บทที่ 1
ความรู้เบื้องต้นเกี่ยวกับ Selenium

1.1 บทนำ

นิยาม Scenario โดยปกติจะหมายถึงเรื่องราวที่แสดงรายละเอียดของสถานการณ์ที่เกิดขึ้น ดังนั้นเมื่อนำมาใช้กับนิยามของการทดสอบคำว่า Test Scenario จึงหมายถึงการทดสอบการทำงานของโปรแกรมเป็นไปตามสมมุติฐานหรือสถานการณ์ที่กำหนดไว้หรือไม่ โดยมีวัตถุประสงค์เพื่อให้การทดสอบเป็นไปอย่างถูกต้อง ในขณะที่นิยาม Test Case หมายถึงกลุ่มของตัวแปรหรือเงื่อนไขที่ถูกกำหนดขึ้นจากนักทดสอบตามความต้องการของระบบที่กำหนดไว้ นอกจากนั้น Test Case ยังประกอบไปด้วยรายละเอียดของข้อมูลนำเข้าสู่ระบบ การกระทำหรือลำดับของเหตุการณ์ที่เกิดขึ้น รวมไปถึงผลลัพธ์ที่คาดไว้ เพื่อใช้สำหรับทดสอบการทำงานของระบบที่ถูกพัฒนาขึ้น ในมุมมองของนักทดสอบ Test Scenario จะมีความสัมพันธ์กับ Test Case ในรูปของความสัมพันธ์แบบหนึ่งต่อกลุ่ม นั่นคือหนึ่ง Test Scenario ประกอบไปด้วยหลาย ๆ Test Case นั่นเอง โดยปกติแล้วรายละเอียดของ Test Case จะถูกกำหนดขึ้นจาก Test Scenario ในขณะที่ Test Scenario จะถูกกำหนดขึ้นจากยูสเคสหรือความต้องการของระบบรูปแบบอื่น ๆ

ในทางปฏิบัติแล้วการทดสอบฟังก์ชันการทำงานของระบบใดก็ตามมักเริ่มต้นจากการสร้าง Test Scenario ก่อนจากนั้นจึงถูกแปลงให้อยู่ในรูปของข้อมูลที่ใช้ในการทดสอบด้วยมือ ซึ่งประกอบไปด้วยลำดับขั้นตอนของการทดสอบที่ถูกสร้างขึ้นในรูปของภาษาเขียนที่ใช้ทั่ว ๆ ไป ซึ่งแต่ละขั้นตอนของการทดสอบส่วนใหญ่จะเป็นชุดคำสั่งเฉพาะที่ใช้ในการทดสอบการทำงานของระบบที่อยู่ภายใต้การทดสอบเป็นหลัก แต่อย่างไรก็ตามการทดสอบด้วยมือที่ปกติจะถูกดำเนินการโดยนักทดสอบที่นั่งอยู่หน้าจอคอมพิวเตอร์และประมวลผลการทดสอบตามขั้นตอนที่กำหนดไว้ ซึ่งการทำงานในลักษณะนี้อาจก่อให้เกิดความเบื่อหน่ายในการทำงาน โดยเฉพาะอย่างยิ่งการทดสอบที่จำเป็นต้องทำซ้ำ ๆ ติดต่อกันเป็นเวลานาน นอกจากนั้นการทดสอบด้วยวิธีการนี้ยังมีค่าใช้จ่ายในการลงทุนด้านทรัพยากรมนุษย์ค่อนข้างสูง เนื่องจากปริมาณงานที่มากจำเป็นต้องใช้นักทดสอบมากขึ้นตามไปด้วย และข้อจำกัดในเรื่องของเวลาที่บ่อยครั้งส่งผลให้การทดสอบด้วยมือในทุก ๆ ฟังก์ชันการทำงานของระบบก่อนการส่งมอบเป็นสิ่งที่เป็นไปไม่ได้ และบ่อยครั้งที่ผู้ใช้ระบบอาจพบว่าข้อบกพร่องที่ยังค้นหาไม่พบจากการทดสอบในลักษณะดังกล่าวยังคงหลงเหลืออยู่ภายในระบบเสมอ ในเวลาต่อมาจึงได้มีการคิดค้นกระบวนการทดสอบอัตโนมัติขึ้นเพื่อช่วยในการแก้ไขปัญหาดังกล่าว โดยการแปลงขั้นตอนการทดสอบด้วยมือให้เป็นขั้นตอนการทดสอบอัตโนมัติโดย

