org.quartz

Class CronExpression

java.lang.Object
  └ **org.quartz.CronExpression**

All Implemented Interfaces:

Serializable, Cloneable

```
public final class CronExpression
extends Object
implements Serializable, Cloneable
```

Provides a parser and evaluator for unix-like cron expressions. Cron expressions provide the ability to specify complex time combinations such as "At 8:00am every Monday through Friday" or "At 1:30am every last Friday of the month".

Cron expressions are comprised of 6 required fields and one optional field separated by white space. The fields respectively are described as follows:

Field Name	Allowed Values	Allowed Special Characters
Seconds	0-59	, - * /
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month	0-11 or JAN-DEC	, - * /
Day-of-Week	1-7 or SUN-SAT	, - * ? / L #
Year (Optional)	empty, 1970-2199	, - * /

The '*' character is used to specify all values. For example, "*" in the minute field means "every minute".

The '?' character is allowed for the day-of-month and day-of-week fields. It is used to specify 'no specific value'. This is useful when you need to specify something in one of the two fields, but not the other.

The '-' character is used to specify ranges For example "10-12" in the hour field means "the hours 10, 11 and 12".

The ',' character is used to specify additional values. For example "MON,WED,FRI" in the day-of-week field means "the days Monday, Wednesday, and Friday".

The '/' character is used to specify increments. For example "0/15" in the seconds field means "the seconds 0, 15, 30, and 45". And "5/15" in the seconds field means "the seconds 5, 20, 35, and 50". Specifying '*' before the '/' is equivalent to specifying 0 is the value to start with. Essentially, for each field in the expression, there is a set of numbers that can be turned on or off. For seconds and minutes, the numbers range from 0 to 59. For hours 0 to 23, for days of the month 0 to 31, and for months 0 to 11 (JAN to DEC). The "/" character simply helps you turn on every "nth" value in the given set. Thus "7/6" in the month field only turns on month "7", it does NOT mean every 6th month, please note that subtlety.

The 'L' character is allowed for the day-of-month and day-of-week fields. This character is short-hand for "last", but it has different meaning in each of the two fields. For example, the value "L" in the day-of-month field means "the last day of the month" - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means "7" or "SAT". But if used in the day-of-week field after another value, it means "the last xxx day of the month" - for example "6L" means "the last friday of the month". You can also specify an offset from the last day of the month, such as "L-3" which would mean the third-to-last day of the calendar month. *When using the 'L' option, it is important not to specify lists, or ranges of values, as you'll get confusing/unexpected results.*

The 'W' character is allowed for the day-of-month field. This character is used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: "the nearest weekday to the 15th of the month". So if the 15th is a Saturday, the trigger will fire on Friday the 14th. If the 15th is a Sunday, the trigger will fire on Monday the 16th. If the 15th is a Tuesday, then it will fire on Tuesday the 15th. However if you specify "1W" as the value for day-of-month, and the 1st is a Saturday, the trigger will fire on Monday the 3rd, as it will not 'jump' over the boundary of a month's days. The 'W' character can only be specified when the day-of-month is a single day, not a range or list of days.

The 'L' and 'W' characters can also be combined for the day-of-month expression to yield 'LW', which translates to "last weekday of the month".

The '#' character is allowed for the day-of-week field. This character is used to specify "the nth" XXX day of the month. For example, the value of "6#3" in the day-of-week field means the third Friday of the month (day 6 = Friday and "#3" = the 3rd one in the month). Other examples: "2#1" = the first Monday of the month and "4#5" = the fifth Wednesday of the month. Note that if you specify "#5" and there is not 5 of the given day-of-week in the month, then no firing will occur that month. If the '#' character is used, there can only be one expression in the day-of-week field ("3#1,6#3" is not valid, since there are two expressions).

The legal characters and the names of months and days of the week are not case sensitive.

NOTES:

- Support for specifying both a day-of-week and a day-of-month value is not complete (you'll

need to use the '?' character in one of these fields).

- Overflowing ranges is supported - that is, having a larger number on the left hand side than the right. You might do 22-2 to catch 10 o'clock at night until 2 o'clock in the morning, or you might have NOV-FEB. It is very important to note that overuse of overflowing ranges creates ranges that don't make sense and no effort has been made to determine which interpretation CronExpression chooses. An example would be "0 0 14-6 ? * FRI-MON".

