Org Class

Use the `Cache.Org` class to add, retrieve, and manage values in the org cache. Unlike the session cache, the org cache is not tied to any session and is available to the organization across requests and to all users.

Namespace

Cache

Usage

Cache Key Format

This table lists the format of the key parameter that some methods in this class take, such as `put`, `get`, and `contains`.

Key Format	Description
namespace.partition.key	Fully qualified key name.
key	Refers to a partition marked as default when the *namespace.partition* prefix is omitted.
`local.` *partition.key*	Use the `local` prefix to refer to the org's namespace when the org doesn't have a namespace defined. If the org has a namespace defined, the `local` prefix also refers to that org's namespace.

Note

- If no default partition is specified in the org, calling a cache method without fully qualifying the key name causes a `Cache.Org.OrgCacheException` to be thrown.
- The `local` prefix in an installed managed package refers to the namespace of the subscriber org and not the package's namespace. The cache `put` calls aren't allowed in a partition that the invoking class doesn't own.

Example

This class is the controller for a sample Visualforce page (shown in the subsequent code sample). The cached values are initially added to the cache by the `init()` method, which the Visualforce page invokes when it loads through the `action` attribute. The cache keys don't contain the `namespace.partition` prefix. They all refer to the default partition in your org. To run this sample, create a partition and mark it as default.

The Visualforce page contains four output components. These components call `get` methods on the controller that returns the following values from the cache: a date, data based on the `MyData` inner class, a counter, a text value, and a list. The size of the list is also returned.

The Visualforce page also contains two buttons. The Rerender button invokes the `go()` method on the controller. This method increases the values of the counter and the custom data in the cache.

As a result, the value next to `Cached datetime:` is cleared on the page.

Note

If another user logs in and runs this sample, this user gets the cache values that were last added or updated by the previous user. For example, if the counter value was five, the next user sees the counter value as increased to six.

```
public class OrgCacheController {

    // Inner class.
    // Used as the data type of a cache value.
    class MyData {
        public String value { get; set; }
        public Integer counter { get; set; }

        public MyData(String value) {
            this.value = value;
            this.counter = 0;
        }

        public void inc() {
            counter++;
        }

        override public String toString() {
            return this.value + ':' + this.counter;
        }
    }

    // Apex List.
    // Used as the data type of a cached value.
    private List<String> numbers =
            new List<String> { 'ONE', 'TWO', 'THREE', 'FOUR', 'FIVE' };

    // Constructor of the controller for the Visualforce page.
    public OrgCacheController() {
    }

    // Adds various values to the cache.
    // This method is called when the Visualforce page loads.
    public void init() {
        // All key values are not qualified by the namespace.partition
        // prefix because they use the default partition.

        // Add counter to the cache with initial value of 0
        //  or increment it if it's already there.
        if (!Cache.Org.contains('counter')) {
            Cache.Org.put('counter', 0);
        } else {
            Cache.Org.put('counter', getCounter() + 1);
        }

        // Add the datetime value to the cache only if it's not already there.
        if (!Cache.Org.contains('datetime')) {
            DateTime dt = DateTime.now();
            Cache.Org.put('datetime', dt);
        }

        // Add the custom data to the cache only if it's not already there.
        if (!Cache.Org.contains('data')) {
            Cache.Org.put('data', new MyData('Some custom value'));
        }

        // Add a list of number to the cache if not already there.
        if (!Cache.Org.contains('list')) {
            Cache.Org.put('list', numbers);
        }

        // Return counter from the cache.
        public Integer getCounter() {
            return (Integer)Cache.Org.get('counter');
        }

        // Return datetime value from the cache.
        public String getCachedDatetime() {
            DateTime dt = (DateTime)Cache.Org.get('datetime');
            return dt != null ? dt.format() : null;
        }

        // Return cached value whose type is the inner class MyData.
        public String getCachedData() {
            MyData mydata = (MyData)Cache.Org.get('data');
            return mydata != null ? mydata.toString() : null;
        }

        // Return output from the cache.
        public String getOutput() {
            return (String)Cache.Org.get('output');
        }

        // Return list from the cache.
        public List<String> getList() {
            return (List<String>)Cache.Org.get('list');
        }

        // Method invoked by the Rerender button on the Visualforce page.
        // Updates the values of various cached values.
        // Increases the values of counter and the MyData counter if those
        //   cache values are still in the cache.
        public PageReference go() {
            // Increase the cached counter value or set it to 0
            //  if it's not cached.
            if (Cache.Org.contains('counter')) {
                Cache.Org.put('counter', getCounter() + 1);
            } else {
                Cache.Org.put('counter', 0);
            }

            // Get the custom data value from the cache.
            MyData d = (MyData)Cache.Org.get('data');
            // Only if the data is already in the cache, update it.
            if (Cache.Org.contains('data')) {
                d.inc();
                Cache.Org.put('data', d);
            }

            return null;
        }

        // Method invoked by the Remove button on the Visualforce page.
        // Removes the datetime cached value from the org cache.
        public PageReference remove() {
            Cache.Org.remove('datetime');

            return null;
        }
    }
```

