# **Thợ lành nghề #30: Phòng trừng phạt (Giám sát liều lượng – Phần 7)**

**Robert C. Martin**

**3/9/2004**

*(…Tiếp nối phần trước)*

*Khi năm 1939 dần trôi qua, và châu Âu chìm vào thời chiến tranh hỗn loạn, Trái Đất vẫn là trọng tâm trong quãng đường bay ngày càng thu hẹp khoảng cách của Clyde. Mặc dù xác suất va chạm vẫn còn khá nhỏ nhưng vẫn không ngừng tăng cao. Các thành viên của Đội ngũ Stockholm đang liên lạc thông qua mạng lưới bí mật của Lise Meitner rốt cuộc kết luận rằng cách giải quyết đúng mực duy nhất là hành động như thể sự va chạm là điều chắc chắn.*

*Hội đoàn rơi vào thế tiến thoái lưỡng nan. Cho đến giờ họ đã cố gắng giữ cho việc phát hiện phân hạch Uranium chỉ trong phạm vi của hội đoàn. Tuy nhiên, nếu sức mạnh nguyên tử được sử dụng như một vũ khí phòng thủ chống lại Clyde, nó sẽ đòi hỏi nguồn lực của một quốc gia rất giàu có để đạt được công nghệ thiết yếu kịp lúc. Ở châu Âu, tất cả các quốc gia như thế đều đang có chiến sự, và chắc chắn sẽ sử dụng công nghệ đó để tạo ra một thứ vũ khí rất khủng khiếp.*

*Hội đoàn quyết định đi đến nơi mà chiến tranh chưa tràn tới. Leo Szilard, một thành viên sáng lập hội đoàn, đã thuyết phục Albert Einstein nhân danh hội đoàn viết thư cho Franklin Roosevelt. Tầm quan trọng của bức thư càng tăng lên bởi trên thực tế phần đông thành viên hội đoàn đã xoay sở bằng cách nào đó đến được đất Mỹ cùng thời điểm lá thư được gửi đi.*

***21/02/2002, 1230***

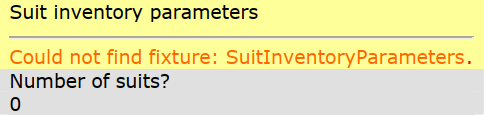
Jean đưa Avery tới phòng đối thoại nhỏ cho dành hai người với tường bằng kính. Họ vẫn ở đó cho tới khi Jerry và tôi trở về sau bữa trưa. Vẻ mặt của Avery làm tôi mừng rằng tôi không phải là người trong phòng đó với Jean.

Jerry nói: “Tốt hơn hết chúng ta nên trở lại làm việc. Hãy xem chúng ta tiến triển được bao nhiêu khi nó không có ở đây.”

“Anh nghĩ cậu ấy sẽ quay lại chứ?” Tôi hỏi.

Jerry thoáng liếc nhìn Avery và Jean với vẻ mặt hiểu biết. “Tao khá chắc là như thế, Alphonse. Giờ thì đi làm việc thôi.”

Chúng tôi ngồi tại khu làm việc của chúng tôi và Jerry chạy kiểm thử chấp nhận RegisterNormalSuit. Hai bảng đầu tiên không có gặp lỗi, nhưng cái thứ ba lại trông như thế này:



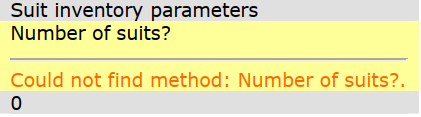
“Vây chúng ta cần một fixture có tên SuitlnventoryParameters, đúng chứ?” Tôi hỏi.

“Phải.” Jerry nói. “Lại đây và để tao xem mày có nhớ cách viết nó không.”

Thế nên tôi cầm lấy bàn phím và gõ như sau:

| 1  2 | public class SuitInventoryParameters extends ColumnFixture {<em>  }</em> |
| --- | --- |

Giờ thì quá trình chạy kiểm thử chấp nhận đưa ra kết quả sau:



“Nó không hẳn là những gì tôi trông đợi.” Tôi nói.

“Lần cuối chúng ta làm điều này nó không hề đề cập đến phương thức. Nó đã muốn một biến.”

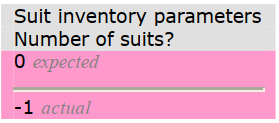
“Đúng thế, Alphonse, nhưng lần này nó muốn một phương thức bởi vì có một dấu chấm hỏi ngay sau cái tên.”

“Ồ, đúng rồi, anh trước đó đã đề cập đến thứ gì đó về việc này vào sáng nay. Những dấu chấm hỏi đồng nghĩa với việc bảng đang đặt câu hỏi về ứng dụng. Vậy nên chúng ta đang hỏi hệ thống rằng nó có bao nhiêu bộ đồ trong kho hàng?”

