

# scheduler



## Where is the method executed?

The scheduled methods are executed in the client in which they are started. This means that if the client is closed, the scheduled method(s) will not run anymore. See [Batch Processors](#) for information how to continuously run methods in the background, in a client that will not be closed (automatically)

## Method Summary

void	<a href="#">addCronJob</a> (jobname, cronTimings, method) Adds a cron job to the scheduler.
void	<a href="#">addCronJob</a> (jobname, cronTimings, method, startDate) Adds a cron job to the scheduler.
void	<a href="#">addCronJob</a> (jobname, cronTimings, method, startDate, endDate) Adds a cron job to the scheduler.
void	<a href="#">addCronJob</a> (jobname, cronTimings, method, startDate, endDate, arguments) Adds a cron job to the scheduler.
void	<a href="#">addJob</a> (jobname, startDate, method) Adds a job to the scheduler.
void	<a href="#">addJob</a> (jobname, startDate, method, repeatInterval) Adds a job to the scheduler.
void	<a href="#">addJob</a> (jobname, startDate, method, repeatInterval, repeatCount) Adds a job to the scheduler.
void	<a href="#">addJob</a> (jobName, startDate, method, repeatInterval, repeatCount, endDate) Adds a job to the scheduler.
void	<a href="#">addJob</a> (jobname, startDate, method, repeatInterval, repeatCount, endDate, arguments) Adds a job to the scheduler.
void	<a href="#">addJob</a> (jobname, startDate, method, arguments) Adds a job to the scheduler.
<a href="#">String</a> []	<a href="#">getCurrentJobNames</a> () Returns an array with the current jobs.
<a href="#">String</a>	<a href="#">getLastRunJobName</a> () Returns the last job run from the scheduler.
<a href="#">Boolean</a>	<a href="#">removeJob</a> (jobname) Removes a job from the scheduler.

## Method Details

### addCronJob

void **addCronJob** (jobname, cronTimings, method)

Adds a cron job to the scheduler. A cron job must have at least one minute between each execution (otherwise it won't execute).

#### Parameters

[{String}](#) jobname  
[{String}](#) cronTimings  
[{Function}](#) method

#### Returns

void

#### Sample

```
// see: http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html for more info
// add a job that runs every 20 minutes after the hour (0,20,40)
plugins.scheduler.addCronJob('20mins','0 0/20 * * * ?',method)
// add a job that runs every day at 23:30 between now and 5 days from now
var dateNow = new Date();
var date5Days = new Date(dateNow.getTime()+5*24*60*60*1000);
plugins.scheduler.addCronJob('23:30','0 30 23 ? * *',method,dateNow,date5Days)
```

### addCronJob

void **addCronJob** (jobname, cronTimings, method, startDate)

Adds a cron job to the scheduler. A cron job must have at least one minute between each execution (otherwise it won't execute).

**Parameters**

```

{String} jobname
{String} cronTimings
{Function} method
{Date} startDate

```

**Returns**

```
void
```

**Sample**

```

// see: http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html for more info
// add a job that runs every 20 minutes after the hour (0,20,40)
plugins.scheduler.addCronJob('20mins','0 0/20 * * * ?',method)
// add a job that runs every day at 23:30 between now and 5 days from now
var dateNow = new Date();
var date5Days = new Date(dateNow.getTime()+5*24*60*60*1000);
plugins.scheduler.addCronJob('23:30','0 30 23 ? * *',method,dateNow,date5Days)

```

**addCronJob**

```
void addCronJob (jobname, cronTimings, method, startDate, endDate)
```

Adds a cron job to the scheduler. A cron job must have at least one minute between each execution (otherwise it won't execute).

**Parameters**

```

{String} jobname
{String} cronTimings
{Function} method
{Date} startDate
{Date} endDate

```

**Returns**

```
void
```

**Sample**

```

// see: http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html for more info
// add a job that runs every 20 minutes after the hour (0,20,40)
plugins.scheduler.addCronJob('20mins','0 0/20 * * * ?',method)
// add a job that runs every day at 23:30 between now and 5 days from now
var dateNow = new Date();
var date5Days = new Date(dateNow.getTime()+5*24*60*60*1000);
plugins.scheduler.addCronJob('23:30','0 30 23 ? * *',method,dateNow,date5Days)

```

**addCronJob**

```
void addCronJob (jobname, cronTimings, method, startDate, endDate, arguments)
```

Adds a cron job to the scheduler. A cron job must have at least one minute between each execution (otherwise it won't execute).

**Parameters**

```

{String} jobname
{String} cronTimings
{Function} method
{Date} startDate
{Date} endDate
{Object[]} arguments

```

**Returns**

```
void
```

**Sample**

```

// see: http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html for more info
// add a job that runs every 20 minutes after the hour (0,20,40)
plugins.scheduler.addCronJob('20mins','0 0/20 * * * ?',method)
// add a job that runs every day at 23:30 between now and 5 days from now
var dateNow = new Date();
var date5Days = new Date(dateNow.getTime()+5*24*60*60*1000);
plugins.scheduler.addCronJob('23:30','0 30 23 ? * *',method,dateNow,date5Days)

```

**addJob**

```
void addJob (jobname, startDate, method)
```

---

Adds a job to the scheduler.

