Smart Pointers: Cyclic References

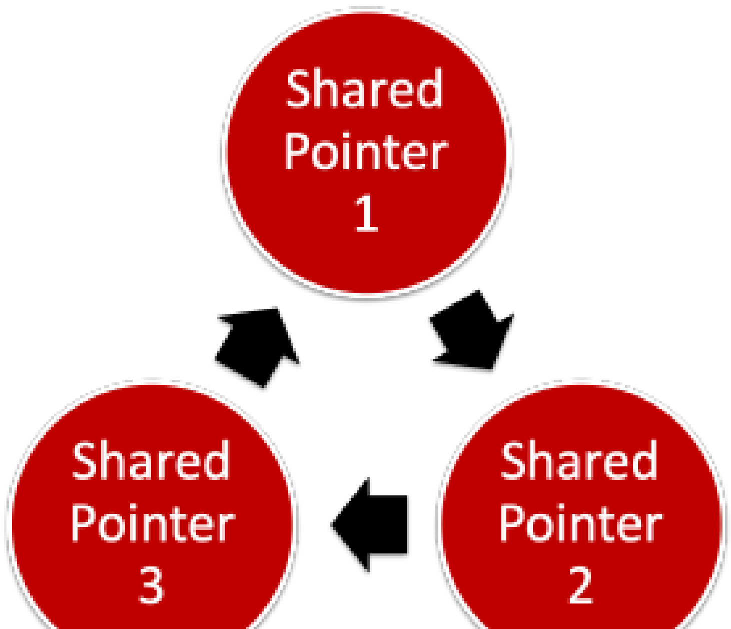
In this lesson, we'll examine how the use of shared pointers can create a reference cycle and why this could be harmful.

You get **cyclic references** of `std::shared_ptr` if they refer to each other.

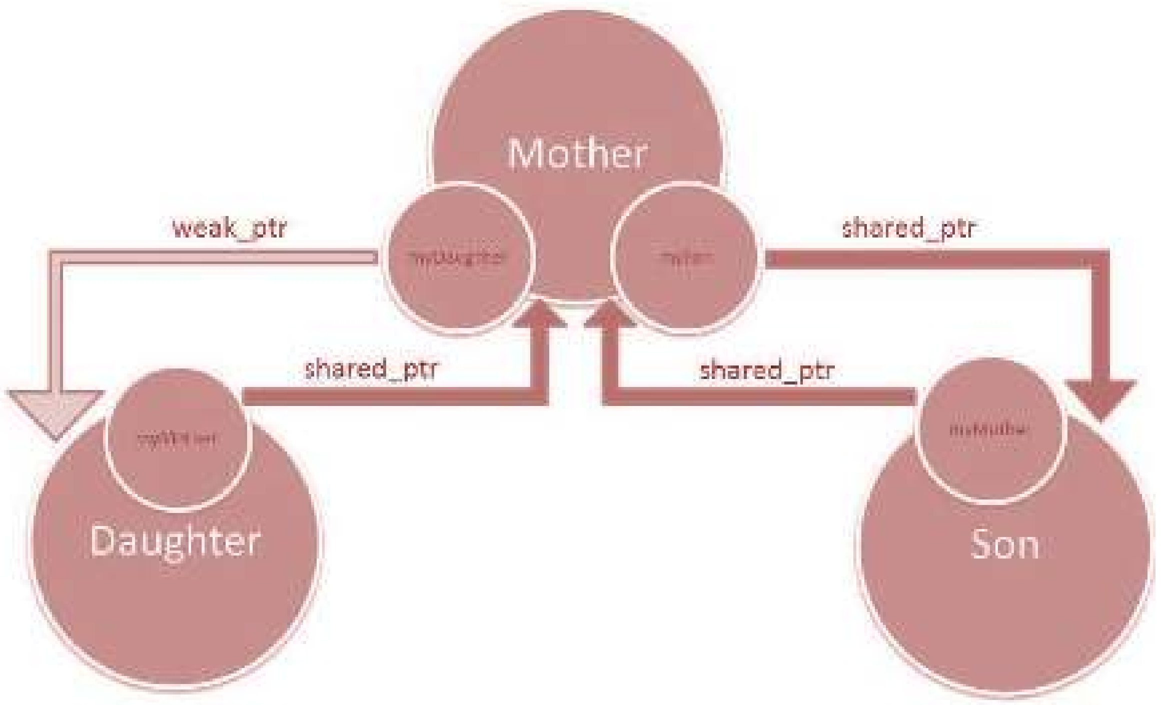
The Issue

If you have a cyclic reference of `std::shared_ptr`, the reference counter will never become 0. You can break this cycle if you embed an `std::weak_ptr` in the cycle. `std::weak_ptr` does not modify the reference counter.

> Theoretically, you can use a raw pointer to break the cycle of `std::shared_ptr`'s, but a raw pointer has two disadvantages. First, a raw pointer has no well-defined interface. Second, a raw pointer does not support an interface to create a `std::shared_ptr` out of it.



There are two cycles in the graphic below: first, between the mother and her daughter; second, between the mother and her son. The subtle difference is that the mother references her daughter with an `std::weak_ptr`. Therefore, the `std::shared_ptr` cycle is broken.



To better understand, take a look at the corresponding source code in the example in the following lesson.