Фильтры в Hibernate - XML ​​и аннотации

[Facebook](https://www.codeflow.site/#facebook)[Tumblr](https://www.codeflow.site/#tumblr)[Pinterest](https://www.codeflow.site/#pinterest)[Pocket](https://www.codeflow.site/#pocket)[Evernote](https://www.codeflow.site/#evernote)[Twitter](https://www.codeflow.site/#twitter)[Line](https://www.codeflow.site/#line)[Email](https://www.codeflow.site/#email)[Reddit](https://www.codeflow.site/#reddit)[Digg](https://www.codeflow.site/#digg)[VK](https://www.codeflow.site/#vk)[Ресурс](https://www.addtoany.com/share#url=https%3A%2F%2Fwww.codeflow.site%2Fru%2Farticle%2Fhibernate__hibernate-data-filter-example-xml-and-annotation&title=%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D1%80%20%D1%84%D0%B8%D0%BB%D1%8C%D1%82%D1%80%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20Hibernate%20-%20XML%20%E2%80%8B%E2%80%8B%D0%B8%20%D0%B0%D0%BD%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D0%B8)

Фильтр данных в спящем режиме - это инновационный способ фильтрации извлечения данных из базы данных более удобным для использования способом и правилами «видимости». Фильтр данных имеет уникальное имя, глобальный доступ и принимает параметризованное значение для правила фильтра, его можно включить и отключить в сеансе Hibernate.

Пример фильтра данных Hibernate

В этом примере он определил фильтр данных и отфильтровал данные сбора с указанной датой. Фильтр данных Hibernate может быть реализован как в файле отображения XML, так и в аннотации.

1. Фильтр данных гибернации в файле отображения XML

Определите фильтр данных с помощью ключевого слова filter **filter-def**‘и примите параметр даты.

<filter-def name="stockRecordFilter">

<filter-param name="stockRecordFilterParam" type="date"/>

</filter-def>

Пример отображения XML

Пример файла сопоставления XML для объявления и назначения его в набор сбора.

<hibernate-mapping>

<class name="com.mkyong.common.Stock" table="stock" catalog="mkyong">

...

<set name="stockDailyRecords" inverse="true" table="stock\_\_daily\_\_record">

<key>

<column name="STOCK\_\_ID" not-null="true"/>

</key>

<one-to-many class="com.mkyong.common.StockDailyRecord"/>

<filter name="stockRecordFilter" condition="date >= :stockRecordFilterParam"/>

</set>

</class>

<filter-def name="stockRecordFilter">

<filter-param name="stockRecordFilterParam" type="date"/>

</filter-def>

</hibernate-mapping>

В **condition = ”date> =: stockRecordFilterParam”**, «date» - это свойства, принадлежащие «StockDailyRecord».

**2. Фильтр данных Hibernate в аннотации**

Определите фильтр данных с помощью ключевого слова **@ FilterDef** и задайте параметр даты с помощью **@ ParamDef** :

@FilterDef(name = "stockRecordFilter",

Parameters = @ParamDef(name="stockRecordFilterParam", type="date"))

Пример файла аннотации для объявления и назначения фильтра к классу и его методу:

@Entity

**@FilterDef**(name = "stockRecordFilter",

Parameters = @ParamDef( name="stockRecordFilterParam", type="date" или “string” или .. ) )

@Table(name = "**stock**", catalog = "mkyong")

public class **Stock** implements java.io.Serializable {

...

@OneToMany(fetch = FetchType.LAZY, mappedBy = "**stock**")

**@Filter**( name = "stockRecordFilter",

condition="date >= :stockRecordFilterParam")

public Set<StockDailyRecord> getStockDailyRecords() {

return this.stockDailyRecords;

}

В **condition = ”date> =: stockRecordFilterParam”**, «date» - это свойство, принадлежащее «StockDailyRecord».

**Как включить и отключить фильтр данных**

Включить фильтр данных.

Filter filter = session.enableFilter("stockRecordFilter");

filter.setParameter("stockRecordFilterParam", new Date());

Отключить фильтр данных.

session.disableFilter ( "stockRecordFilter");

**Применение и внедрение фильтра даты**

Вот фрагмент кода, чтобы показать, как применять и реализовывать данные фильтр.

Session session = HibernateUtil.getSessionFactory().openSession();

System.out.println("\*\*\*\*\*\* Enabled Filter \*\*\*\*\*\*");

//включение фильтра (и установка его параметров), который будет применяться

//к любому запросу в данной сессии

Filter filter = session.enableFilter("stockRecordFilter");

filter.setParameter("stockRecordFilterParam", new Date());

Stock stock = (Stock)session.get(Stock.class, 2);

Set<StockDailyRecord> sets = stock.getStockDailyRecords();

for(StockDailyRecord sdr : sets){

System.out.println(sdr.getDailyRecordId());

System.out.println(sdr.getDate());

}

System.out.println("\*\*\*\*\*\* Disabled Filter \*\*\*\*\*\*");

//выключение фильтра

session.disableFilter("stockRecordFilter");

// очистка загруженного экземпляра и снова получения Stock, только для демонстрации

session.evict(stock);

Stock stock2 = (Stock)session.get(Stock.class, 2);

Set<StockDailyRecord> sets2 = stock2.getStockDailyRecords();

for(StockDailyRecord sdr : sets2){

System.out.println(sdr.getDailyRecordId());

System.out.println(sdr.getDate());

}

*Выведет*

\*\* \*\* \*\* \*\* \*\* \*\* Enabled Filter \*\* \*\* \*\* \*\* \*\* \*\*

58

2010-01-31

\*\* \*\* \*\* \*\* \*\* \*\* Disabled Filter \*\* \*\* \*\* \*\* \*\* \*\*

60

2010-01-02

58

2010-01-31

63

2010-01-23

61

2010-01-03

...

В этом примере (как XML, так и аннотации) после включения фильтра вся его коллекция StockDailyRecord фильтруется по дате вашего параметра.

*P.S filter.setParameter («stockRecordFilterParam», new Date ()); текущая новая дата - 2010-01-27.*