Package org.springframework.util

Class ObjectUtils

java.lang.Object
    org.springframework.util.ObjectUtils

```
public abstract class ObjectUtils
extends Object
```

Miscellaneous object utility methods.

Mainly for internal use within the framework.

Thanks to Alex Ruiz for contributing several enhancements to this class!

Since:

19.03.2004

Author:

Juergen Hoeller, Keith Donald, Rod Johnson, Rob Harrop, Chris Beams, Sam Brannen

See Also:

`ClassUtils`, `CollectionUtils`, `StringUtils`

Constructor Summary

Constructors

Constructor	Description
`ObjectUtils()`	

Method Summary

All Methods Static Methods Concrete Methods Deprecated Methods

Modifier and Type	Method	Description
`static <A,O extends A> A[]`	`addObjectToArray(A[] array, O obj)`	Append the given object to the given array, returning a new array consisting of the input array contents plus the given object.
`static <A,O extends A> A[]`	`addObjectToArray(A[] array, O obj, int position)`	Add the given object to the given array at the specified position, returning a new array consisting of the input array contents plus the given object.

`static <E extends Enum <?>> E`	`caseInsensitiveValueOf (E[] enumValues, String constant)`	Case insensitive alternative to `Enum.valueOf(Class, String)` .
`static boolean`	`containsConstant (Enum <?>[] enumValues, String constant)`	Check whether the given array of enum constants contains a constant with the given name, ignoring case when determining a match.
`static boolean`	`containsConstant (Enum <?>[] enumValues, String constant, boolean caseSensitive)`	Check whether the given array of enum constants contains a constant with the given name.
`static boolean`	`containsElement (Object [] array, Object element)`	Check whether the given array contains the given element.
`static String`	`getDisplayString (Object obj)`	Return a content-based String representation if `obj` is not `null`; otherwise returns an empty String.
`static String`	`getIdentityHexString (Object obj)`	Return a hex String form of an object's identity hash code.
`static String`	`identityToString (Object obj)`	Return a String representation of an object's overall identity.
`static boolean`	`isArray (Object obj)`	Determine whether the given object is an array: either an Object array or a primitive array.
`static boolean`	`isCheckedException (Throwable ex)`	Return whether the given throwable is a checked exception: that is, neither a RuntimeException nor an Error.
`static boolean`	`isCompatibleWithThrowsClause (Throwable ex, Class <?>... declaredExceptions)`	Check whether the given exception is compatible with the specified exception types, as declared in a `throws` clause.
`static boolean`	`isEmpty (Object obj)`	Determine whether the given object is empty.
`static boolean`	`isEmpty (Object [] array)`	Determine whether the given array is empty: i.e.
`static String`	`nullSafeClassName (Object obj)`	Determine the class name for the given object.
`static String`	`nullSafeConciseToString (Object obj)`	Generate a null-safe, concise string representation of the

		supplied object as described below.
static boolean	nullSafeEquals(Object o1, Object o2)	Determine if the given objects are equal, returning true if both are null or false if only one is null.
static int	nullSafeHash(Object... elements)	Return a hash code for the given elements, delegating to nullSafeHashCode(Object) for each element.
static int	nullSafeHashCode(boolean[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(boolean[])
static int	nullSafeHashCode(byte[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(byte[])
static int	nullSafeHashCode(char[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(char[])
static int	nullSafeHashCode(double[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(double[])
static int	nullSafeHashCode(float[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(float[])
static int	nullSafeHashCode(int[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(int[])
static int	nullSafeHashCode(long[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(long[])
static int	nullSafeHashCode(short[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(short[])
static int	nullSafeHashCode(Object obj)	Return a hash code for the given object, typically the value of Object.hashCode().
static int	nullSafeHashCode(Object[] array)	**Deprecated.** as of 6.1 in favor of Arrays.hashCode(Object[])
atic String	nullSafeToString(boolean[] array)	Return a String representation of the contents of the specified

		array.
static String	nullSafeToString (byte[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString (char[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString (double[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString (float[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString (int[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString (long[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString (short[] array)	Return a String representation of the contents of the specified array.
static String	nullSafeToString(Object obj)	Return a String representation of the specified Object.
static String	nullSafeToString(Object[] array)	Return a String representation of the contents of the specified array.
static Object []	toObjectArray(Object source)	Convert the given array (which may be a primitive array) to an object array (if necessary, to an array of primitive wrapper objects).
static Object	unwrapOptional(Object obj)	Unwrap the given object which is potentially a Optional .