Author:

Sharada Jambula, James House, Contributions from Mads Henderson, Refactoring from CronTrigger to CronExpression by Aaron Craven

See Also:

Serialized Form

Field Summary

protected static Integer	**ALL_SPEC**
protected static int	**ALL_SPEC_INT**
protected static int	**DAY_OF_MONTH**
protected static int	**DAY_OF_WEEK**
protected static Map<String,Integer>	**dayMap**
protected TreeSet<Integer>	**daysOfMonth**
protected TreeSet<Integer>	**daysOfWeek**
protected boolean	**expressionParsed**
protected static int	**HOUR**
protected TreeSet<Integer>	**hours**
protected int	**lastdayOffset**
protected boolean	**lastdayOfMonth**
protected boolean	**lastdayOfWeek**

static int	**MAX_YEAR**
protected static int	**MINUTE**
protected TreeSet<Integer>	**minutes**
protected static int	**MONTH**
protected static Map<String,Integer>	**monthMap**
protected TreeSet<Integer>	**months**
protected boolean	**nearestWeekday**
protected static Integer	**NO_SPEC**
protected static int	**NO_SPEC_INT**
protected int	**nthdayOfWeek**
protected static int	**SECOND**
protected TreeSet<Integer>	**seconds**
protected static int	**YEAR**
protected TreeSet<Integer>	**years**

Constructor Summary

CronExpression(CronExpression expression)

Constructs a new `CronExpression` as a copy of an existing instance.

CronExpression(String cronExpression)

Constructs a new `CronExpression` based on the specified parameter.

Method Summary

protected void	**addToSet**(int val, int end, int incr, int type)
protected void	**buildExpression**(String expression)

protected int	**checkNext**(int pos, String s, int val, int type)
Object	**clone**() **Deprecated.**
protected int	**findNextWhiteSpace**(int i, String s)
String	**getCronExpression**()
protected int	**getDayOfWeekNumber**(String s)
protected String	**getExpressionSetSummary**(ArrayList<Integer> list)
protected String	**getExpressionSetSummary**(Set<Integer> set)
String	**getExpressionSummary**()
Date	**getFinalFireTime**() NOT YET IMPLEMENTED: Returns the final time that the CronExpression will match.
protected int	**getLastDayOfMonth**(int monthNum, int year)
protected int	**getMonthNumber**(String s)
Date	**getNextInvalidTimeAfter**(Date date) Returns the next date/time *after* the given date/time which does *not* satisfy the expression
Date	**getNextValidTimeAfter**(Date date) Returns the next date/time *after* the given date/time which satisfies the cron expression.
protected int	**getNumericValue**(String s, int i)
Date	**getTimeAfter**(Date afterTime)
Date	**getTimeBefore**(Date endTime) NOT YET IMPLEMENTED: Returns the time before the given time that the CronExpression matches.
TimeZone	**getTimeZone**() Returns the time zone for which this CronExpression will be resolved.
protected org.quartz.ValueSet	**getValue**(int v, String s, int i)

protected boolean	**isLeapYear**(int year)
boolean	**isSatisfiedBy**(Date date) Indicates whether the given date satisfies the cron expression.
static boolean	**isValidExpression**(String cronExpression) Indicates whether the specified cron expression can be parsed into a valid cron expression
protected void	**setCalendarHour**(Calendar cal, int hour) Advance the calendar to the particular hour paying particular attention to daylight saving problems.
void	**setTimeZone**(TimeZone timeZone) Sets the time zone for which this CronExpression will be resolved.
protected int	**skipWhiteSpace**(int i, String s)
protected int	**storeExpressionVals**(int pos, String s, int type)
String	**toString**() Returns the string representation of the CronExpression
static void	**validateExpression**(String cronExpression)

Methods inherited from class java.lang.Object
equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

SECOND

protected static final int **SECOND**

See Also:

Constant Field Values

MINUTE

protected static final int **MINUTE**

See Also:

Constant Field Values

HOUR

protected static final int **HOUR**

See Also:

Constant Field Values

DAY_OF_MONTH

protected static final int **DAY_OF_MONTH**

See Also:

Constant Field Values

MONTH

protected static final int **MONTH**

See Also:

Constant Field Values

DAY_OF_WEEK

protected static final int **DAY_OF_WEEK**

See Also:

Constant Field Values

YEAR

protected static final int **YEAR**

See Also:

Constant Field Values

ALL_SPEC_INT

protected static final int **ALL_SPEC_INT**

See Also:

Constant Field Values

NO_SPEC_INT

protected static final int **NO_SPEC_INT**

See Also:

Constant Field Values

ALL_SPEC

```
protected static final Integer ALL_SPEC
```

NO_SPEC

```
protected static final Integer NO_SPEC
```

---

monthMap

```
protected static final Map<String,Integer> monthMap
```

---

dayMap

```
protected static final Map<String,Integer> dayMap
```

---

seconds

```
protected transient TreeSet<Integer> seconds
```

---

minutes

```
protected transient TreeSet<Integer> minutes
```

---

hours

```
protected transient TreeSet<Integer> hours
```

---

daysOfMonth

```
protected transient TreeSet<Integer> daysOfMonth
```

---

months

```
protected transient TreeSet<Integer> months
```

---

daysOfWeek

protected transient TreeSet<Integer> **daysOfWeek**

---

years

protected transient TreeSet<Integer> **years**

---

lastdayOfWeek

protected transient boolean **lastdayOfWeek**

---

nthdayOfWeek

protected transient int **nthdayOfWeek**

---

lastdayOfMonth

protected transient boolean **lastdayOfMonth**

---

nearestWeekday

protected transient boolean **nearestWeekday**

---

lastdayOffset

protected transient int **lastdayOffset**

---

expressionParsed

protected transient boolean **expressionParsed**

MAX_YEAR

public static final int **MAX_YEAR**

Constructor Detail

CronExpression

public **CronExpression**(String cronExpression)
throws ParseException

Constructs a new `CronExpression` based on the specified parameter.

