TKHTS.COM

Techknow Heights - tkhts.com

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.

Their 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 ?

```
@Entity
@Table(name="student")
public class Student {
    @Id
    @GeneratedValue
    private int id;
    private String studentName;
    @OneToMany(mappedBy="student",
            cascade=CascadeType.ALL)
    //This annotation is use in one to many relationship

    @Fetch(FetchMode.SELECT)
    //this annotation declaring fetch mode to select

    @BatchSize(size=2) //this define a batch size
    private List<Address> address =
            new ArrayList<Address>();

    public List<Address> getAddress() {
        return address;
    }
    public void setAddress(List<Address> address) {
        this.address = address;
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {

        this.studentName = studentName;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

Address.java File

view plain copy to clipboard print ?

```
@Entity
public class Address {
    @Id
    @GeneratedValue
    private int Id;
    private String city;
    private String state;
    @ManyToOne //this annotation is use
            // in many to one relationship
    private Student student;

    public Student getStudent() {
        return student;
    }
    public void setStudent(Student student) {
        this.student = student;
    }
    public int getId() {
        return Id;
    }
    public void setId(int id) {
        Id = id;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
}
```

Test.java File

view plain copy to clipboard print ?

```
public class Test {
    public static void main(String[] args)
     {

        Session session = sessionfactory.openSession();
        session.beginTransaction();
        Student student=(Student)
            session.get(Student.class,1);

        System.out.println("Retrieving Data");
        System.out.println( student);
        System.out.println("Address retrieval initiated");
        List<Address> addresses = student.getAddress();
        System.out.println("Address retrieval complete");
        int i=1;
        for (Address address : addresses)
        {
            System.out.println
                ("#### individual address: " + i);
            System.out.println(address);
            System.out.println("#### Address complete");
            i++;
        }
        session.getTransaction().commit();
        session.close();
    }
}
```

Output On Console

view plain copy to clipboard print ?

```
SLF4J: Failed to load class
        "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation
        (NOP) logger implementation
SLF4J: See
http://www.slf4j.org/codes.html#StaticLoggerBinder
        for further details.
Hibernate: select student0_.id as
        id0_0_, student0_.studentName
as studentN2_0_0_ from student
        student0_ where student0_.id=?
Retrieving Data
com.tkhts.Student@e43c6f
Address retrieval initiated.
Address retrieval complete.
Hibernate: select address0_.student_id as
        student4_0_1_, address0_.Id
    as Id1_, address0_.Id as Id1_0_,
        address0_.city as city1_0_,
    address0_.state as state1_0_,
        address0_.student_id as
    student4_1_0_ from Address address0_ where
        address0_.student_id=?
#### individual address: 1
com.tkhts.Address@1981537
#### Address complete
#### individual address: 2
com.tkhts.Address@198d2cc
#### Address complete
```

Example To Show Fetch-JOIN Strategies With Annotation

DESCRIPTION

Student.java File

view plain copy to clipboard print ?

```
@Entity
@Table(name="student")
public class Student {
    @Id
    @GeneratedValue
    private int id;
    private String studentName;
    @OneToMany(mappedBy="student",
        cascade=CascadeType.ALL)
    //This annotation is use in one to many relationship

    @Fetch(FetchMode.JOIN)
        //this annotation declaring
        //fetch mode to JOIN
    @BatchSize(size=2) //this define a batch size
    private List<Address> address =
        new ArrayList<Address>();

    public List<Address> getAddress() {
        return address;
    }
    public void setAddress(List<Address> address) {
        this.address = address;
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {

        this.studentName = studentName;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

Test.java File

view plain copy to clipboard print ?

```
public class Test {
    public static void main(String[] args)
     {

        Session session = sessionfactory.openSession();
        session.beginTransaction();
        Student student=(Student)
            session.get(Student.class,1);

        System.out.println("Retrieving Data");
        System.out.println( student);
        System.out.println("Address retrieval initiated");
        List<Address> addresses = student.getAddress();
        System.out.println("Address retrieval complete");
        int i=1;
        for (Address address : addresses)
        {
            System.out.println
                ("#### individual address: " + i);
            System.out.println(address);
            System.out.println("#### Address complete");
            i++;
        }
        session.getTransaction().commit();
        session.close();
    }
}
```

Output On Console

view plain copy to clipboard print ?

```
SLF4J: Failed to load class
    "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation
        (NOP) logger implementation
SLF4J: See
http://www.slf4j.org/codes.html#StaticLoggerBinder
                    for further details.
Hibernate: select student0_.id as id0_1_,
            student0_.studentName
    as studentN2_0_1_, address1_.student_id
            as student4_0_3_,
    address1_.Id as Id3_, address1_.Id as Id1_0_,
            address1_.city
    as city1_0_, address1_.state as state1_0_,
            address1_.student_id
    as student4_1_0_ from student
            student0_ left outer join
    Address address1_ on
        student0_.id=address1_.student_id
    where student0_.id=?
Retrieving Data
com.tkhts.Student@1eb6e46
Address retrieval initiated.
Address retrieval complete.
#### individual address: 1
com.tkhts.Address@209b8c
#### Address complete
#### individual address: 2
com.tkhts.Address@26b13c
#### Address complete
```

