21.     }
22.     public void setAddress(List<Address> address) {
23.         this.address = address;
24.     }
25.     public String getStudentName() {
26.         return studentName;
27.     }
28.     public void setStudentName(String studentName) {
29.
30.         this.studentName = studentName;
31.     }
32.     public int getId() {
33.         return id;
34.     }
35.     public void setId(int id) {
36.         this.id = id;
37.     }
38. }

```

Test.java File

view plain copy to clipboard print ?

```

01. public class Test {
02.     public static void main(String[] args)
03.     {
04.
05.         Session session = sessionFactory.openSession();
06.         session.beginTransaction();
07.         Student student=(Student)
08.             session.get(Student.class,1);
09.
10.         System.out.println("Retrieving Data");
11.         System.out.println( student);
12.         System.out.println("Address retrieval initiated");
13.         List<Address> addresses = student.getAddress();
14.         System.out.println("Address retrieval complete");
15.         int i=1;

```



```

16.         for (Address address : addresses)
17.         {
18.             System.out.println
19.             ("#### individual address: " + i);
20.             System.out.println(address);
21.             System.out.println("#### Address complete");
22.             i++;
23.         }
24.         session.getTransaction().commit();
25.         session.close();
26.     }
27. }

```

Output On Console

[view plain](#) [copy to clipboard](#) [print](#) ?

```

01. SLF4J: Failed to load class
02.     "org.slf4j.impl.StaticLoggerBinder".
03. SLF4J: Defaulting to no-operation
04.     (NOP) logger implementation
05. SLF4J: See
06.     http://www.slf4j.org/codes.html#StaticLoggerBinder
07.     for further details.
08. Hibernate: select student0_.id as id0_0_,
09.             student0_.studentName
10.             as studentN2_0_0_ from student
11.             student0_ where student0_.id=?
12. Retrieving Data
13. com.tkhts.Student@100362a
14. Address retrieval initiated.
15. Address retrieval complete.
16. Hibernate: select address0_.student_id
17.             as student4_0_1_,
18.             address0_.Id as Id1_, address0_.Id
19.             as Id1_0_, address0_.city
20.             as city1_0_, address0_.state as state1_0_,
21.             address0_.student_id
22.             as student4_1_0_ from Address address0_
23.             where address0_.student_id=?
24. #### individual address: 1
25. com.tkhts.Address@236c40
26. #### Address complete
27. #### individual address: 2
28. com.tkhts.Address@1981537
29. #### Address complete

```

[« \(hibernate-criteria-api.jsp\)](#)
[Index \(index.jsp\)](#)
[Examples \(examples.jsp\)](#)
[Videos \(videos.jsp\)](#)
[» \(hibernate-caching-strategy.jsp\)](#)

Tutorials Available

[Core Java \(/core-java/introduction/index.jsp\)](#)