
GRAILS


spring
source

gemfire - Reference Documentation

Authors:

Version: 0.1-SNAPSHOT

Table of Contents

1. Introduction To The GemFire Plugin

The GemFire plugin provides integration with the GemFire in-memory distributed data management platform.
This user guide describes details on configuring and using GemFire specifically in the context of a Grails application.
Complete documentation on GemFire is available at http://www.gemstone.com/products/gemfire.

2. Cache Regions

GemFire allows your data to be organized within a cache using data regions. The Grails GemFire plugin provides a DSL for describing the regions available to the application and provides a simple convention based approach to accessing regions.

2.1 Cache Region Configuration

GemFire regions may be described in `Config.groovy` by assigning a value to the `grails.gemfire.regions` property. The value should be a closure which contains GemFire Region DSL code. Details about the DSL are described below.
The code below declares 2 regions with the names `region1` and `region2`.

```
// grails-app/conf/Config.groovy
grails.gemfire.region = {
    // declare region1
    region1()
    // declare region2
    region2()
}
```

Regions may be configured with a syntax like this:

```
// grails-app/conf/Config.groovy
import com.gemstone.gemfire.cache.DataPolicy
grails.gemfire.region = {
    region1 {
        // configure region1…
        dataPolicy = DataPolicy.REPLICATE
        publisher = false
    }
    region2 {
        dataPolicy = DataPolicy.PARTITION
    }
}
```

The DSL supports all of the configuration attributes which are supported by the AttributesFactory.
The DSL provides some syntax to simplify the configuration and eliminate explicit references to GemFire classes like `DataPolicy`. The previous example could be written like this:

```
// grails-app/conf/Config.groovy
grails.gemfire.region = {
    region1 {
        // configure region1…
        dataPolicy = REPLICATE
        publisher = false
    }
}
```

The following table lists all of the properties which may be referenced directly without a class prefix.

Class	Property Name
com.gemstone.gemfire.cache.DataPolicy	EMPTY
com.gemstone.gemfire.cache.DataPolicy	NORMAL
com.gemstone.gemfire.cache.DataPolicy	PARTITION
com.gemstone.gemfire.cache.DataPolicy	PERSISTENT_REPLICATE
com.gemstone.gemfire.cache.DataPolicy	PRELOADED
com.gemstone.gemfire.cache.DataPolicy	REPLICATE
com.gemstone.gemfire.cache.ExpirationAction	DESTROY
com.gemstone.gemfire.cache.ExpirationAction	INVALIDATE
com.gemstone.gemfire.cache.ExpirationAction	LOCAL_DESTROY
com.gemstone.gemfire.cache.ExpirationAction	LOCAL_INVALIDATE
com.gemstone.gemfire.cache.Scope	DISTRIBUTED_ACK
com.gemstone.gemfire.cache.Scope	DISTRIBUTED_NO_ACK
com.gemstone.gemfire.cache.Scope	GLOBAL
com.gemstone.gemfire.cache.Scope	LOCAL

Several properties require an instance of the ExpirationAttributes class. These include `regionTimeToLive`, `regionIdleTimeout`, `entryTimeToLive` and `entryIdleTimeout`. Configuring those properties might look something like this:

```
// grails-app/conf/Config.groovy
import com.gemstone.gemfire.cache.ExpirationAction
import com.gemstone.gemfire.cache.ExpirationAttributes
grails.gemfire.region = {
    region1 {
        entryTimeToLive = new ExpirationAttributes(120)
        entryTimeToIdle = new ExpirationAttributes(200, ExpirationAction.DESTROY)
    }
}
```

The DSLS allows the explicit references to the `ExpirationAction` and `ExpirationAttributes` classes to be removed.

```
// grails-app/conf/Config.groovy
grails.gemfire.region = {
    region1 {
        entryTimeToLive = expirationAttributes(120)
        entryTimeToIdle = expirationAttributes(200, DESTROY)
    }
}
```