Apex Reference Guide / Cache Namespace / Partition Class

Partition Class

Base class of `Cache.OrgPartition` and `Cache.SessionPartition`. Use the subclasses to manage the cache partition for org caches and session caches.

Namespace

Cache

Cache Key Format for Partition Methods

After you obtain the partition object (an instance of `Cache.OrgPartition` or `Cache.SessionPartition`), the methods to add, retrieve, and manage the cache values in a partition take the key name. The key name that you supply to these methods (`get()`, `put()`, `remove()`, and `contains()`) doesn't include the `namespace.partition` prefix.

- Partition Methods

See Also

- OrgPartition Class
- SessionPartition Class
- *Apex Developer Guide*: Platform Cache

Partition Methods

The following are methods for `Partition`.

- **contains(key)**
  Returns `true` if the cache partition contains a cached value corresponding to the specified key.
- **contains(setOfKeys)**
  Returns `true` if the cache partition contains values for a specified set of keys.
- **createFullyQualifiedKey(namespace, partition, key)**
  Generates a fully qualified key from the passed-in key components. The format of the generated key string is `namespace.partition.key`.
- **createFullyQualifiedPartition(namespace, partition)**
  Generates a fully qualified partition name from the passed-in namespace and partition. The format of the generated partition string is `namespace.partition`.
- **get(key)**
  Returns the cached value corresponding to the specified key from the cache partition.
- **get(keys)**
  Returns the cached values corresponding to the specified set of keys from the cache partition.
- **get(cacheBuilder, key)**
  Returns the cached value corresponding to the specified key from the partition cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

- getAvgValueSize()
  Deprecated and available only in API versions 49.0 and earlier. Returns the average item size for keys in the partition, in bytes.
- getCapacity()
  Returns the percentage of cache used of the total capacity for this partition.
- getKeys()
  Returns a set of all keys that are stored in the cache partition and visible to the invoking namespace.
- getMaxGetSize()
  Returns the maximum item size of all the keys fetched from the partition, in bytes.
- getMaxGetTime()
  Returns the maximum time taken to get a key from the partition, in nanoseconds.
- getMaxValueSize()
  Deprecated and available only in API versions 49.0 and earlier. Returns the maximum item size for keys in the partition, in bytes.
- getMissRate()
  Returns the miss rate in the partition.
- getName()
  Returns the name of this cache partition.
- getNumKeys()
  Returns the total number of keys in the partition.
- isAvailable()
  Returns `true` if the Salesforce session is available. Only applies to `Cache.SessionPartition`. The session cache isn't available when an active session isn't present, such as in asynchronous Apex or code called by asynchronous Apex. For example, if batch Apex causes an Apex trigger to execute, the session cache isn't available in the trigger because the trigger runs in asynchronous context.
- put(key, value)
  Stores the specified key/value pair as a cached entry in the cache partition. The `put` method can write only to the cache in your org's namespace.
- put(key, value, visibility)
  Stores the specified key/value pair as a cached entry in the cache partition and sets the cached value's visibility.
- put(key, value, ttlSecs)
  Stores the specified key/value pair as a cached entry in the cache partition and sets the cached value's lifetime.
- put(key, value, ttlSecs, visibility, immutable)
  Stores the specified key/value pair as a cached entry in the cache partition. This method also sets the cached value's lifetime, visibility, and whether it can be overwritten by another namespace.
- remove(key)
  Deletes the cached value corresponding to the specified key from this cache partition.
- remove(cacheBuilder, key)
  Deletes the cached value corresponding to the specified key from the partition cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.
- validateCacheBuilder(cacheBuilder)
  Validates that the specified class implements the `CacheBuilder` interface.
- validateKey(isDefault, key)
  Validates a cache key. This method throws a `Cache.InvalidParamException` if the key is not valid. A valid key is not `null` and contains alphanumeric characters.
- validateKeyValue(isDefault, key, value)
  Validates a cache key and ensures that the cache value is non-null. This method throws a

the key is not valid. A valid key is not `null` and contains alphanumeric characters.

