**Thợ lành nghề #14: Giao dịch (SMCRemote – phần 4)**

“OK, cao thủ -2- xem thử cậu thế nào.”

Độ căng thẳng trong cái nhìn của Jasmine làm tôi dán chặt vào ghế. “Ý… ý cô thế nào?” tôi lắp bắp.

“Thôi đi cao thủ, bộ cậu định không làm tôi quê như cậu đã làm Jerry quê sao,” nàng nói.

“Tôi chẳng cố tình làm cho ai quê cả,” tôi chống chế một cách yếu ớt. “Tôi chỉ….” ”

Ừa, hẳn nhiên rồi,” nàng bất chợt ngắt ngang câu nói của tôi, tỏ vẻ chán chường.

“Thôi, hãy bắt tay vào công việc cho rồi. Cậu định sẽ thay đổi những gì tiếp theo đây?”

Chúng tôi ngồi trong phòng làm việc, xem xét đoạn mã Jerry và tôi vừa viết xong. Tôi chỉ cho Jasmine cách tôi thay đổi đoạn mã nguồn của Jerry dùng để gửi chuỗi thay vì đối tượng đi qua socket.

“Tôi… ừm… ờ không biết. Tôi chỉ nghĩ là gửi đối tượng chắc tốt hơn chuỗi.” Nàng làm tôi hết sức bối rối. Mức căng thẳng cứ đổ dồn từ ánh mắt và thái độ của nàng.

“Suy nghĩ đi cao thủ, suy nghĩ! Cậu không chỉ thuần tuý gói vài cái chuỗi và số nguyên vào trong một đối tượng, phải thế không? Gói như vậy ngầm định vấn đề gì? Cậu có thể làm gì với nó?”

“Tôi, ừm….” giá như đôi mắt nàng ngưng đè nặng lên tôi, có lẽ tôi có thể suy ngẫm. Tôi nhắm nghiền đôi mắt và thầm tụng một đoạn kinh -3-. Trong vòng vài giây, tôi đã có thể xem xét câu hỏi của nàng.

Cái trường hợp kiểm thử chúng tôi đã làm dùng để xác thực liệu chúng tôi có thể gởi tệp tin có xuyên qua socket. Đoạn mã dùng để gởi tệp tin như thế này:

private void writeSendFileCommand() throws IOException {

os.writeObject(“Sending”);

os.writeObject(itsFilename);

os.writeLong(itsFileLength);

char buffer[] = new char[(int) itsFileLength];

fileReader.read(buffer);

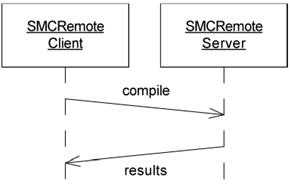
os.writeObject(buffer);

os.flush();

}

Nhưng tại sao chúng tôi lại gửi tệp tin đi? Chúng tôi gửi nó đến SMCRemoteServer để biên dịch. Sau đó máy chủ sẽ trả về tệp tin đã được biên dịch. Tại sao Jerry lại gửi chuỗi “Sending” trước? Gã nói rằng mục đích là để báo máy chủ có tệp tin đang được gửi đến – nhưng chúng tôi lại không muốn thông báo cho máy chủ là có tệp tin đang được gửi đến; chúng tôi muốn ra lệnh cho máy chủ biên dịch một tệp tin và gửi ngược lại kết quả.

Tôi suy nghĩ rất kỹ lưỡng, nhưng một phần nào đó trong não bộ của tôi vẫn đang tiếp tục tụng kinh. Hầu như trong trạng thái nửa tỉnh, nửa mê, tôi đi thẳng đến bức tường và vẽ ra sơ đồ “kết quả biên dịch”.



Tôi nhác thấy khuôn mặt nghiêm trọng của Jasmine thoáng một nụ cười. “Tôi khoái cái lối suy nghĩ của cậu đó cao thủ. Đừng dừng lại ở đó.”