This is the Visualforce page that corresponds to the `OrgCacheController` class.

```
    <apex:page controller="OrgCacheController" action="{!init}">

        <apex:outputPanel id="output">
            <br/>Cached datetime: <apex:outputText value="{!cachedDatetime}"/>
            <br/>Cached data: <apex:outputText value="{!cachedData}"/>
        </apex:outputPanel>

        <br/><br/>
        <apex:form >
            <apex:commandButton id="go" action="{!go}" value="Rerender" rerender="output"/>
            <apex:commandButton id="remove" action="{!remove}" value="Remove datetime Key" rer
        </apex:form>

    </apex:page>
```

This is the output of the page after clicking the **Rerender** button twice. The counter value could differ in your case if a key named `counter` was already in the cache before running this sample.

```
Cached datetime:8/11/2015 1:58 PM
Cached data:Some custom value:2
Cached counter:2
Output:Cached text value
Repeat:ONE TWO THREE FOUR FIVE
List size:5
```

- **Org Constants**
  The Org class provides a constant that you can use when setting the time-to-live (TTL) value.
- **Org Methods**

See Also

- *Apex Developer Guide*: Platform Cache

Org Constants

The Org class provides a constant that you can use when setting the time-to-live (TTL) value.

Constant	Description
`MAX_TTL_SECS`	Represents the maximum amount of time, in seconds, to keep the cached value in the org cache.

Org Methods

The following are methods for `Org`. All methods are static.

- **contains(key)**
  Returns `true` if the org cache contains a cached value corresponding to the specified key.
- **contains(keys)**
  Returns `true` if the org cache contains values for the specified key entries.
- **contains(setOfKeys)**
  Returns `true` if the org cache contains values for a specified set of keys.
- **get(key)**
  Returns the cached value corresponding to the specified key from the org cache.
- **get(cacheBuilder, key)**
  Returns the cached value corresponding to the specified key from the org cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

- getAvgGetTime()
  Returns the average time taken to get a key from the org cache, in nanoseconds.
- getAvgValueSize()
  Deprecated and available only in API versions 49.0 and earlier. Returns the average item size for keys in the org cache, in bytes.
- getCapacity()
  Returns the percentage of org cache capacity that has been used.
- getKeys()
  Returns a set of all keys that are stored in the org cache and visible to the invoking namespace.
- getMaxGetSize()
  Returns the maximum item size of all the keys fetched from the org cache, in bytes.
- getMaxGetTime()
  Returns the maximum time taken to get a key from the org cache, in nanoseconds.
- getMaxValueSize()
  Deprecated and available only in API versions 49.0 and earlier. Returns the maximum item size for keys in the org cache, in bytes.
- getMissRate()
  Returns the miss rate in the org cache.
- getName()
  Returns the name of the default cache partition.
- getNumKeys()
  Returns the total number of keys in the org cache.
- getPartition(partitionName)
  Returns a partition from the org cache that corresponds to the specified partition name.
- put(key, value)
  Stores the specified key/value pair as a cached entry in the org cache. 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 org cache and sets the cached value's visibility.
- put(key, value, ttlSecs)
  Stores the specified key/value pair as a cached entry in the org cache 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 org cache. 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 the org cache.
- remove(cacheBuilder, key)
  Deletes the cached value corresponding to the specified key from the org cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

contains(key)

