future 4 student book pdf

AI generated article from Bing

---

std::future - cppreference.com

The class template std::future provides a mechanism to access the result of asynchronous operations: An asynchronous operation (created via std::async, std::packaged_task, or std::promise) can provide a std::future object to the creator of that asynchronous operation. The creator of the asynchronous operation can then use a variety of methods to query, wait for, or extract a value from the std ...

std::future::wait_until - cppreference.com

If the future is the result of a call to async that used lazy evaluation, this function returns immediately without waiting. The behavior is undefined if valid () is false before the call to this function, or Clock does not meet the Clock requirements. The programs is ill-formed if std::chrono::is_clock_v is false.(since C++20)

c++ - std::future in simple words? - Stack Overflow

In summary: std::future is an object used in multithreaded programming to receive data or an exception from a different thread; it is one end of a single-use, one-way communication channel between two threads, std::promise object being the other end.

std:: promise - cppreference.com

The promise is the "push" end of the promise-future communication channel: the operation that stores a value in the shared state synchronizes-with (as defined in std::memory_order) the successful return from any function that is waiting on the shared state (such as std::future::get).

std::future::get - cppreference.com

The get member function waits (by calling wait ()) until the shared state is ready, then retrieves the value stored in the shared state (if any). Right after calling this function, valid () is false. If valid () is false before the call to this function, the behavior is undefined.

future grants on a snowflake database - Stack Overflow

Considerations When future grants are defined on the same object type for a database and a schema in the same database, the schema-level grants take precedence over the database level grants, and the database level grants are ignored. This behavior applies to privileges on future objects granted to one role or different roles. Reproducible example:

Cannot build CMake project because "Compatibility with CMake < 3.5 has ...

In this case it does work. In general, it probably doesn't. I'm wondering how this break in backwards compatibility should in general be navigated. Perhaps installing a previous version of CMake is the only way that always works? That would mean that each project in the future should specify the CMake version on which it should be built.

std::future::wait_for - cppreference.com

If the future is the result of a call to std::async that used lazy evaluation, this function returns immediately without waiting. This function may block for longer than timeout_duration due to scheduling or resource contention delays. The standard recommends that a steady clock is used to measure the duration.

std::async - cppreference.com

The return type of std::async is std::future, where V is: ... The call to std::async synchronizes with the call to f, and the completion of f is sequenced before making the shared state ready.

React Router Future Flag Warning in Remix Vite app

 React Router Future Flag Warning: The revalidation behavior after 4xx/5xx action responses is changing in v7. You can use the v7_skipActionErrorRevalidation future flag to opt-in early.