- **validatePartitionName(name)**
  Validates the partition name – for example, that it is not null.

contains(key)

Returns `true` if the cache partition contains a cached value corresponding to the specified key.

Signature

```
public Boolean contains(String key)
```

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value.

Return Value

Type: Boolean

`true` if a cache entry is found. Othewise, `false`.

contains(setOfKeys)

Returns `true` if the cache partition contains values for a specified set of keys.

Signature

```
public Map <String, Boolean> contains (Set<String> keys)
```

Parameters

setOfKeys

Type: Set <String>

A set of keys that uniquely identifies cached values. For information about the format of the key name, see Usage.

Return Value

Type: Map <String, Boolean>

Returns the cache key and corresponding Boolean value indicating that the key entry exists. The Boolean value is `false` if the key entry doesn't exist.

Usage

The number of input keys cannot exceed the maximum limit of 10.

Example

In this example, the code checks for the presence of multiple keys on a partition. It fetches the cache key and the corresponding Boolean value for the key entry from the org cache of the partition.

```
// Assuming there is a partition p1 in the default 'local' namespace

Set<String> keys = new Set<String>{'key1','key2','key3','key4','key5'};
Cache.OrgPartition orgPart = Cache.Org.getPartition('local.p1');
```

In this example, the code checks for the presence of multiple keys on a partition. It fetches the cache key and the corresponding Boolean value for the key entry from the session cache of the partition.

```
// Assuming there are three partitions p1, p2, p3 with default 'local' namespace

Set<String> keys = new Set<String>{'key1','key2','key3','key4','key5'};
Cache.SessionPartition sessionPart = Cache.Session.getPartition('local.p1');
Map<String,Boolean> result = sessionPart.contains(keys);
for(String key : result.keySet()) {
    system.debug('key: ' + key);
    system.debug('value: ' +  result.get(key));
}
```

createFullyQualifiedKey(namespace, partition, key)

Generates a fully qualified key from the passed-in key components. The format of the generated key string is `namespace.partition.key`.

Signature

`public static String createFullyQualifiedKey(String namespace, String partition, String key)`

Parameters

namespace

Type: String

The namespace of the cache key.

partition

Type: String

The partition of the cache key.

key

Type: String

The name of the cache key.

Return Value

Type: String

createFullyQualifiedPartition(namespace, partition)

Generates a fully qualified partition name from the passed-in namespace and partition. The format of the generated partition string is `namespace.partition`.

Signature

`public static String createFullyQualifiedPartition(String namespace, String partition)`

Parameters

namespace

Type: String

The namespace of the cache key.

Return Value

Type: String

get(key)

Returns the cached value corresponding to the specified key from the cache partition.

Signature

```
public Object get(String key)
```

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value.

Return Value

Type: Object

The cached value as a generic object type. Cast the returned value to the appropriate type.

get(keys)

Returns the cached values corresponding to the specified set of keys from the cache partition.

Signature

```
public Map <String, Object> get (Set <String> keys)
```

Parameters

keys

Type: Set <String>

A set of keys that uniquely identify cached values. For information about the format of the key name, see Usage.

Return Value

Type: Map <String, Object>

Returns the cache key and corresponding value. Returns null when no corresponding value is found for an input key.

Usage

The number of input keys cannot exceed the maximum limit of 10.

Examples

Fetch multiple keys from the org cache of a partition.

```
// Assuming there is a partition p1 in the default 'local' namespace

Set<String> keys = new Set<String>{'key1','key2','key3','key4','key5'};
Cache.OrgPartition orgPart = Cache.Org.getPartition('local.p1');
Map<String,Object> result = orgPart.get(keys);
for(String key : result.keySet()) {
    system.debug('key: ' + key);
```

```
// Assuming there is a partition p1 in the default 'local' namespace

Set<String> keys = new Set<String>{'key1','key2','key3','key4','key5'};
Cache.SessionPartition sessionPart = Cache.Session.getPartition('local.p1');
Map<String,Object> result = sessionPart.get(keys);
for(String key : result.keySet()) {
    system.debug('key: ' + key);
    system.debug('value: ' +  result.get(key));
}
```

get(cacheBuilder, key)