“Chính xác. Thế nên làm tiếp đi và viết phương thức đó”

| 1  2  3  4  5 | public class SuitInventoryParameters extends ColumnFixture {  public int numberOfSuits() {<strong>  return -1;  }  </strong>} |
| --- | --- |

Lúc này kiểm thử chấp nhận trông như thế này:



“Tốt lắm.” Jerry nói. “Giờ hãy kết nối fixture với ứng dụng.”

“Hàm nào trong ứng dụng nhận được số lượng bộ đồ trong kho hàng vậy?” Tôi hỏi.

“Hỏi hay lắm.” Jerry đáp. “Mày nghĩ hàm đó sẽ hàm nào?”

Tôi kiểm tra mã trong dự án một lúc và nói: “Từ giờ chúng ta có thể để nó trong lớp Utilities.”

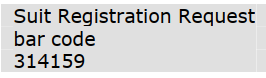
“Có thể như thế. Dù rằng tao nghĩ nó sẽ chẳng ở đó được bao lâu.”

Tôi chỉnh mã như sau:

| 1  2  3  4 | public class SuitInventoryParameters extends ColumnFixture {  public int numberOfSuits() {  return Utilities.getNumberOfSuitsInInventory();  } } public class Utilities { public static Date testDate = null; public static Date getDate() { return testDate != null ? testDate : new Date(); } public static int getNumberOfSuitsInInventory() {<strong> return -1; } </strong>}Không có gì thay đổi trong kết quả kiểm thử chấp nhận. “Tôi có nên làm bảng này qua được luôn không?” Tôi hỏi Jerry. |
| --- | --- |

“Không, hãy kết nối các fixture khác lại trước đã.” Hắn ta đáp.

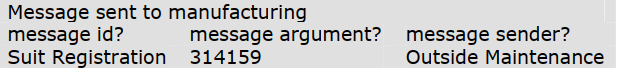
Bảng tiếp theo trong kiểm thử nhìn như thế này:



Kết nối nó với FitNesse thật dễ dàng.

| 1  2  3  4  5  6  7  8  9  1 | public class SuitRegistrationRequest extends ColumnFixture {  public int barCode;  public void execute() {  Utilities.registerSuit(barCode);  }  }  public class Utilities {  …  public static void registerSuit(int barCode) {<strong>  }  </strong>} |
| --- | --- |

Bảng kế tiếp trông có chút phức tạp hơn một chút. Trông như vậy:



Cấu trúc cơ bản của các fixture cũng ở mức độ dễ. Các dấu chấm hỏi chỉ cho tôi biết rằng mỗi một tiêu đề cột là một phương thức. Vậy nên tôi viết như sau:

| 1  2 | public class MessageSentToManufacturing extends ColumnFixture { public String messageId() { return null; } public int messageArgument() { return -1; }  public String messageSender() { return null; } } |
| --- | --- |

Nhưng sau đó tôi lâm thế bí. “Làm thế nào để tôi kết nối nó với ứng dụng đây?” Tôi hỏi.

“Mày có nhớ bảng này đang kiểm tra gì không?”

“Đương nhiên, chúng ta đang xác minh nội dung tin nhắn mà DTrack được cho là đã gửi tới hệ thống sản xuất.”

“Đúng thế. Nên mày cần có được tin nhắn đó, và giải nén nó.”

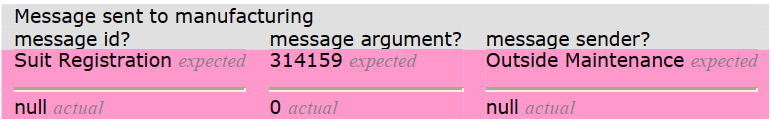
“Tôi làm điều đó bằng cách nào đây? Không có phương thức nào trong fixture này nhìn có vẻ là một nơi thích hợp.”

“Tao đồng ý. Không phải chúng. Tuy nhiên, FitNesse cung cấp cho mày một lựa chọn khác. Phương thức thực hiện trong ColumnFixture được gọi trước khi các tiêu đề cột được gọi. Thế nên trong phương thức thực hiện mày có thể yêu cầu DTrack cho một bản sao của tin nhắn đã được gửi, và sau đó các phương thức tiêu đề cột có thể giải nén tin nhắn đó.”