Parameters:

`cronExpression` - String representation of the cron expression the new object should represent

Throws:

`ParseException` - if the string expression cannot be parsed into a valid `CronExpression`

CronExpression

`public CronExpression(CronExpression expression)`

Constructs a new `CronExpression` as a copy of an existing instance.

Parameters:

`expression` - The existing cron expression to be copied

Method Detail

isSatisfiedBy

`public boolean isSatisfiedBy(Date date)`

Indicates whether the given date satisfies the cron expression. Note that milliseconds are ignored, so two Dates falling on different milliseconds of the same second will always have the same result here.

Parameters:

`date` - the date to evaluate

Returns:

a boolean indicating whether the given date satisfies the cron expression

---

getNextValidTimeAfter

`public Date getNextValidTimeAfter(Date date)`

Returns the next date/time *after* the given date/time which satisfies the cron expression.

Parameters:

`date` - the date/time at which to begin the search for the next valid date/time

Returns:

the next valid date/time

---

getNextInvalidTimeAfter

`public Date getNextInvalidTimeAfter(Date date)`

Returns the next date/time *after* the given date/time which does *not* satisfy the expression

Parameters:

`date` - the date/time at which to begin the search for the next invalid date/time

Returns:

the next valid date/time

---

getTimeZone

```
public TimeZone getTimeZone()
```

Returns the time zone for which this `CronExpression` will be resolved.

---

setTimeZone

```
public void setTimeZone(TimeZone timeZone)
```

Sets the time zone for which this `CronExpression` will be resolved.

---

toString

```
public String toString()
```

Returns the string representation of the `CronExpression`

Overrides:

`toString` in class `Object`

Returns:

a string representation of the `CronExpression`

---

isValidExpression

```
public static boolean isValidExpression(String cronExpression)
```

Indicates whether the specified cron expression can be parsed into a valid cron expression

Parameters:

`cronExpression` - the expression to evaluate

Returns:

a boolean indicating whether the given expression is a valid cron expression

---

validateExpression

```
public static void validateExpression(String cronExpression)
                               throws ParseException
```

Throws:

ParseException

---

buildExpression

```
protected void buildExpression(String expression)
                        throws ParseException
```

Throws:

ParseException

---

storeExpressionVals

```
protected int storeExpressionVals(int pos,
                                  String s,
                                  int type)
                           throws ParseException
```

Throws:

ParseException

---

checkNext

```
protected int checkNext(int pos,
                        String s,
                        int val,
                        int type)
                 throws ParseException
```

Throws:

ParseException

---

getCronExpression

```
public String getCronExpression()
```

---

getExpressionSummary

```
public String getExpressionSummary()
```

---

getExpressionSetSummary

```
protected String getExpressionSetSummary(Set<Integer> set)
```

---

getExpressionSetSummary

```
protected String getExpressionSetSummary(ArrayList<Integer> list)
```

---

skipWhiteSpace

```
protected int skipWhiteSpace(int i,
                             String s)
```

---

findNextWhiteSpace

```
protected int findNextWhiteSpace(int i,
                                 String s)
```

---

addToSet

```
protected void addToSet(int val,
                        int end,
                        int incr,
                        int type)
                 throws ParseException
```

Throws:

ParseException

---

getValue

```
protected org.quartz.ValueSet getValue(int v,
                                       String s,
                                       int i)
```

---

getNumericValue

```
protected int getNumericValue(String s,
                              int i)
```

---

getMonthNumber

```
protected int getMonthNumber(String s)
```

---

getDayOfWeekNumber

```
protected int getDayOfWeekNumber(String s)
```

---

getTimeAfter

```
public Date getTimeAfter(Date afterTime)
```

---

setCalendarHour

```
protected void setCalendarHour(Calendar cal,
                               int hour)
```

Advance the calendar to the particular hour paying particular attention to daylight saving problems.

Parameters:

`cal` - the calendar to operate on

`hour` - the hour to set

---

getTimeBefore

```
public Date getTimeBefore(Date endTime)
```

NOT YET IMPLEMENTED: Returns the time before the given time that the `CronExpression` matches.

---

getFinalFireTime

```
public Date getFinalFireTime()
```

NOT YET IMPLEMENTED: Returns the final time that the `CronExpression` will match.

---

isLeapYear

```
protected boolean isLeapYear(int year)
```

---

getLastDayOfMonth

```
protected int getLastDayOfMonth(int monthNum,
                                int year)
```

---

clone

```
@Deprecated
public Object clone()
```

Deprecated.

Overrides:

`clone` in class `Object`