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

Signature

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

Parameters

key

Return Value

Type: Boolean

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

contains(keys)

Returns `true` if the org cache contains values for the specified key entries.

Signature

```
public static List<Boolean> contains(List<String> keys)
```

Parameters

keys

Type: List<String>

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

Return Value

Type: List<Boolean>

`true` if the key entries are found. Othewise, `false`.

contains(setOfKeys)

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

Signature

```
public static 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 the default partition. It fetches the cache key and the corresponding Boolean value for the key entry from the org cache of the default partition.

```
Set<String> keys = new Set<String>{'key1','key2','key3','key4','key5'};
Map<String,Boolean> result = Cache.Org.contains(keys);
```

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

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

Set<String> keys = new Set<String>{'local.p1.key','local.p2.key', 'local.p3.key'};
Map<String,Boolean> result = Cache.Org.contains(keys);
for(String key : result.keySet()) {
    system.debug('key: ' + key);
    system.debug('Is Key Present in the cache : +  result.get(key));
}
```

get(key)

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

Signature

`public static Object get(String key)`

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value. For information about the format of the key name, see Usage.

Return Value

Type: Object

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

Usage

Because `Cache.Org.get()` returns an object, cast the returned value to a specific type to facilitate use of the returned value.

```
// Get a cached value
Object obj = Cache.Org.get('ns1.partition1.orderDate');
// Cast return value to a specific data type
DateTime dt2 = (DateTime)obj;
```

If a `Cache.Org.get()` call doesn't find the referenced key, it returns `null`.

get(cacheBuilder, key)

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

Signature

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

Parameters

cacheBuilder

Type: System.Type

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.

Usage

Because `Cache.Org.get(cacheBuilder, key)` returns an object, cast the returned value to a specific type to facilitate use of the returned value.

```
return ((DateTime)Cache.Org.get(DateCache.class, 'datetime')).format();
```

get(keys)

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

Signature

`public static 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 the default partition.

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

Fetch multiple keys from the org cache of different partitions.

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

Set<String> keys = new Set<String>{'local.p1.key','local.p2.key', 'local.p3.key'};
```

getAvgGetSize()

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

Signature

```
public static Long getAvgGetSize()
```

Return Value

Type: Long

Example

In this example the following keys and their corresponding value sizes are inserted. The code then fetches the keys: key 1, key 2, key 3 and key 4 and returns the average item size of the fetched keys.

Key	Key Value Size
key 1	42
key 2	42
key 3	58
key 4	36
key 5	36

```
// Inserting keys key1, key2, key3, key4, key5
Cache.Org.put('key1', 'value1');
Cache.Org.put('key2', 'value2');
Cache.Org.put('key3', 'this is a big value !!!');
Cache.Org.put('key4', 4);
Cache.Org.put('key5', 5);


// Fetching keys - key1, key2, key3, key4
Object v1 = Cache.Org.get('key1');
Object v2 = Cache.Org.get('key2');
Object v3 = Cache.Org.get('key3');
Object v4 = Cache.Org.get('key4');

// Fetching average get size
Long val = Cache.Org.getAvgGetSize();
// Avg item size returned is 44 ( average of 42(key1), 42(key2), 58(key3) and 36(key4) key
System.debug('Avg Get Size :' + val);
```

getAvgGetTime()

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

Signature

```
public static Long getAvgGetTime()
```

Return Value

Deprecated and available only in API versions 49.0 and earlier. Returns the average item size for keys in the org cache, in bytes.

Signature

```
public static Long getAvgValueSize()
```

Return Value

Type: Long

getCapacity()

Returns the percentage of org cache capacity that has been used.

Signature

```
public static Double getCapacity()
```

Return Value

Type: Double

Used cache as a percentage number.

getKeys()

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

Signature

```
public static 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 org cache, in bytes.