อยู่ในรูปของสคริปต์ที่สามารถนำไปประมวลผลบนเครื่องมือทดสอบแบบอัตโนมัติได้ ดังนั้นอาจสรุปได้ว่าการทดสอบอัตโนมัติเป็นการทำงานโดยใช้ซอฟต์แวร์ควบคุมการประมวลผลการทดสอบและนำผลลัพธ์ที่ได้มาเปรียบเทียบกับผลลัพธ์ที่คาดไว้โดยอัตโนมัติ

ปัจจุบันการพัฒนาซอฟต์แวร์ส่วนใหญ่ถูกสร้างขึ้นในรูปของเว็บแอพพลิเคชั่นที่ทำงานบนบราวเซอร์เป็นหลัก ผลลัพธ์ที่ได้จากการทดสอบเว็บแอพพลิเคชั่นเหล่านี้จะถูกนำไปใช้ในการแก้ไขและปรับปรุงเพื่อให้เว็บแอพพลิเคชั่นมีการทำงานที่ถูกต้องสมบูรณ์ ส่งผลให้เกิดความน่าเชื่อถือตลอดจนมีคุณภาพมากขึ้นตามไปด้วย ด้วยเหตุนี้การทดสอบอัตโนมัติจึงกลายเป็นส่วนประกอบที่สำคัญในวงจรชีวิตของการพัฒนาซอฟต์แวร์ นอกจากนั้นการทดสอบอัตโนมัติยังสามารถรันการทดสอบซ้ำได้ตามต้องการ ซึ่งจะเป็นประโยชน์อย่างยิ่งในกรณีที่มีการทดสอบที่เรียกว่าการทดสอบแบบถดถอย (Regression testing) สำหรับข้อดีของการทดสอบอัตโนมัติสามารถสรุปได้ดังต่อไปนี้

- ช่วยลดเวลาและค่าใช้จ่ายในการทดสอบ ในกรณีที่มีการแก้ไขซอร์สโค้ดจำเป็นต้องทดสอบซ้ำเสมอ การทดสอบซ้ำด้วยมือจะสิ้นเปลืองเวลาและค่าใช้จ่ายสูง ในขณะที่การทดสอบอัตโนมัติหลังจากสร้างการทดสอบแล้วสามารถนำไปรันซ้ำ ๆ ได้โดยไม่ต้องมีค่าใช้จ่ายเพิ่มแต่อย่างใด
- ช่วยเพิ่มความถูกต้องในการทำงาน การทดสอบด้วยมือเป็นการทำงานซ้ำ ๆ ซึ่งอาจก่อให้เกิดความล้าแก่นักทดสอบส่งผลให้ความถูกต้องของการทดสอบลดลง ส่วนการทดสอบอัตโนมัติสามารถแก้ปัญหานี้ได้โดยการทดสอบสามารถทำงานได้อย่างถูกต้องและได้ผลลัพธ์เท่ากันทุกครั้งที่มีการรันการทดสอบ
- ช่วยเพิ่มความครอบคลุมการทดสอบ การทดสอบอัตโนมัติสามารถเพิ่มระดับความลึกและขยายขอบเขตของการทดสอบได้มากขึ้น โดยมีความสามารถในการประมวลผล Test Case ที่มีความซับซ้อนและมีจำนวนมากได้โดยง่าย ส่งผลให้ความครอบคลุมในการทดสอบสูงกว่าการทดสอบด้วยมืออย่างเห็นได้ชัด
- สามารถทำงานที่การทดสอบด้วยมือทำไม่ได้ เช่น ในกรณีที่ต้องการทดสอบเว็บแอพพลิเคชั่นที่รองรับผู้ใช้เป็นจำนวนมาก และผู้ใช้มีการเข้าถึงแอพพลิเคชั่นในเวลาใกล้เคียงกัน เป็นต้น