Bốn mẩu dữ liệu được gửi đến server: tên tệp tin, độ dài tệp tin, nội dung tệp tin và chuỗi “Sending”. Tại sao những mẩu này được gửi riêng biệt? Chúng đều thuộc vào một gói tin của một suất chuyển tải! Đúng rồi! Tôi tự lắc đầu với chính mình và thay đổi đoạn kiểm thử như sau:

public void testCompileFile() throws Exception {

File f = createTestFile(“testSendFile”, “I am sending this file.”);

c.setFilename(“testSendFile”);

assertTrue(c.connect());

assertTrue(c.prepareFile());

assertTrue(c.compileFile());

Thread.sleep(50);

assertTrue(server.fileReceived);

assertEquals(“testSendFile”, server.filename);

assertEquals(23, server.fileLength);

assertEquals(“I am sending this file.”, new String(server.content));

f.delete();

}

Thế rồi tôi đổi hàm sendFile cũ như sau:

public boolean compileFile() {

boolean fileSent = false;

char buffer[] = new char[(int) itsFileLength];

try {

fileReader.read(buffer);

CompileFileTransaction cft = new CompileFileTransaction(itsFilename, buffer);

os.writeObject(cft);

os.flush();

fileSent = true;

} catch (Exception e) {

fileSent = false;

}

return fileSent;

}

Jasmine theo dõi rất sát sao. Tôi không thể dò nổi cảm giác của nàng nhưng tôi biết chắc là mình đang đi đúng hướng. Kế tiếp tôi viết lớp CompileFileTransaction:

public class CompileFileTransaction implements Serializable {

private String filename;

private char contents[];

public CompileFileTransaction(String filename, char buffer[]) {

this.filename = filename;

this.contents = buffer;

}

public String getFilename() {

return filename;

}

public char[] getContents() {

return contents;

}

}

Đoạn này cho phép chương trình được biên dịch. Tất nhiên là mấy cái kiểm thử bị hỏng, bởi thế tôi lại thay đổi phần máy chủ giả như sau:

public void serve(Socket socket) {

try {

os = new PrintStream(socket.getOutputStream());

is = new ObjectInputStream(socket.getInputStream());

os.println(“SMCR Test Server”);

os.flush();

parse(is.readObject());

} catch (Exception e) {

}

}

private void parse(Object cmd) throws Exception {

if (cmd != null) {

if (cmd instanceof CompileFileTransaction) {

CompileFileTransaction cft = (CompileFileTransaction) cmd;

filename = cft.getFilename();

content = cft.getContents();

fileLength = content.length;

fileReceived = true;

}

}

}

Những thay đổi này giúp cho các kiểm thử đều đạt. “Phải ý cô giống như thế này không?” Tôi hỏi.

“Ừa, chỉ là khởi điểm thôi,” nàng xác nhận một cách dè chừng. “Chắc chắn là nó hay hơn lối chuyển mỗi phần dữ liệu thành chuỗi của Jerry – và nó cũng hay hơn lối chuyển gửi mỗi phần dữ liệu riêng biệt.”

“Vậy cô làm thế nào cho hay hơn nữa vậy?” Tôi hỏi.

“Hẳn đã,” nàng nói một cách thiếu kiên nhẫn. “Ngay lúc này hãy hoàn tất phần chuyển tải. Cậu phải làm cho client tiếp nhận hồi đáp từ server.”

“Cái đó chắc không khó lắm,” tôi đáp, cảm thấy phấn chấn hơn một chút, và thêm vào ba dòng như sau vào đoạn testCompileFile như sau:

File resultFile = new File(“resultFile.java”);

assertTrue(“Result file does not exist”, resultFile.exists());

resultFile.delete();

Tôi chạy đoạn test và xác thật nó bị hỏng.

“Sau khi mình gọi compileFile, kết quả hẳn phải được viết vào một tệp tin,” tôi giải thích cho Jasmine, rồi nói thêm, “ngay lúc này tôi không quan tâm đến chuyện có gì trong tệp tin; tôi chỉ muốn chắc là tệp tin đó được tạo ra.”

“Vậy cậu làm cách nào để tạo ra nó?” nàng thách thức.

“Tôi sẽ cho cô thấy,” tôi nói, thay đổi đoạn máy chủ giả như sau:

private void parse(Object cmd) throws Exception {

if (cmd != null) {

if (cmd instanceof CompileFileTransaction) {

CompileFileTransaction cft = (CompileFileTransaction) cmd;

filename = cft.getFilename();

content = cft.getContents();

fileLength = content.length;

fileReceived = true;

CompilerResultsTransaction crt = new CompilerResultsTransaction(“resultFile.java”);

os.writeObject(crt);

os.flush();

}

}

}

Thế rồi tôi tạo phần biên dịch này bằng cách thêm một cái sườn của CompileResultsTransaction class

public class CompilerResultsTransaction implements Serializable {

public CompilerResultsTransaction(String filename) {

}

public void write() {

}

}

Tất nhiên phần test vẫn hỏng, bởi thế tôi thay đổi compileFile như sau:

public boolean compileFile() {

boolean fileCompiled = false;

char buffer[] = new char[(int) itsFileLength];

try {

fileReader.read(buffer);

