**[Open Declaration](eclipse-open:%E2%98%82=concurrent/F:\/softhome\/Java\/jdk1.6.0_24\/jre\/lib\/rt.jar%3cjava.util.concurrent(Future.class%E2%98%83Future)java.util.concurrent.Future<**[**V**](eclipse-javadoc:%E2%98%82=concurrent/F:%5C/softhome%5C/Java%5C/jdk1.6.0_24%5C/jre%5C/lib%5C/rt.jar%3Cjava.util.concurrent(Future.class%E2%98%83Future%5dV)**>**

A Future represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method get when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the cancel method. Additional methods are provided to determine if the task completed normally or was cancelled. Once a computation has completed, the computation cannot be cancelled. If you would like to use a Future for the sake of cancellability but not provide a usable result, you can declare types of the form Future<?> and return null as a result of the underlying task.

**Sample Usage** (Note that the following classes are all made-up.)

interface ArchiveSearcher { String search(String target); }

class App {

ExecutorService executor = ...

ArchiveSearcher searcher = ...

void showSearch(final String target)

throws InterruptedException {

Future<String> future

= executor.submit(new Callable<String>() {

public String call() {

return searcher.search(target);

}});

displayOtherThings(); // do other things while searching

try {

displayText(future.get()); // use future

} catch (ExecutionException ex) { cleanup(); return; }

}

}

The [FutureTask](eclipse-javadoc:%E2%98%82=concurrent/F:%5C/softhome%5C/Java%5C/jdk1.6.0_24%5C/jre%5C/lib%5C/rt.jar%3Cjava.util.concurrent(Future.class%E2%98%83Future%E2%98%82FutureTask) class is an implementation of Future that implements Runnable, and so may be executed by an Executor. For example, the above construction with submit could be replaced by:

FutureTask<String> future =

new FutureTask<String>(new Callable<String>() {

public String call() {

return searcher.search(target);

}});

executor.execute(future);

Memory consistency effects: Actions taken by the asynchronous computation [*happen-before*](http://java.sun.com/javase/6/docs/api/java/util/concurrent/package-summary.html#MemoryVisibility) actions following the corresponding Future.get() in another thread.

**Parameters:**

**<V>** The result type returned by this Future's get method