# [[Học nghề] Đai trắng](https://codegym.vn/blog/2018/07/27/hoc-nghe-dai-trang/#respond)



*Như một luật lệ, mỗi bước đi nên có cảm giác như vừa mới bắt đầu. Đó là tư tưởng của người mới – trạng thái phát triển.*

*-Shunryu Suzuki, Zen Mind, Beginner’s Mind-*

## **Bối cảnh**

Bạn đã tích lũy được một lượng kiến thức khá tốt về một ngôn ngữ đầu tiên và đang tự tin về năng lực của mình. Đồng nghiệp nhận ra khả năng của bạn và nhờ bạn giải quyết các vấn đề thuộc lĩnh vực chuyên môn của mình. Bạn cảm thấy tự hào về khả năng của mình.

## **Vấn đề**

Bạn đang vật lộn để học những thứ mới và có vẻ như việc học thêm các kỹ năng mới trở nên khó khăn hơn so với trước. Tốc độ học của bạn có vẻ như đang chậm lại mặc dù đã cố gắng và nỗ lực hết sức. Bạn đang lo rằng vấn đề phát triển bản thân của mình đang diễn ra chậm chạp.

## **Giải pháp**

Khi bước vào một tình huống mới, bạn cần giữ được sự tự tin tích lũy được trong quá trình học tập, nhưng đồng thời cũng cần gạt bỏ những kiến thức đã học qua một bên. Như Yoda đã nói rõ trong bộ phim The Empire Strikes Back (Đế chế phản công): “Bạn phải bỏ những gì đã học”.

Sử dụng đai trắng dựa trên việc nhận ra rằng trong khi đai đen “biết” đã biết cách đi thì đai trắng không còn cách nào khác ngoài việc “học” cách đi.

Một phần trong cách tiếp cận Dave dùng như một nhà trị liệu gia đình bao gồm việc duy trì một lập trường không biết. Khi gia đình gặp phải hoàn cảnh khó khăn thực tế, cho dù đã được đào tạo tốt Dave biết anh ấy không thể thấy rõ hoàn toàn mọi thứ. Dù Dave thừa nhận rằng mình có kỹ năng trong việc tạo điều kiện thuận lợi cho mọi người đặt câu hỏi và đoạn hội thoại có tính xây dựng, nhưng Dave được dạy là phải kiềm chế và tin là mình chưa có bất cứ kiến thức chuyên môn ở trong tình huống thực tế mà những gia đình này đã trải qua.Trong khi điều này có thể có vẻ phản trực giác, thật ra nó thúc đẩy thái độ tôn trọng và tính hiếu kỳ để mở ra những khả năng và giải pháp không đoán trước được. Thay vì áp đặt các giải pháp cho gia đình, lập trường không biết của Dave đã giúp ông hợp tác với gia đình để tìm các giải pháp theo nhóm.

Sử dụng cách tiếp cận này để học công nghệ mới sẽ đẩy nhanh quá trình học lên rất nhanh. Tự nhắc nhở bản thân luyện tập để ngăn những thói quen lập trình cũ lại cho phép bạn khám phá ra khả năng mới. Tuy nhiên, dưới góc độ là lập trình viên cảm thấy tự hào về trình độ chuyên môn đáng kể mà mình đã đạt được cuối cùng, việc thực hiện một bước tiến tới “sự ngu dốt” và cho phép mình trông ngu dốt có thể rất đau đớn. Những hãy xem xét những từ ngữ của George Leonard trong những trang cuối của cuốn *Mastery* (sự tinh thông):

*Đã có bao nhiêu lần bạn không dám thử làm một cái gì đó vì sợ bị chê là ngớ ngẩn? Bao lâu một lần bạn ngăn không cho mình đưa ra những quyết định mới mẻ vì sợ bị nghĩ trẻ con? Nhà tâm lý học Abraham Maslow đã phát hiện tính trẻ con của những người có khả năng tiềm ẩn cao vượt trội. Ashleigh Montagu sử dụng thuật ngữ neotany (xuất phát từ từ neonate, nghĩa là trẻ mới sinh) để miêu tả thiên tài như là Mozart và Einstein. Nheo mày trước những hành động ngu ngốc của bạn bè hay chính mình khiến chúng ta có xu hướng cười vào một thiên tài nổi tiếng thế giới như kẻ lập dị thông thường và không bao giờ dừng lại để nghĩ rằng sự tự do để ngu ngốc có thể là một trong những chìa khóa thành công của các thiên tài.*

Hoặc lấy ví dụ sau đây của một cựu chiến binh 10 năm làm trong ngành công nghiệp này luôn giữ một tâm trí cởi mở và học tập một cái gì đó mới:

*Tôi đã viết phần mềm chuyên nghiệp và khá thành công trong khoảng 10 năm và TDD trong vòng một vài năm cho đến khi tôi biết đến cuốn sách “Working Effectively with Legacy Code” của Michael Feathers, Prentice Hall. Cuốn sách đó đã có tác động ngay tức khắc đến cách tôi viết code, và tôi đã sao chép thành nhiều bản gửi cho tất cả các lập trình viên trong công ty và yêu cầu họ đọc nó. Kể từ đó, mã nguồn của tôi đã dần dần được cải thiện thành một hệ thống kiểm thử tốt hơn, ít phụ thuộc lẫn nhau hơn, hệ thống dễ thích nghi hơn và thú vị hơn khi làm việc với nó.*

-Steve Smith, email-

Như Steve đã học được, chúng ta phải biết cách bỏ qua những kinh nghiệm và định kiến trong quá khứ để có thể tiếp thu kiến thức mới. Điều này đặc biệt khó khăn khi cố gắng học ngôn ngữ lập trình thứ hai, vì đây có thể là lần đầu tiên bạn phải hi sinh hiệu suất để nâng cao kỹ năng của mình. Trước đây chắc bạn sẽ tiếp cận các vấn đề như một chiếc ly rỗng, với vài khái niệm về “*cách đúng đắn*” để giải quyết chúng. Bây giờ bạn phải cố gắng tránh đồng nhất kiến thức mới và cũ cho đến khi bắt đầu ngấm kiến thức mới, và tiếp cận kiến thức mới với cái đầu của một người chưa biết gì. Điều này có thể sẽ làm giảm hiệu suất trong thời gian ngắn trước khi có thể thực hiện bước nhảy vọt khi bạn đã làm chủ được cách tiếp cận mới.