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

nstructor Details

ObjectUtils

```
public ObjectUtils()
```

Method Details

isCheckedException

```
public static boolean isCheckedException(Throwable ex)
```

Return whether the given throwable is a checked exception: that is, neither a RuntimeException nor an Error.

Parameters:

`ex` - the throwable to check

Returns:

whether the throwable is a checked exception

See Also:

`Exception` , `RuntimeException` , `Error`

isCompatibleWithThrowsClause

```
public static boolean isCompatibleWithThrowsClause(Throwable ex,
                                                   @Nullable
                                                   Class <?>... declaredExceptions)
```

Check whether the given exception is compatible with the specified exception types, as declared in a `throws` clause.

Parameters:

`ex` - the exception to check

`declaredExceptions` - the exception types declared in the throws clause

Returns:

whether the given exception is compatible

isArray

```
@Contract("null -> false")
public static boolean isArray(@Nullable
                              Object obj)
```

Determine whether the given object is an array: either an Object array or a primitive array.

Parameters:

`j` - the object to check

isEmpty

```
@Contract("null -> true")
public static boolean isEmpty(@Nullable
                              Object[] array)
```

Determine whether the given array is empty: i.e. `null` or of zero length.

Parameters:

`array` - the array to check

See Also:

`isEmpty(Object)`

isEmpty

```
@Contract("null -> true")
public static boolean isEmpty(@Nullable
                              Object obj)
```

Determine whether the given object is empty.

This method supports the following object types.

- `Optional`: considered empty if not `Optional.isPresent()`
- `Array`: considered empty if its length is zero
- `CharSequence`: considered empty if its length is zero
- `Collection`: delegates to `Collection.isEmpty()`
- `Map`: delegates to `Map.isEmpty()`

If the given object is non-null and not one of the aforementioned supported types, this method returns `false`.

Parameters:

`obj` - the object to check

Returns:

`true` if the object is `null` or *empty*

Since:

4.2

See Also:

`Optional.isPresent()`,
`isEmpty(Object[])`,
`StringUtils.hasLength(CharSequence)`,
`CollectionUtils.isEmpty(java.util.Collection)`,
`CollectionUtils.isEmpty(java.util.Map)`

unwrapOptional

```
@Nullable
public static Object unwrapOptional(@Nullable
                                    Object obj)
```

Unwrap the given object which is potentially a `Optional` .

Parameters:

`obj` - the candidate object

Returns:

either the value held within the `Optional`, `null` if the `Optional` is empty, or simply the given object as-is

Since:

5.0

containsElement

```
public static boolean containsElement(@Nullable
                                      Object [] array,
                                      Object element)
```

Check whether the given array contains the given element.

Parameters:

`array` - the array to check (may be `null`, in which case the return value will always be `false`)

`element` - the element to check for

Returns:

whether the element has been found in the given array

containsConstant

```
public static boolean containsConstant(Enum <?>[] enumValues,
                                       String constant)
```

Check whether the given array of enum constants contains a constant with the given name, ignoring case when determining a match.

Parameters:

`enumValues` - the enum values to check, typically obtained via `MyEnum.values()`

`constant` - the constant name to find (must not be null or empty string)

Returns:

whether the constant has been found in the given array

containsConstant

```
public static boolean containsConstant(Enum <?>[] enumValues,
                                       String constant,
                                       boolean caseSensitive)
```

Check whether the given array of enum constants contains a constant with the given name.

