Module jdk.incubator.foreign
Package jdk.incubator.foreign

Interface LibraryLookup

`public interface LibraryLookup`

A native library lookup. Exposes a lookup operation for searching symbols, see `lookup(String)`. A given native library remains loaded as long as there is at least one *live* library lookup instance referring to it. All symbol instances (see `LibraryLookup.Symbol`) generated by a given library lookup object contain a strong reference to said lookup object, therefore preventing library unloading; in turn method handle instances obtained from `CLinker.downcallHandle(Addressable, MethodType, FunctionDescriptor)` also maintain a strong reference to the addressable parameter used for their construction. This means that there is always a strong reachability chain from a native method handle to a lookup object (the one that was used to lookup the native library symbol the method handle refers to); this is useful to prevent situations where a native library is unloaded in the middle of a native call.

In cases where a client wants to create a memory segment out of a lookup symbol, the client might want to attach the lookup symbol to the newly created segment, so that the symbol will be kept reachable as long as the memory segment is reachable; this can be achieved by creating the segment using the `MemoryAddress.asSegmentRestricted(long, Runnable, Object)` restricted segment factory, as follows:

```
LibraryLookup defaultLookup = LibraryLookup.defaultLookup();
LibraryLookup.Symbol errno = defaultLookup.lookup("errno");
MemorySegment errnoSegment = errno.address().asRestrictedSegment(4, errno);
```

To allow for a library to be unloaded, a client will have to discard any strong references it maintains, directly, or indirectly to a lookup object associated with given library.

Unless otherwise specified, passing a `null` argument, or an array argument containing one or more `null` elements to a method in this class causes a `NullPointerException` to be thrown.

Nested Class Summary

Nested Classes

Modifier and Type	Interface	Description
`static interface`	**`LibraryLookup.Symbol`**	A symbol retrieved during a library lookup.

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Abstract Methods**

Modifier and Type	Method	Description
`Optional<LibraryLookup.`	`lookup(String name)`	Looks up a symbol with given name in this library.
`static LibraryLookup`	`ofDefault()`	Obtain a default library lookup object.
`static LibraryLookup`	`ofLibrary(String libName)`	Obtain a library lookup object corresponding to a library identified by given library name.
`static LibraryLookup`	`ofPath(Path path)`	Obtain a library lookup object corresponding to a library identified by given path.

Method Details

lookup

```
Optional<LibraryLookup.Symbol> lookup(String name)
```

Looks up a symbol with given name in this library. The returned symbol maintains a strong reference to this lookup object.

Parameters:

`name` - the symbol name.

Returns:

the library symbol (if any).

ofDefault

```
static LibraryLookup ofDefault()
```

Obtain a default library lookup object.

Returns:

the default library lookup object.

ofPath

```
static LibraryLookup ofPath(Path path)
```

Obtain a library lookup object corresponding to a library identified by given path.

Parameters:

`path` - the library absolute path.

Returns:

a library lookup object for given path.

Throws:

`IllegalArgumentException` - if the specified path does not correspond to an absolute path, e.g. if `!path.isAbsolute()`.

ofLibrary

```
static LibraryLookup ofLibrary(String libName)
```

Obtain a library lookup object corresponding to a library identified by given library name. The library name is decorated according to the platform conventions (e.g. on Linux, the `lib` prefix is added, as well as the `.so` extension); the resulting name is then looked up in the standard native library path (which can be overriden, by setting the `java.library.path` property).

Parameters:

`libName` - the library name.

Returns:

a library lookup object for given library name.