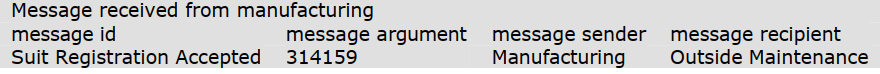
“OK, tôi nghĩ tôi hiểu ra rồi.” Và tôi thay đổi mã như sau:

| 1  2  3  4  5  6  7  8  9  1 | public class MessageSentToManufacturing extends ColumnFixture {  private SuitRegistrationMessage message;<strong>  public void execute() throws Exception {  message = (SuitRegistrationMessage)  Utilities.getLastMessageToManufacturing();  }  </strong>public String messageId() {  return message.id;  }  public int messageArgument() {  return message.argument;  }  public String messageSender() {  return message.sender;  }  }  public class SuitRegistrationMessage {  public String id;  public int argument;  public String sender;  }  public class Utilities {  …  public static Object getLastMessageToManufacturing() {<strong>  return new SuitRegistrationMessage();  }  </strong>} |
| --- | --- |

Điều này làm cho bảng thành như thế này:



Bảng kế tiếp lại quá đơn giản. Nó trông như thế này:



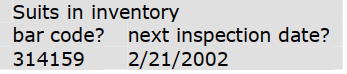
Tôi kết nối nó với DTrack bằng cách dùng fixture sau:

| 1  2  3  4  5  6  7  8  9  1 | public class MessageReceivedFromManufacturing extends ColumnFixture {  public String messageId;  public int messageArgument;  public String messageSender;  public String messageRecipient;  public void execute() {  SuitRegistrationAccepted message =  new SuitRegistrationAccepted(messageId,  messageArgument,  messageSender,  messageRecipient);  Utilities.acceptMessageFromManufacuring(message);  }  }  public class SuitRegistrationAccepted {  String id;  int argument;  String sender;  String recipient;  public SuitRegistrationAccepted(String id, int argument,  String sender, String recipient) {  this.id = id;  this.argument = argument;  this.sender = sender;  this.recipient = recipient;  }  }  public class Utilities {  …  public static void acceptMessageFromManufacuring(Object message) {}  } |
| --- | --- |

“Lớp Utilities đang gom một lượng lớn mã thừa.” Tôi than vãn.

“Đúng là nó như thế thật. Chúng ta sẽ quay lại và tái cấu trúc nó ngay khi nhận được kiểm thử chấp nhận đã thành công. Nhưng trước mắt làm cho bảng cuối cùng được kết nối đã.”

Tôi thở dài và nhìn vào cái bảng cuối cùng. Nó có một chút khác biệt:



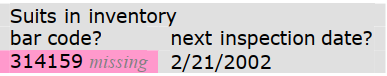
“Có vẻ như bảng này có thể yêu cầu nhiều hơn một bộ đồ.” Tôi nói.

“Chà, mỗi bảng đòi hỏi duy nhất một bộ đồ, nhưng một trong những phương thức không mong đợi đó là số lượng bộ đồ không phải chỉ có một. Mày phải sử dụng thể loại fixture khác cho việc này. Để tao chỉ cho mày thấy.”

Jerry chộp lấy bàn phím và viết như sau:

| 1  2  3  4  5  6  7  8  9  1 | public class SuitsInInventory extends RowFixture {  public Object[] query() throws Exception {  return Utilities.getSuitsInInventory();  }  public Class getTargetClass() {  return Suit.class;  }  }  public class Suit {  public Suit(int barCode, Date nextInspectionDate) {  this.barCode = barCode;  this.nextInspectionDate = nextInspectionDate;  }  private int barCode;  private Date nextInspectionDate;  public int barCode() {  return barCode;  }  public Date nextInspectionDate() {  return nextInspectionDate;  }  }  public class Utilities {  …  <strong>public static Suit[] getSuitsInInventory() {  return new Suit[0];  }  </strong>} |
| --- | --- |

Khi Jerry chạy kiểm thử, bảng có dạng sau:



Tôi nhìn đoạn mã này một hồi và nói: “Tôi nghĩ tôi hiểu rồi. Anh đã ghi đè phương thức query() của SuitsInInventory để trả về một mảng các đối tượng Suit. Anh cũng viết đè phương thức getTargetClass() để trả về lớp Suit. Bất kì mục nào được liệt kê trong bảng nhưng không hiện hữu trong mảng sẽ được đánh dấu là thiếu.”

“Đúng rồi.” Jerry nói. “Hơn nữa, nếu như phương thức query() trả về nhiều hơn một đối tượng Suit, nó cũng sẽ được đánh dấu là bổ sung.”

“OK, giờ thì chúng ta có được toàn bộ trang thử nghiệm được kết nối với ứng dụng DTrack. Đến lúc làm nó thành công rồi.”

“Chính xác. Tao cá là chúng ta có thể làm nó trước khi Avery ra khỏi phòng trừng phạt.”

“Phòng trừng phạt?”

“Đó là cách bọn tao gọi cái phòng đối thoại có lớp tường mỏng kia.”

Tôi nhìn qua Avery và Jean ở trong phòng trừng phạt. Trông giống như một cuộc nói chuyện một chiều khá dữ dội.

*Còn tiếp…*