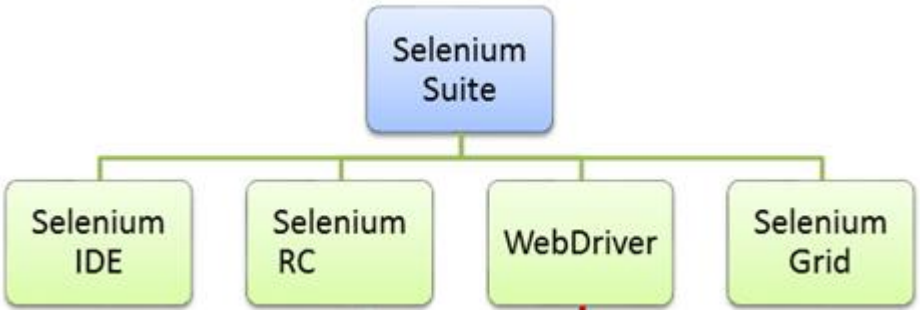
แม้ว่าการทดสอบอัตโนมัติจะมีข้อดีหลายประการ แต่ยังคงมีข้อเสียตรงที่มีค่าใช้จ่ายในด้านการลงทุนสูง โดยเฉพาะอย่างยิ่งการจัดซื้อเครื่องมือทดสอบอัตโนมัติ และเนื่องจากเครื่องมือทดสอบอัตโนมัติแต่ละชนิดถูกพัฒนาขึ้นจากบริษัทผู้ผลิตที่แตกต่างกัน จึงส่งผลให้การทำงานของ

เครื่องมือแต่ละชนิดแตกต่างกันตามไปด้วย ดังนั้นปัญหาที่เกิดขึ้นตามมาได้แก่ บุคลากรที่เป็นนักทดสอบมักมีความเชี่ยวชาญเฉพาะการใช้เครื่องมือทดสอบอัตโนมัติชนิดหนึ่ง ๆ ที่มีความคุ้นเคยเท่านั้น ซึ่งในกรณีนี้จะเกิดผลเสียตรงที่ไม่สามารถนำประสบการณ์จากการใช้เครื่องมือหนึ่ง ๆ ไปใช้กับเครื่องมือทดสอบอัตโนมัติชนิดที่แตกต่างออกไปได้นั่นเอง ดังนั้นในการเลือกใช้เครื่องมือทดสอบอัตโนมัติจำเป็นต้องคำนึงถึงความเชี่ยวชาญของนักทดสอบเฉพาะทางเสมอ ปัจจุบันตลาดของเครื่องมือทดสอบอัตโนมัติมีทั้งสองประเภท โดยประเภทแรกเป็นซอฟต์แวร์เชิงพาณิชย์ที่จำหน่ายตามจำนวนไลเซนส์ (License) ที่ใช้ เช่น QuickTest Professional หรือ Rational Functional Tester ที่มีการพัฒนามาอย่างต่อเนื่องและยาวนาน ซึ่งซอฟต์แวร์ประเภทนี้จะมีราคาค่อนข้างสูง ส่วนประเภทที่สองเป็นซอฟต์แวร์แบบรหัสเปิด (Open Source) ที่สามารถใช้งานได้ฟรีโดยไม่มีค่าใช้จ่ายแต่อย่างใด เครื่องมือทดสอบประเภทหลังนี้จะได้รับความนิยมในการใช้งานเพิ่มขึ้น ๆ ทุกปี ซึ่งหนึ่งในจำนวนนั้นได้แก่ Selenium ซึ่งนักทดสอบที่มีประสบการณ์การใช้เครื่องมือชนิดนี้จะเป็นที่ต้องการของตลาดเป็นอย่างมากในปัจจุบัน