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

DESCRIPTION

Student.java File

view plain copy to clipboard print ?

```
@Entity
@Table(name="student")
public class Student {
    @Id
    @GeneratedValue
    private int id;
    private String studentName;
    @OneToMany(mappedBy="student",
        cascade=CascadeType.ALL)
    //This annotation is use in one to many relationship

    @BatchSize(size=2) //this define a batch size
    private List<Address> address =
        new ArrayList<Address>();

    public List<Address> getAddress() {
        return address;
    }
    public void setAddress(List<Address> address) {
        this.address = address;
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {

        this.studentName = studentName;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

Test.java File

view plain copy to clipboard print ?

```
public class Test {
    public static void main(String[] args)
     {

        Session session = sessionfactory.openSession();
        session.beginTransaction();
        Student student=(Student)
            session.get(Student.class,1);

        System.out.println("Retrieving Data");
        System.out.println( student);
        System.out.println("Address retrieval initiated");
        List<Address> addresses = student.getAddress();
        System.out.println("Address retrieval complete");
        int i=1;
        for (Address address : addresses)
        {
            System.out.println
                ("#### individual address: " + i);
            System.out.println(address);
            System.out.println("#### Address complete");
            i++;
        }
        session.getTransaction().commit();
        session.close();
    }
}
```

Output On Console

view plain copy to clipboard print ?

```
SLF4J: Failed to load class
    "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation
    (NOP) logger implementation
SLF4J: See
http://www.slf4j.org/codes.html#StaticLoggerBinder
    for further details.
Hibernate: select student0_.id as
    id0_0_, student0_.studentName
    as studentN2_0_0_ from student
    student0_ where student0_.id=?
Retrieving Data
com.tkhts.Student@1d69131
Address retrieval initiated.
Address retrieval complete.
Hibernate: select address0_.student_id
    as student4_0_1_, address0_.Id
    as Id1_, address0_.Id as Id1_0_,
    address0_.city as city1_0_,
    address0_.state as state1_0_, address0_.student_id as
    student4_1_0_ from Address address0_
    where address0_.student_id=?
#### individual address: 1
com.tkhts.Address@743d2c
#### Address complete
#### individual address: 2
com.tkhts.Address@15c1703
#### Address complete
```

```
Note : In above example of  batch-size fetch strategy the 'batch-size=2' indicate that p
er-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 coll
ections should be loaded.
```

Example To Show Fetch-SUBSELECT Strategies With Annotation

DESCRIPTION

Student.java File

view plain copy to clipboard print ?

```
@Entity
@Table(name="student")
public class Student {
    @Id
    @GeneratedValue
    private int id;
    private String studentName;
    @OneToMany(mappedBy="student",
        cascade=CascadeType.ALL)
    //This annotation is use in one to many relationship

    @Fetch(FetchMode.SUBSELECT)
    //this annotation declaring fetch mode to JOIN

    @BatchSize(size=2) //this define a batch size
    private List<Address> address =
    new ArrayList<Address>();

    public List<Address> getAddress() {
        return address;
    }
    public void setAddress(List<Address> address) {
        this.address = address;
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {

        this.studentName = studentName;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

Test.java File

view plain copy to clipboard print ?

```
public class Test {
    public static void main(String[] args)
     {

        Session session = sessionfactory.openSession();
        session.beginTransaction();
        Student student=(Student)
            session.get(Student.class,1);

        System.out.println("Retrieving Data");
        System.out.println( student);
        System.out.println("Address retrieval initiated");
        List<Address> addresses = student.getAddress();
        System.out.println("Address retrieval complete");
        int i=1;
        for (Address address : addresses)
        {
            System.out.println
            ("#### individual address: " + i);
            System.out.println(address);
            System.out.println("#### Address complete");
            i++;
        }
        session.getTransaction().commit();
        session.close();
    }
}
```

Output On Console

view plain copy to clipboard print ?

```
SLF4J: Failed to load class
    "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation
        (NOP) logger implementation
SLF4J: See
http://www.slf4j.org/codes.html#StaticLoggerBinder
    for further details.
Hibernate: select student0_.id as id0_0_,
        student0_.studentName
    as studentN2_0_0_ from student
    student0_ where student0_.id=?
Retrieving Data
com.tkhts.Student@100362a
Address retrieval initiated.
Address retrieval complete.
Hibernate: select address0_.student_id
    as student4_0_1_,
    address0_.Id as Id1_, address0_.Id
    as Id1_0_, address0_.city
    as city1_0_, address0_.state as state1_0_,
    address0_.student_id
    as student4_1_0_ from Address address0_
    where address0_.student_id=?
#### individual address: 1
com.tkhts.Address@236c40
#### Address complete
#### individual address: 2
com.tkhts.Address@1981537
#### 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)