Returns the cached value corresponding to the specified key from the partition cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

Signature

`public Object get(System.Type cacheBuilder, String key)`

Parameters

cacheBuilder

Type: System.Type

The Apex class that implements the `CacheBuilder` interface.

key

Type: String

A case-sensitive string value that, combined with the class name corresponding to the *cacheBuilder* parameter, uniquely identifies a cached value.

Return Value

Type: Object

The cached value as a generic object type. Cast the returned value to the appropriate type.

getAvgGetSize()

Returns the average item size of all the keys fetched from the partition, in bytes.

Signature

`public Long getAvgGetSize()`

Return Value

Type: Long

getAvgGetTime()

Returns the average time taken to get a key from the partition, in nanoseconds.

Signature

`public Long getAvgGetTime()`

Return Value

Type: Long

Signature

```
public Long getAvgValueSize()
```

Return Value

Type: Long

getCapacity()

Returns the percentage of cache used of the total capacity for this partition.

Signature

```
public Double getCapacity()
```

Return Value

Type: Double

Used partition cache as a percentage number.

getKeys()

Returns a set of all keys that are stored in the cache partition and visible to the invoking namespace.

Signature

```
public Set<String> getKeys()
```

Return Value

Type: Set<String>

A set containing all cache keys.

getMaxGetSize()

Returns the maximum item size of all the keys fetched from the partition, in bytes.

Signature

```
public Long getMaxGetSize()
```

Return Value

Type: Long

getMaxGetTime()

Returns the maximum time taken to get a key from the partition, in nanoseconds.

Signature

```
public Long getMaxGetTime()
```

Return Value

Type: Long

getMaxValueSize()

```
public Long getMaxValueSize()
```

Return Value

Type: Long

getMissRate()

Returns the miss rate in the partition.

Signature

```
public Double getMissRate()
```

Return Value

Type: Double

getName()

Returns the name of this cache partition.

Signature

```
public String getName()
```

Return Value

Type: String

The name of this cache partition.

getNumKeys()

Returns the total number of keys in the partition.

Signature

```
public Long getNumKeys()
```

Return Value

Type: Long

isAvailable()

Returns `true` if the Salesforce session is available. Only applies to `Cache.SessionPartition`. The session cache isn’t available when an active session isn’t present, such as in asynchronous Apex or code called by asynchronous Apex. For example, if batch Apex causes an Apex trigger to execute, the session cache isn’t available in the trigger because the trigger runs in asynchronous context.

Signature

```
public Boolean isAvailable()
```

Return Value

Type: Boolean

put(key, value)

```
public void put(String key, Object value)
```

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value.

value

Type: Object

The value to store in the cache. The cached value must be serializable.

Return Value

Type: void

put(key, value, visibility)

Stores the specified key/value pair as a cached entry in the cache partition and sets the cached value’s visibility.

Signature

```
public void put(String key, Object value, cache.Visibility visibility)
```

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value.

value

Type: Object

The value to store in the cache. The cached value must be serializable.

visibility

Type: Cache.Visibility

Indicates whether the cached value is available only to Apex code that is executing in the same namespace or to Apex code executing from any namespace.

Return Value

Type: void

put(key, value, ttlSecs)

Stores the specified key/value pair as a cached entry in the cache partition and sets the cached value’s lifetime.

Signature

```
public void put(String key, Object value, Integer ttlSecs)
```

Parameters

key

Type: String

The value to store in the cache. The cached value must be serializable.

ttlSecs

Type: Integer

The amount of time, in seconds, to keep the cached value in the cache.

Return Value

Type: void

put(key, value, ttlSecs, visibility, immutable)

Stores the specified key/value pair as a cached entry in the cache partition. This method also sets the cached value’s lifetime, visibility, and whether it can be overwritten by another namespace.