**Parameters**

`{String}` jobname  
`{Date}` startDate  
`{Function}` method

**Returns**

void

**Sample**

```
// add a job that runs at the given date (20 seconds in the future)
// and repeats that every 20 seconds for 40 times or the enddate is reached (0 for no repeats = just one call)
var startDate = new Date();
startDate.setTime(startDate.getTime()+20000);
var endDate = new Date(startDate.getTime()+100000);
plugins.scheduler.addJob('in20seconds',startDate,method,20000,40,endDate)
```

**addJob**

void **addJob** (jobname, startDate, method, repeatInterval)

Adds a job to the scheduler.

**Parameters**

`{String}` jobname  
`{Date}` startDate  
`{Function}` method  
`{Number}` repeatInterval - ms

**Returns**

void

**Sample**

```
// add a job that runs at the given date (20 seconds in the future)
// and repeats that every 20 seconds for 40 times or the enddate is reached (0 for no repeats = just one call)
var startDate = new Date();
startDate.setTime(startDate.getTime()+20000);
var endDate = new Date(startDate.getTime()+100000);
plugins.scheduler.addJob('in20seconds',startDate,method,20000,40,endDate)
```

**addJob**

void **addJob** (jobname, startDate, method, repeatInterval, repeatCount)

Adds a job to the scheduler.

**Parameters**

`{String}` jobname  
`{Date}` startDate  
`{Function}` method  
`{Number}` repeatInterval - ms  
`{Number}` repeatCount

**Returns**

void

**Sample**

```
// add a job that runs at the given date (20 seconds in the future)
// and repeats that every 20 seconds for 40 times or the enddate is reached (0 for no repeats = just one call)
var startDate = new Date();
startDate.setTime(startDate.getTime()+20000);
var endDate = new Date(startDate.getTime()+100000);
plugins.scheduler.addJob('in20seconds',startDate,method,20000,40,endDate)
```

**addJob**

void **addJob** (jobName, startDate, method, repeatInterval, repeatCount, endDate)

Adds a job to the scheduler.

**Parameters**

```

{String} jobName
{Date} startDate
{Function} method
{Number} repeatInterval - ms
{Number} repeatCount
{Date} endDate

```

**Returns**

```
void
```

**Sample**

```

// add a job that runs at the given date (20 seconds in the future)
// and repeats that every 20 seconds for 40 times or the enddate is reached (0 for no repeats = just one call)
var startDate = new Date();
startDate.setTime(startDate.getTime()+20000);
var endDate = new Date(startDate.getTime()+100000);
plugins.scheduler.addJob('in20seconds',startDate,method,20000,40,endDate)

```

**addJob**

```
void addJob (jobname, startDate, method, repeatInterval, repeatCount, endDate, arguments)
```

Adds a job to the scheduler.

**Parameters**

```

{String} jobname
{Date} startDate
{Function} method
{Number} repeatInterval - ms
{Number} repeatCount
{Date} endDate
{Object[]} arguments

```

**Returns**

```
void
```

**Sample**

```

// add a job that runs at the given date (20 seconds in the future)
// and repeats that every 20 seconds for 40 times or the enddate is reached (0 for no repeats = just one call)
var startDate = new Date();
startDate.setTime(startDate.getTime()+20000);
var endDate = new Date(startDate.getTime()+100000);
plugins.scheduler.addJob('in20seconds',startDate,method,20000,40,endDate)

```

**addJob**

```
void addJob (jobname, startDate, method, arguments)
```

Adds a job to the scheduler.

**Parameters**

```

{String} jobname
{Date} startDate
{Function} method
{Object[]} arguments

```

**Returns**

```
void
```

**Sample**

```

// add a job that runs at the given date (20 seconds in the future)
// and repeats that every 20 seconds for 40 times or the enddate is reached (0 for no repeats = just one call)
var startDate = new Date();
startDate.setTime(startDate.getTime()+20000);
var endDate = new Date(startDate.getTime()+100000);
plugins.scheduler.addJob('in20seconds',startDate,method,20000,40,endDate)

```

---

**getCurrentJobNames****String[]** **getCurrentJobNames** ()

Returns an array with the current jobs.

**Returns****String[]****Sample**

```
plugins.scheduler.getCurrentJobNames()
```

**getLastRunJobName****String** **getLastRunJobName** ()

Returns the last job run from the scheduler.

**Returns****String****Sample**

```
plugins.scheduler.getLastRunJobName();
```

**removeJob****Boolean** **removeJob** (jobname)

Removes a job from the scheduler.

**Parameters**{**String**} jobname**Returns****Boolean****Sample**

```
// removes a job 'myjob' from the scheduler  
plugins.scheduler.removeJob('myjob');
```