Clean Coders Hate What Happens To Your Code When You Use These Enterprise Programming Tricks

@KevlinHenney FizzBuzzFeed

Singleton Configuration
Singletons
Noisy Logging
Log And Throw
Repetition And Duplication
Unnecessary Code
Mixed Levels Of Abstraction
Legacy Coding Habits
Programming By Coincidence
Programming By Superstition

```
Connection * CreateServerConnection()
{
    // Declarations
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;

    // Get address and check that its OK (throw an exception if its not)
    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Convert adress to bytes and check that its OK (throw an exception if its not)
    address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Get port and check that its OK (throw an exception if its not)
    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Convert port too bytes
    port = htons(atoi(cfgPort.data()));

    // Creation connection and check that its OK (throw an exception if its not)
    result = new Connection(address, port);
    if(!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Return the connection
    return result;
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Get address and check that its OK (throw an exception if its not)
    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Convert adress to bytes and check that its OK (throw an exception if its not)
    address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Get port and check that its OK (throw an exception if its not)
    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Convert port too bytes
    port = htons(atoi(cfgPort.data()));
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Creation connection and check that its OK (throw an exception if its not)
    result = new Connection(address, port);
    if(!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Return the connection
    return result;
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    ...

    // Get address and check that its OK (throw an exception if its not)
    ...

    // Convert adress to bytes and check that its OK (throw an exception if its not)
    ...

    // Get port and check that its OK (throw an exception if its not)
    ...

    // Convert port too bytes
    ...

    // Creation connection and check that its OK (throw an exception if its not)
    ...

    // Return the connection
    ...
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    ...

    // Get address and check that it's OK (throw an exception if it's not)
    ...

    // Convert address to bytes and check that it's OK (throw an exception if it's not)
    ...

    // Get port and check that it's OK (throw an exception if it's not)
    ...

    // Convert port to bytes
    ...

    // Creation connection and check that it's OK (throw an exception if it's not)
    ...

    // Return the connection
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...

    ...

    ...

    ...

    ...

    ...

    ...
}
```

```cpp
Connection * CreateServerConnection()
{
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;

    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    port = htons(atoi(cfgPort.data()));

    result = new Connection(address, port);
    if(!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    std::string cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    unsigned long address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    std::string cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    unsigned short port = htons(atoi(cfgPort.data()));

    Connection * result = new Connection(address, port);
    if(!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```cpp
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.data()));

    Connection * result = new Connection(address, port);
    if(!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    ...
    Connection * result = new Connection(address, port);
    if(!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    ...
    Connection * result = new Connection(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```cpp
std::auto_ptr<Connection> CreateServerConnection()
{
    ...
    std::auto_ptr<Connection> result(new Connection(address, port));
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
std::unique_ptr<Connection> CreateServerConnection()
{
    ...
    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    ...
    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.data()));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.data());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.data()));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.c_str()));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        snprintf(buffer, sizeof buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
    {
        snprintf(buffer, sizeof buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        snprintf(buffer, sizeof buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        snprintf(buffer, sizeof buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];
    ...
    if(cfgAddress.empty())
    {
        snprintf(buffer, sizeof buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
    if(address == -1)
    {
        snprintf(buffer, sizeof buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    if(cfgAddress.empty())
    {
        std::stringstream buffer;
        buffer << "Configuration value missing: " << "address";
        Log::Instance().Write(buffer.str());
        throw ConnectionException(buffer.str());
    }
    ...
    if(address == -1)
    {
        std::stringstream buffer;
        buffer << "Invalid address: " << cfgAddress;
        Log::Instance().Write(buffer.str());
        throw ConnectionException(buffer.str());
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    if(cfgAddress.empty())
    {
        static const char * logMessage = "Configuration value missing: address";
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }
    ...
    if(address == -1)
    {
        auto logMessage = "Invalid address: " + cfgAddress;
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        static const char * logMessage = "Configuration value missing: address";
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
    {
        auto logMessage = "Invalid address: " + cfgAddress;
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        static const char * logMessage = "Configuration value missing: port");
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        auto logMessage = "Failed to connect: " + cfgAddress + ":" + cfgPort;
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    return result.release();
}
```

```cpp
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
    {
        FailedToConnect("Configuration value missing: address");
    }

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
    {
        FailedToConnect("Invalid address: " + cfgAddress);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
    {
        FailedToConnect("Configuration value missing: port");
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
    {
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);
    }

    return result.release();
}
```

```cpp
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result.release();
}
```

```
std::unique_ptr<Connection> CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if(cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if(cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().ValueOf("address");
    if(cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().ValueOf("port");
    if(cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```cpp
std::unique_ptr<Connection> CreateServerConnection()
{
    auto cfgAddress = Configuration::Instance().ValueOf("address");
    if(cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = Configuration::Instance().ValueOf("port");
    if(cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> CreateServerConnection(
    const std::string & cfgAddress, const std::string & cfgPort)
{
    if(cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if(address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    if(cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> CreateServerConnection(
    in_addr_t address, in_port_t port)
{
    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect(address, port);
    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer(in_addr_t address, in_port_t port)
{
    auto result = std::make_unique<Connection>(address, port);
    if(!result->IsOK())
        FailedToConnect(address, port);
    return result;
}
```

```cpp
std::unique_ptr<Connection> ConnectToServer(in_addr_t address, in_port_t port)
{
    return std::make_unique<Connection>(address, port);
}
```

Enterprise Coders Should Love What Happens To Their Code With These Refactoring Tricks

@KevlinHenney FizzBuzzFeed