Parameters:

`enumValues` - the enum values to check, typically obtained via `MyEnum.values()`

`constant` - the constant name to find (must not be null or empty string)

`caseSensitive` - whether case is significant in determining a match

Returns:

whether the constant has been found in the given array

caseInsensitiveValueOf

```
public static <E extends Enum<?>> E caseInsensitiveValueOf(E[] enumValues,
                                                           String constant)
```

Case insensitive alternative to `Enum.valueOf(Class, String)`.

Type Parameters:

`E` - the concrete Enum type

Parameters:

`enumValues` - the array of all Enum constants in question, usually per `Enum.values()`

`constant` - the constant to get the enum value of

Throws:

`IllegalArgumentException` - if the given constant is not found in the given array of enum values. Use `containsConstant(Enum[], String)` as a guard to avoid this exception.

addObjectToArray

```
public static <A,O extends A> A[] addObjectToArray(@Nullable
                                                    A[] array,
                                                    @Nullable
                                                    O obj)
```

Append the given object to the given array, returning a new array consisting of the input array contents plus the given object.

Parameters:

`array` - the array to append to (can be `null`)

`obj` - the object to append

Returns:

the new array (of the same component type; never `null`)

addObjectToArray

```
public static <A,O extends A> A[] addObjectToArray(@Nullable
                                                   A[] array,
                                                   @Nullable
                                                   O obj,
                                                   int position)
```

Add the given object to the given array at the specified position, returning a new array consisting of the input array contents plus the given object.

Parameters:

`array` - the array to add to (can be `null`)

`obj` - the object to append

`position` - the position at which to add the object

Returns:

the new array (of the same component type; never `null`)

Since:

6.0

toObjectArray

```
public static Object[] toObjectArray(@Nullable
                                     Object source)
```

Convert the given array (which may be a primitive array) to an object array (if necessary, to an array of primitive wrapper objects).

A `null` source value or empty primitive array will be converted to an empty Object array.

Parameters:

`source` - the (potentially primitive) array

Returns:

the corresponding object array (never `null`)

Throws:

`IllegalArgumentException` - if the parameter is not an array

nullSafeEquals

```
@Contract("null, null -> true; null, _ -> false; _, null -> false")
public static boolean nullSafeEquals(@Nullable
                                     Object o1,
                                     @Nullable
                                     Object o2)
```

Determine if the given objects are equal, returning `true` if both are `null` or `false` if only one is `null`.

Compares arrays with `Arrays.equals`, performing an equality check based on the array elements rather than the array reference.

Parameters:

`o1` - first Object to compare

`o2` - second Object to compare

Returns:

whether the given objects are equal

See Also:

`Object.equals(Object)` ,
`Arrays.equals(long[], long[])`

nullSafeHash

```
public static  int  nullSafeHash(@Nullable
                                  Object ...  elements)
```

Return a hash code for the given elements, delegating to `nullSafeHashCode(Object)` for each element. Contrary to `Objects.hash(Object...)` , this method can handle an element that is an array.

Parameters:

`elements` - the elements to be hashed

Returns:

a hash value of the elements

Since:

6.1

nullSafeHashCode

```
public static  int  nullSafeHashCode(@Nullable
                                      Object   obj)
```

Return a hash code for the given object, typically the value of `Object.hashCode()` . If the object is an array, this method will delegate to one of the `Arrays.hashCode` methods. If the object is `null`, this method returns `0`.

See Also:

`Object.hashCode()` , `Arrays`

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                      Object []  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(Object[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                    boolean[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(boolean[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns 0.

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                    byte[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(byte[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns 0.

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                    char[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(char[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns 0.

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                    double[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(double[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                     float[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(float[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns 0.

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                     int[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(int[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns 0.