1.2 ความเป็นมาของ Selenium

Selenium เป็นกลุ่มของเครื่องมือทดสอบต่าง ๆ ซึ่งแต่ละชนิดมีวิธีการทำงานร่วมกับการทดสอบอัตโนมัติที่แตกต่างกัน โดยนักทดสอบส่วนใหญ่จะเน้นไปที่การใช้เครื่องมือเพียงหนึ่งหรือสองชนิดเท่านั้นเพื่อให้เป็นไปตามความต้องการในการทดสอบ แต่อย่างไรก็ตามการใช้เครื่องมือทั้งหมดร่วมกับการทดสอบอัตโนมัติจะช่วยให้การทดสอบสามารถครอบคลุมการทำงานของเว็บแอพพลิเคชั่นได้ในทุก ๆ มิติ ข้อดีประการหนึ่งของการใช้ Selenium ในการทดสอบอัตโนมัติคือความสามารถในการรองรับการทำงานร่วมกับบราวเซอร์และระบบปฏิบัติการที่แตกต่างกันได้ ส่งผลให้นักทดสอบสามารถทดสอบระบบที่มีความหลากหลายได้มากขึ้น ดังที่ได้กล่าวมาแล้วว่า Selenium ถูกออกแบบมาเพื่อวัตถุประสงค์ในการใช้งานที่แตกต่างกัน และสามารถแบ่งออกได้เป็น 4 ชนิดดังต่อไปนี้

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid



รูปที่ 1.1 กลุ่มของเครื่องมือทดสอบที่ใช้ใน Selenium

1.3 พัฒนาการของ Selenium

Selenium ถูกพัฒนาขึ้นเป็นครั้งแรกในปี 2004 และมีพัฒนาการต่อเนื่องมาจนถึงปัจจุบัน ซึ่งในแต่ละช่วงเวลาของการพัฒนาได้มีการเพิ่มส่วนประกอบต่าง ๆ เข้าไปเพื่อให้มีความสามารถในการทำงานมากยิ่งขึ้น ในขณะเดียวกันได้มีการตัดบางส่วนที่ล้าสมัยหรือไม่จำเป็นออกไปเพื่อให้เครื่องมือมีความทันสมัยมากขึ้น ซึ่งพัฒนาการดังกล่าวสามารถแบ่งออกได้ตามช่วงเวลาที่สำคัญ ๆ ดังต่อไปนี้

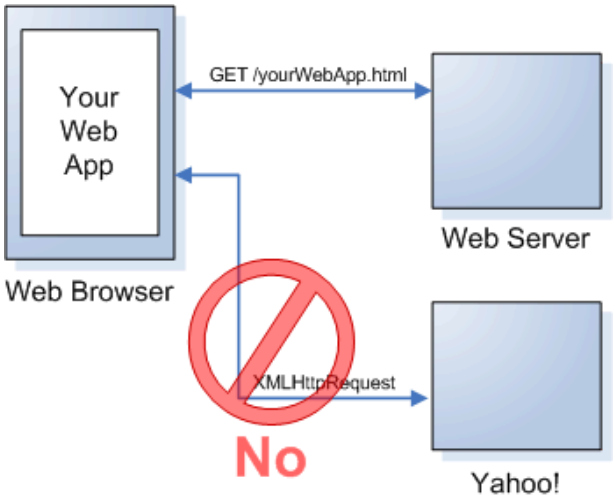
1.3.1 Selenium Core

ในปี 2004 Jason Huggins วิศวกรของบริษัท ThoughtWorks ซึ่งทำหน้าที่ในการพัฒนาเว็บแอพพลิเคชั่นที่ต้องการทดสอบบ่อย ๆ ซึ่งจากการทำงานดังกล่าวก่อให้เกิดแนวคิดที่ว่าการทดสอบด้วยมือซ้ำ ๆ ส่งผลให้ประสิทธิภาพในการทดสอบลดลง ๆ ดังนั้นจึงได้พัฒนาโปรแกรมด้วยจาวาสคริปต์โดยใช้ชื่อว่า JavaScriptTestRunner เพื่อใช้สำหรับควบคุมการกระทำของบราวเซอร์โดยอัตโนมัติ ซึ่งในเวลาต่อมาได้ถูกเปลี่ยนชื่อเป็น Selenium Core และถือเป็นพื้นฐานการทำงานที่สำคัญสำหรับ Selenium Remote Control (RC) และ Selenium IDE ที่ถูกพัฒนาขึ้นในเวลาต่อมา

