Session Class

Use the `Cache.Session` class to add, retrieve, and manage values in the session cache. The session cache is active as long as the user's Salesforce session is valid (the user is logged in, and the session is not expired).

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.Session.SessionCacheException` 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 are not 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 a default partition in your org. The Visualforce page expects a partition named `myPartition`. To run this sample, create a default partition in your org with the name `myPartition`.

The Visualforce page contains four output components. The first three components call `get` methods on the controller that return the following values from the cache: a date, data based on the `MyData` inner class, and a counter. The next output component uses the `$Cache.Session` global the controller. This method increases the values of the counter and the custom data in the cache. If you click **Rerender**, the two counters increase by one each time. The `go()` method retrieves the values of these counters from the cache, increments their values by one, and stores them again in the cache.

The Remove button deletes the date-time value (with key `datetime`) from the cache. As a result, the value next to `Cached datetime:` is cleared on the page.

```
public class SessionCacheController {

    // 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 SessionCacheController() {
    }

    // 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.Session.contains('counter')) {
            Cache.Session.put('counter', 0);
        } else {
            Cache.Session.put('counter', getCounter() + 1);
        }

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

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

        // Add a list of number to the cache if not already there.
        if (!Cache.Session.contains('list')) {
            Cache.Session.put('list', numbers);
        }
        // Return counter from the cache.
        public Integer getCounter() {
            return (Integer)Cache.Session.get('counter');
        }

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

        // 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.Session.contains('counter')) {
                Cache.Session.put('counter', getCounter() + 1);
            } else {
                Cache.Session.put('counter', 0);
            }

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

            return null;
        }

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

            return null;
        }
    }
```

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

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

        <apex:outputPanel id="output">
            <br/>Cached datetime: <apex:outputText value="{!cachedDatetime}"/>
            <br/>Cached data: <apex:outputText value="{!cachedData}"/>
            <br/>Cached counter: <apex:outputText value="{!counter}"/>
            <br/>Output: <apex:outputText value="{!$Cache.Session.local.myPartition.output}"/>
            <br/>Repeat: <apex:repeat var="item" value="{!$Cache.Session.local.myPartition.lis
                <apex:outputText value="{!item}"/>&nbsp;
            </apex:repeat>
            <br/>List size: <apex:outputText value="{!$Cache.Session.local.myPartition.list.si
        </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
```

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
```

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

See Also

- *Apex Developer Guide*: Platform Cache

Session Constants

The Session 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 session cache.

Session Methods

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

- **contains(key)**
  Returns `true` if the session cache contains a cached value corresponding to the specified key.
- **contains(setOfKeys)**
  Returns `true` if the cache contains values for a specified set of keys.
- **get(key)**
  Returns the cached value corresponding to the specified key from the session cache.
- **get(keys)**
  Returns the cached values corresponding to the specified set of keys from the session cache.
- **get(cacheBuilder, key)**
  Returns the cached value corresponding to the specified key from the session cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.
- **getAvgGetSize()**
  Returns the average item size of all the keys fetched from the session cache, in bytes.
- **getAvgGetTime()**
  Returns the average time taken to get a key from the session cache, in nanoseconds.
- **getAvgValueSize()**
  Deprecated and available only in API versions 49.0 and earlier. Returns the average item size for keys in the session cache, in bytes.

namespace.

- **getMaxGetSize()**
  Returns the maximum item size of all the keys fetched from the session cache, in bytes.
- **getMaxGetTime()**
  Returns the maximum time taken to get a key from the session cache, in nanoseconds.
- **getMaxValueSize()**
  Deprecated and available only in API versions 49.0 and earlier. Returns the maximum item size for keys in the session cache, in bytes.
- **getMissRate()**
  Returns the miss rate in the session cache.
- **getName()**
  Returns the name of the default cache partition.
- **getNumKeys()**
  Returns the total number of keys in the session cache.
- **getPartition(partitionName)**
  Returns a partition from the session cache that corresponds to the specified partition name.
- **isAvailable()**
  Returns `true` if the session cache is available for use. 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 session 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 session cache and sets the cached value's visibility.
- **put(key, value, ttlSecs)**
  Stores the specified key/value pair as a cached entry in the session 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 session 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 session cache.
- **remove(cacheBuilder, key)**
  Deletes the cached value corresponding to the specified key from the session cache. Use this method if your cached value is a class that implements the `CacheBuilder` interface.

contains(key)

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

Signature

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

Parameters

key

Type: String

Type: Boolean

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

contains(setOfKeys)

Returns `true` if the 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 session cache of the default partition.

```
Set<String> keys = new Set<String>{'key1','key2','key3','key4','key5'};
Map<String,Boolean> result = Cache.Session.contains(keys);
for(String key : result.keySet()) {
    system.debug('key: ' + key);
    system.debug('Is Key Present in the cache : ' +  result.get(key));
}
```

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 session 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.Session.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 session cache.

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.Session.get()` returns an object, we recommend that you cast the returned value to a specific type to facilitate use of the returned value.

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

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

get(keys)

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

Example

Fetch multiple keys from the session cache of the default partition.

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

```
// 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,Object> result = Cache.Session.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 session 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

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.

Usage

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

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

getAvgGetSize()

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

Signature

`public static Long getAvgGetSize()`

Return Value

Type: Long

getAvgGetTime()

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

Type: Long

getAvgValueSize()

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

Signature

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

Return Value

Type: Long

getCapacity()

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

Signature

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

Return Value

Type: Double

Used cache as a percentage number.

getKeys()

Returns all keys that are stored in the session 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 session cache, in bytes.

Signature

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

Return Value

Type: Long

getMaxGetTime()

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

Signature

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

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

Signature

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

Return Value

Type: Long

getMissRate()

Returns the miss rate in the session cache.

Signature

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

Return Value

Type: Double

getName()

Returns the name of the default cache partition.

Signature

```
public String getName()
```

Return Value

Type: String

The name of the default cache partition.

getNumKeys()

Returns the total number of keys in the session cache.

Signature

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

Return Value

Type: Long

getPartition(partitionName)

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

Signature

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

Parameters

partitionName
Type: String

Example

After you get the session partition, you can add and retrieve the partition's cache values.

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

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

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

isAvailable()

Returns `true` if the session cache is available for use. 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 static Boolean isAvailable()`

Return Value

Type: Boolean

`true` if the session cache is available. Otherwise, `false`.

put(key, value)

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

Signature

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

Parameters

key

Type: String

A string that uniquely identifies the value to be cached. 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

Signature

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

Parameters

key

Type: String

A string that uniquely identifies the value to be cached. 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 session cache and sets the cached value's lifetime.

Signature

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

Parameters

key

Type: String

A string that uniquely identifies the value to be cached. 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 session cache. The cached values remain in the cache as long as the Salesforce session hasn't expired. The maximum value is 28,800 seconds or eight hours. The minimum value is 300 seconds or five minutes.

Return Value

Type: void

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

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

Parameters

key

Type: String

A string that uniquely identifies the value to be cached. 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 session cache. The cached values remain in the cache as long as the Salesforce session hasn't expired. The maximum value is 28,800 seconds or eight hours. The minimum value is 300 seconds or five minutes.

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 the session cache.

Signature

```
public static Boolean remove(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: Boolean

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

remove(cacheBuilder, key)

```
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`.