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                     long[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(long[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns 0.

nullSafeHashCode

```
@Deprecated (since ="6.1")
public static  int  nullSafeHashCode(@Nullable
                                     short[]  array)
```

Deprecated.
as of 6.1 in favor of `Arrays.hashCode(short[])`

Return a hash code based on the contents of the specified array. If `array` is `null`, this method returns

identityToString

```
public static String identityToString(@Nullable
                                      Object obj)
```

Return a String representation of an object's overall identity.

Parameters:

`obj` - the object (may be `null`)

Returns:

the object's identity as String representation, or an empty String if the object was `null`

getIdentityHexString

```
public static String getIdentityHexString(Object obj)
```

Return a hex String form of an object's identity hash code.

Parameters:

`obj` - the object

Returns:

the object's identity code in hex notation

getDisplayString

```
public static String getDisplayString(@Nullable
                                      Object obj)
```

Return a content-based String representation if `obj` is not `null`; otherwise returns an empty String.

Differs from `nullSafeToString(Object)` in that it returns an empty String rather than "null" for a `null` value.

Parameters:

`obj` - the object to build a display String for

Returns:

a display String representation of `obj`

See Also:

`nullSafeToString(Object)`

nullSafeClassName

```
public static String nullSafeClassName(@Nullable
                                       Object obj)
```

Determine the class name for the given object.

turns a `"null"` String if `obj` is `null`.

Parameters:

`obj` - the object to introspect (may be `null`)

Returns:

the corresponding class name

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      Object obj)
```

Return a String representation of the specified Object.

Builds a String representation of the contents in case of an array. Returns a `"null"` String if `obj` is `null`.

Parameters:

`obj` - the object to build a String representation for

Returns:

a String representation of `obj`

See Also:

`nullSafeConciseToString(Object)`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      Object[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      boolean[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). djacent elements are separated by the characters `", "` (a comma followed by a space). Returns a ull" String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      byte[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      char[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      double[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

ameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      float[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      int[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                      long[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

turns:

a String representation of `array`

nullSafeToString

```
public static String nullSafeToString(@Nullable
                                       short[] array)
```

Return a String representation of the contents of the specified array.

The String representation consists of a list of the array's elements, enclosed in curly braces (`"{}"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns a `"null"` String if `array` is `null`.

Parameters:

`array` - the array to build a String representation for

Returns:

a String representation of `array`

nullSafeConciseToString

```
public static String nullSafeConciseToString(@Nullable
                                              Object obj)
```

Generate a null-safe, concise string representation of the supplied object as described below.

Favor this method over `nullSafeToString(Object)` when you need the length of the generated string to be limited.

Returns:

- `"null"` if `obj` is `null`
- `"Optional.empty"` if `obj` is an empty `Optional`
- `"Optional[<concise-string>]"` if `obj` is a non-empty `Optional`, where `<concise-string>` is the result of invoking this method on the object contained in the `Optional`
- `"{}"` if `obj` is an empty array
- `"{...}"` if `obj` is a `Map` or a non-empty array
- `"[...]"` if `obj` is a `Collection`
- Class name if `obj` is a `Class`
- Charset name if `obj` is a `Charset`
- TimeZone ID if `obj` is a `TimeZone`
- Zone ID if `obj` is a `ZoneId`
- Potentially truncated string if `obj` is a `String` or `CharSequence`
- Potentially truncated string if `obj` is a *simple value type* whose `toString()` method returns a non-null value
- Otherwise, a string representation of the object's type name concatenated with `"@"` and a hex string form of the object's identity hash code

In the context of this method, a *simple value type* is any of the following: primitive wrapper (excluding `Void`), `Enum`, `Number`, `Date`, `Temporal`, `File`, `Path`, `URI`, `URL`, `InetAddress`, `Currency`, `Locale`, `UUID`, `Pattern`.

rameters:

`bj` - the object to build a string representation for

Returns:

a concise string representation of the supplied object

Since:

5.3.27

See Also:

`nullSafeToString(Object)`,
`StringUtils.truncate(CharSequence)`,
`ClassUtils.isSimpleValueType(Class)`