1.3.1.1 Same Origin Policy

เป็นชื่อของแนวคิดด้านความปลอดภัยที่ใช้กับจาวาสคริปต์ในการโปรแกรมบนฝั่งไคลเอนต์ โดยป้องกันไม่ให้จาวาสคริปต์สามารถเข้าถึงส่วนประกอบใด ๆ ของโดเมนที่แตกต่างไปจากโดเมนที่ถูกเรียกใช้ครั้งแรก ตัวอย่างเช่น โค้ด HTML ภายใน www.google.com ถูกใช้งานร่วมกับจาวาสคริปต์ ซึ่งกลไกที่ใช้ Same Origin Policy จะยอมให้จาวาสคริปต์ดังกล่าวสามารถเข้าถึงเว็บเพจได้เฉพาะภายในโดเมน google.com เท่านั้น ซึ่งในกรณีนี้ได้แก่ google.com/mail,

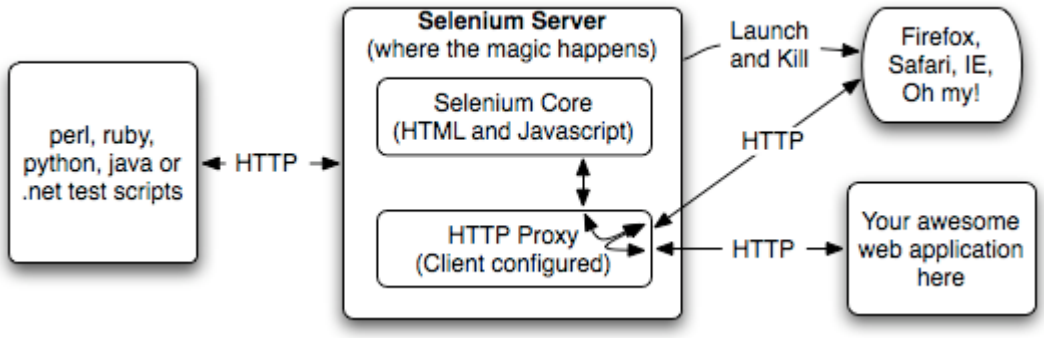
google.com/login, หรือ google.com/signup เป็นต้น แต่ในทางตรงกันข้ามกลไกดังกล่าวจะไม่ยอมให้จาวาสคริปต์ดังกล่าวสามารถเข้าถึงเว็บเพจจากโดเมนที่แตกต่างกันไปจาก www.google.com ได้ เช่น www.yahoo.com/search หรืออื่น ๆ เป็นต้น ทั้งนี้เนื่องจากเป็นโดเมนที่แตกต่างไปจากโดเมนที่ถูกเรียกใช้ครั้งแรกนั่นเอง ซึ่งแนวคิดในการทำงานดังกล่าวสามารถนำเสนอได้ดังรูปต่อไปนี้



รูปที่ 1.2 แนวคิดในการทำงานที่เรียกว่า Same Origin Policy

1.3.2 Selenium RC

ย่อมาจากคำว่า Selenium Remote Control ซึ่งถูกพัฒนาขึ้นเพื่อแก้ไขข้อจำกัดจากแนวคิดแบบ Same Origin Policy โดย Paul Hammant วิศวกรอีกคนหนึ่งจากบริษัท ThoughtWork ได้พัฒนาโปรแกรมที่ใช้เป็นเซิร์ฟเวอร์เพื่อทำหน้าที่เป็น HTTP proxy เพื่อใช้สำหรับหลอกบราวเซอร์ให้เชื่อว่า Selenium Core และเว็บแอพพลิเคชั่นได้ถูกทดสอบจากโดเมนเดียวกัน ซึ่งการทำงานในลักษณะดังกล่าวจึงเป็นที่มาของ Selenium RC หรือ Selenium 1 ในเวลาต่อมา นอกจากนั้นยังส่งผลให้นักทดสอบที่ใช้ Selenium Core จำเป็นต้องติดตั้งทั้งเว็บแอพพลิเคชั่นที่ต้องการทดสอบและ Selenium RC ที่เป็นเซิร์ฟเวอร์ลงในเครื่องเดียวกันตามไปด้วย