*Để có thể trèo, bạn phải bỏ đi những bước đi vững chắc, buông bỏ những gì bạn đã làm tốt và có thể khiến bạn trượt xuống vùng hẻm núi. Nếu bạn không bao giờ buông bỏ những gì bạn đã làm tốt, bạn có thể tiếp tục đạt được sự tiến bộ ổn định, nhưng bạn sẽ không bao giờ thoát khỏi vùng cao nguyên đó.*

-Jerry Weinberg, , Becoming a Technical Leader, p. 42-

Một trong những lợi ích của việc áp dụng tư duy này khi học ngôn ngữ mới, công cụ, hoặc kinh doanh là bạn hoàn toàn cởi mở để có thể học cách diễn đạt ý muốn của bản thân một cách tự nhiên, từ đó làm quá trình giao tiếp với tập thể được suôn sẻ hơn. Bạn nên tránh việc “*viết theo kiểu Fortran khi dùng bất cứ ngôn ngữ nào*” thì bạn hiểu sâu hơn nhiều về kiến thức mới. Vì vậy, khi bạn làm quen hoàn toàn với kiến thức cũ và mới bạn sẽ có khả năng để phát triển những hiểu biết tốt hơn trong cả hai lĩnh vực.

Lấy đoạn mã Java sau đây làm ví dụ. Nó tạo ra các số ngẫu nhiên cho chương trình xổ số Quốc gia Anh bằng cách in một tập gồm sáu số khác biệt từ 1 đến 49:

| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class Lottery {  private static final int NUMBER\_OF\_RANDOM\_NUMBERS = 6;  private static final int MAX\_RANDOM\_NUMBER = 49;  public static void main(String[] args) {  SortedSet randomNumbers = new TreeSet();  Random random = new Random();  while (randomNumbers.size() &lt; NUMBER\_OF\_RANDOM\_NUMBERS) {  Integer randomNumber = new Integer(random.nextInt(MAX\_RANDOM\_NUMBER) + 1);  randomNumbers.add(randomNumber);  }  System.out.println(randomNumbers);  }  } |
| --- | --- |

Nếu bạn được yêu cầu thực hiện lại bằng ngôn ngữ tương đồng như Io (Được thiết kế để giảm số lượng cú pháp tới mức ít nhất có thể, trong khi vẫn tiếp cận được với phần lớn các Lập trình viên), bạn có thể sử dụng lại nhiều kiến thức về Java và viết nó như sau:

| 1  2  3  4  5  6 | list := List clone  while (list size &lt; 6,  n := Random value(1 50) floor  list appendIfAbsent( n )  )  list sort print |
| --- | --- |

Nhưng nếu bạn được yêu cầu thực hiện nó bằng một ngôn ngữ hoàn toàn khác như J, bạn sẽ thấy cách tiếp cận này không khả thi. Chỉ bằng cách *“mặc chiếc đai trắng”* – chấp nhận rằng với một ngôn ngữ không có vòng lặp sẽ phải có một sự khác biệt hoàn toàn nhưng dù sao nhờ một cách giải quyết vấn đề phù hợp thể đạt được tiến bộ. Vì vậy, trong hệ ngôn ngữ J, câu trả lời là:

| 1 | sort 1 + (6 ? 49) |
| --- | --- |

Trong phần sau chúng tôi sẽ chỉ cho bạn các cách để giúp bạn thực hành, lập ra các chương trình đồ chơi và phản hồi liên tục về công việc bạn đang làm. Những mẫu này sẽ cho phép bạn đánh giá sâu hơn những điểm chung giữa các mảng kiến thức khác nhau mà bạn sở hữu, và tạo ra các tình huống mà bạn có thể mài giũa kỹ năng của mình mà không bị áp lực để có thể duy trì mức hiệu suất bình thường.

## **Hành động**

Hãy tìm một cơ hội để quên đi một thứ gì đó. Lý tưởng nhất là tìm điều gì đó ép bạn gạt bỏ lại những kinh nghiệm trước đây sang một bên.

Ví dụ, chọn ra một phần mềm mà bạn đã viết theo một mô hình lập trình nào đó (ví dụ: ra lệnh, hướng tới đối tượng, hướng chức năng, mảng/vector định hướng….) và triển khai nó bằng một ngôn ngữ sử dụng mô hình lập trình khác. Hãy chắc chắn rằng cách thực hiện mới phải tuân thủ đúng quy tắc của ngôn ngữ mới. Nếu tất cả các ngôn ngữ mà bạn biết đều dùng một mô thức (ví dụ như lập trình hướng đối tượng), vậy thì đây chính là cơ hội để bạn có thể học được một mô thức mới.

Mô hình này vượt xa các ngôn ngữ lập trình, nhưng nó lại là một lĩnh vực rất dễ gây hiểu nhầm. Vậy nên hãy tìm một người dùng một ngôn ngữ lập trình hoặc một tập hợp các công nghệ nào đó mà bạn không biết. Nhờ họ giải thích những hiểu lầm mà những người có chung nền tảng như bạn thường gặp phải.