CompileFileTransaction cft = new CompileFileTransaction(itsFilename, buffer);

os.writeObject(cft);

os.flush();

Object response = is.readObject();

CompilerResultsTransaction crt = (CompilerResultsTransaction) response;

crt.write();

fileCompiled = true;

} catch (Exception e) {

fileCompiled = false;

}

return fileCompiled;

}

Cuối cùng, tôi thực hiện chi tiết phần chuyển tải:

public class CompilerResultsTransaction implements Serializable {

private String filename;

public CompilerResultsTransaction(String filename) {

this.filename = filename;

}

public void write() throws Exception {

File resultFile = new File(filename);

resultFile.createNewFile();

}

}

Kết quả biên dịch

Tại sao tên, độ dài, nội dung của tệp tin và “Sending” string đều được gởi đến máy chủ riêng biệt nếu chúng đều thuộc về một chặng chuyển tải?

“Ở giai đoạn này được vậy là tốt rồi,” Jasmine nói. “Tôi đi giải lao một chốc trong khi cậu thực hiện xong quy trình CompileResultsTransaction thực sự viết thành tệp tin thay vì chỉ tạo ra nó. Cũng nên dọn dẹp chút đỉnh nữa. Có khá nhiều mảnh vụn vặt còn sót lại trong lúc cậu và Jerry khuấy vọc chuyện gởi chuỗi và integers. Nhưng trước khi tôi đi, tôi muốn biết ý kiến của cậu trong phần instanceof cậu dùng trong đoạn máy chủ giả.”

Đó là giải pháp đơn giản nhất mà thôi có thể nghĩ ra dùng để kiểm tra xem object sắp trả lại có thật sự là CompileFileTransaction hay không.” Tôi nói, bắt đầu cảm thấy bối rối. “Có gì sai với phần này sao?”

Nàng đứng lên, nhìn về phía tôi và trả lời, “không có gì sai trầm trọng, nhưng cậu có nghĩ rằng máy chủ thật sẽ làm thế sao? Liệu máy chủ thật sẽ có chuỗi if/else dài ngoằng cho instanceof để mà biến xuất các chuyển tải đi vào?”

“Tôi chưa nghĩ xa đến như thế,” tôi thú nhận. “Không,” nàng nói một cách thẳng thừng, “Tôi không hình dung cậu nghĩ xa như vậy.” Và rồi nàng rảo bước ra khỏi phòng.

Căn phòng trống rỗng khi không có nàng, như thể sự hiện diện của tôi chẳng có giá trị gì. Tôi thở dài và lắc lắc cái đầu. Làm việc với Jasmine sắp tới sẽ đầy mệt mỏi và đầy sự giáo huấn đủ mọi kiểu. Một điều tôi biết chắc – tôi ghét bị gọi là cao thủ. Tôi lại thở dài và bắt đầu giải quyết công tác nàng giao cho.

*-1- Dựa trên góp ý của cl trong bài thứ 13 , journeyman có thể được xem như những kẻ du mục, đi tìm những vùng “đất mới”. Nghĩa bóng cho journeyman cũng hết sức thích hợp cho những lập trình viên có cái nhìn khai phá. Tôi tạm dịch journeyman là du mục theo tinh thần này.*

*-2- Hotshot: tiếng lóng chỉ cho một cá nhân kinh nghiệm và nổi bật. Hotshot cũng có thể dùng với tính cách châm biếm, bỡn cợt hoặc thân thiện. Trong bài này, có lẽ Jasmine gọi Alphonse với tính cách bỡn cợt.*

*-3- Mantra: có nghĩa chung là đoạn kinh kệ. Theo đạo Hindu và đạo Phật, mantra có khả năng hoá giải những trắc trở.*

**Bài tiếp:** [Thợ lành nghề #15: Ếch là Bê (SMCRemote – phần 5)](https://codegym.vn/blog/2018/08/13/tho-lanh-nghe-15-ech-la-be-smcremote-phan-5/)

**Bài trước:** [Thợ lành nghề #13: Đối tượng (SMCRemote – phần 3)](https://codegym.vn/blog/2018/08/11/tho-lanh-nghe-13-doi-tuong-smcremote-phan-3/)

**Tác giả:** Robert C. Martin

**Người dịch:** Hoàng Ngọc Diêu (conmale)