Signature

```
public void put(String key, Object value, Integer ttlSecs, cache.Visibility visibility, Boolean
immutable)
```

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value.

value

Type: Object

The value to store in the cache. The cached value must be serializable.

ttlSecs

Type: Integer

The amount of time, in seconds, to keep the cached value in the cache.

visibility

Type: Cache.Visibility

Indicates whether the cached value is available only to Apex code that is executing in the same namespace or to Apex code executing from any namespace.

immutable

Type: Boolean

Indicates whether the cached value can be overwritten by another namespace (`false`) or not (`true`).

Return Value

Type: void

remove(key)

Deletes the cached value corresponding to the specified key from this cache partition.

Signature

```
public Boolean remove(String key)
```

A case-sensitive string value that uniquely identifies a cached value.

Return Value

Type: Boolean

`true` if the cache value was successfully removed. Otherwise, `false`.

remove(cacheBuilder, key)

Deletes the cached value corresponding to the specified key from the partition cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

Signature

```
public Boolean remove(System.Type cacheBuilder, String key)
```

Parameters

cacheBuilder

Type: System.Type

The Apex class that implements the `CacheBuilder` interface.

key

Type: String

A case-sensitive string value that, combined with the class name corresponding to the *cacheBuilder* parameter, uniquely identifies a cached value.

Return Value

Type: Boolean

`true` if the cache value was successfully removed. Otherwise, `false`.

validateCacheBuilder(cacheBuilder)

Validates that the specified class implements the `CacheBuilder` interface.

Signature

```
public static void validateCacheBuilder(System.Type cacheBuilder)
```

Parameters

cacheBuilder

Type: System.Type

The class to validate.

Return Value

Type: void

validateKey(isDefault, key)

Validates a cache key. This method throws a `Cache.InvalidParamException` if the key is not valid. A valid key is not `null` and contains alphanumeric characters.

Signature

```
public static void validateKey(Boolean isDefault, String key)
```

Set to `true` if the key references a default partition. Otherwise, set to `false`.

key

Type: String

The key to validate.

Return Value

Type: void

validateKeyValue(isDefault, key, value)

Validates a cache key and ensures that the cache value is non-null. This method throws a `Cache.InvalidParamException` if the key or value is not valid. A valid key is not `null` and contains alphanumeric characters.

Signature

```
public static void validateKeyValue(Boolean isDefault, String key, Object value)
```

Parameters

isDefault

Type: Boolean

Set to `true` if the key references a default partition. Otherwise, set to `false`.

key

Type: String

The key to validate.

value

Type: Object

The cache value to validate.

Return Value

Type: void

validateKeys(isDefault, keys)

Validates the specified cache keys. This method throws a `Cache.InvalidParamException` if the key is not valid. A valid key is not `null` and contains alphanumeric characters.

Signature

```
public static void validateKeys(Boolean isDefault, Set<String> keys)
```

Parameters

isDefault

Type: Boolean

Set to `true` if the key references a default partition. Otherwise, set to `false`.

keys

Type: Set<String>

A set of key string values to validate.

validatePartitionName(name)

Validates the partition name – for example, that it is not null.

Signature

```
public static void validatePartitionName(String name)
```

Parameters

name

Type: String

The name of the partition to validate.

Return Value

Type: void

DID THIS ARTICLE SOLVE YOUR ISSUE?
Let us know so we can improve!

Share your feedback

DEVELOPER CENTERS
- Heroku
- MuleSoft
- Tableau
- Commerce Cloud
- Lightning Design System
- Einstein
- Quip

POPULAR RESOURCES
- Documentation
- Component Library
- APIs
- Trailhead
- Sample Apps
- Podcasts
- AppExchange

COMMUNITY
- Trailblazer Community
- Events and Calendar
- Partner Community
- Blog
- Salesforce Admins
- Salesforce Architects

Privacy Information Terms of Service Legal Use of Cookies Trust Cookie Preferences

Your Privacy Choices Responsible Disclosure Contact