Signature

```
public static Long getMaxGetSize()
```

Return Value

Type: Long

Example

In this example the following keys and their corresponding value sizes are inserted. The code fetches the keys: key 1, key 2 and key 4 and returns the maximum key value size from the fetched keys.

Key	Key Value Size
key 1	42
key 2	42

key 4	36
key 5	36

```
// Inserting keys key1, key2, key3, key4, key5
Cache.Org.put('key1', 'value1');
Cache.Org.put('key2', 'value2');
Cache.Org.put('key3', 'this is a big value !!!');
Cache.Org.put('key4', 4);
Cache.Org.put('key5', 5);


// Fetching keys - key1, key2, key4
Object v1 = Cache.Org.get('key1');
Object v2 = Cache.Org.get('key2');
Object v4 = Cache.Org.get('key4');

// Fetching max get size
Long val = Cache.Org.getMaxGetSize();
// Max item size returned is 42 ( max of 42(key1), 42(key2), and 36(key4) keys that were f
System.debug('Max Get Size :' + val);
```

getMaxGetTime()

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

Signature

```
public static Long getMaxGetTime()
```

Return Value

Type: Long

getMaxValueSize()

Deprecated and available only in API versions 49.0 and earlier. Returns the maximum item size for keys in the org cache, in bytes.

Signature

```
public static Long getMaxValueSize()
```

Return Value

Type: Long

getMissRate()

Returns the miss rate in the org cache.

Signature

```
public static Double getMissRate()
```

Return Value

Type: Double

```
public String getName()
```

Return Value

Type: String

The name of the default cache partition.

getNumKeys()

Returns the total number of keys in the org cache.

Signature

```
public static Long getNumKeys()
```

Return Value

Type: Long

getPartition(partitionName)

Returns a partition from the org cache that corresponds to the specified partition name.

Signature

```
public static cache.OrgPartition getPartition(String partitionName)
```

Parameters

partitionName

Type: String

A partition name that is qualified by the namespace, for example, *namespace.partition*.

Return Value

Type: Cache.OrgPartition

Example

After you get the org partition, you can add and retrieve the partition’s cache values.

```
// Get partition
Cache.OrgPartition orgPart = Cache.Org.getPartition('myNs.myPartition');
// Retrieve cache value from the partition
if (orgPart.contains('BookTitle')) {
    String cachedTitle = (String)orgPart.get('BookTitle');
}

// Add cache value to the partition
orgPart.put('OrderDate', Date.today());

// Or use dot notation to call partition methods
String cachedAuthor = (String)Cache.Org.getPartition('myNs.myPartition').get('BookAuthor'
```

put(key, value)

Stores the specified key/value pair as a cached entry in the org cache. The `put` method can write only to the cache in your org’s namespace.

key

Type: String

A case-sensitive string value that uniquely identifies a cached value. For information about the format of the key name, see Usage.

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 org cache and sets the cached value's visibility.

Signature

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

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value. For information about the format of the key name, see Usage.

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 org cache and sets the cached value's lifetime.

Signature

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

Parameters

key

Type: String

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 org cache. The maximum is 172,800 seconds (48 hours). The minimum value is 300 seconds or 5 minutes. The default value is 86,400 seconds (24 hours).

Return Value

Type: void

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

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

Signature

```
public static 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. For information about the format of the key name, see Usage.

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 org cache. The maximum is 172,800 seconds (48 hours). The minimum value is 300 seconds or 5 minutes. The default value is 86,400 seconds (24 hours).

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)

Parameters

key

Type: String

A case-sensitive string value that uniquely identifies a cached value. For information about the format of the key name, see Usage.

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 org cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

Signature

```
public static 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`.

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

Share your feedback