รูปที่ 1.3 แสดงการทำงานของ Selenium RC

Selenium RC นอกจากจะใช้เป็นเครื่องมือทดสอบอัตโนมัติสำหรับเว็บแอพพลิเคชั่นในช่วงต้น ๆ แล้ว ยังสนับสนุนการทำงานร่วมกับโปรแกรมภาษาต่าง ๆ ได้แก่ Java, C#, PHP, Python, Perl, Ruby เป็นต้น นอกจากนั้น Selenium RC อาศัยการทำงานแบบไคลเอนต์เซิร์ฟเวอร์ที่ยอมให้ผู้ใช้ควบคุมการทำงานของบราวเซอร์จากภายในเครื่องเดียวกันหรือจากเครื่องคอมพิวเตอร์อื่น ๆ ได้ สำหรับข้อดีและข้อเสียของการใช้ Selenium RC สามารถสรุปได้ดังตารางต่อไปนี้

ตารางที่ 1.1 สรุปข้อดีและข้อเสียของการใช้งานร่วมกับ Selenium RC

ข้อดี	ข้อเสีย
สามารถทำงานร่วมกับบราวเซอร์และระบบปฏิบัติการที่แตกต่างกันได้	การติดตั้งมีความยุ่งยากกว่า Selenium IDE
สนับสนุนการทำงานแบบวนซ้ำ (Loop) และเงื่อนไขต่าง ๆ (if/switch)	นักทดสอบต้องมีประสบการณ์ทางด้านการโปรแกรมมาก่อน
สนับสนุนการทำงานแบบ Data Driven Testing	ต้องการ Selenium RC เซิร์ฟเวอร์สำหรับรันการทำงาน
มี API ที่สมบูรณ์และพร้อมใช้งาน	มี API มากเกินจำเป็นและบางคำสั่งซ้ำซ้อนกันก่อให้เกิดความสับสน
สนับสนุนการทำงานของบราวเซอร์รุ่นใหม่ ๆ	ไม่สามารถลำดับการรันการทดสอบในกรณีที่มีหลาย ๆ Test Case

1.3.3 Selenium Grid

ถูกพัฒนาขึ้นโดย Patrick Lightbody เพื่อจุดประสงค์ในการลดเวลาที่ใช้ในการประมวลผลการทดสอบ โดยใช้แนวคิดของการประมวลผลการทดสอบแบบขนาน นั่นคือการรันหลาย ๆ การทดสอบที่แตกต่างกันในเวลาเดียวกันสามารถเกิดขึ้นได้ในบราวเซอร์และระบบปฏิบัติการที่แตกต่างกัน โดยการผ่านคำสั่ง Selenium ไปยังเครื่องคอมพิวเตอร์อื่น ๆ เพื่อการประมวลผลการทดสอบในเวลาเดียวกันได้ นั่นคือผู้ใช้สามารถใช้ Selenium Grid เป็นเครื่องมือที่ใช้ในการทดสอบร่วมกับ Selenium RC เพื่อรันการทำงานแบบขนานได้ ปัจจุบัน Selenium Grid สามารถแบ่งออกได้เป็น 2 เวอร์ชัน ได้แก่ Grid 1 ซึ่งเป็นเวอร์ชันเก่าและ Grid 2 เป็นเวอร์ชันที่ใช้งานอยู่ในปัจจุบัน Selenium Grid ใช้แนวคิดในการทำงานแบบ Hub-Node โดยผู้ใช้สามารถรันการทดสอบเฉพาะบนเครื่องคอมพิวเตอร์ซึ่งทำหน้าที่เป็น Hub และผ่านค่าคำสั่งไปยังคอมพิวเตอร์ซึ่งทำหน้าที่เป็น Nodes เพื่อประมวลผลการทดสอบในเวลาเดียวกันดังรูปต่อไปนี้

รูปที่ 1.4 แสดงการทำงานของ Selenium Grid

การเลือกใช้งานของ Selenium Grid จะพิจารณาจากการทำงานสองลักษณะดังต่อไปนี้

- รันการทดสอบร่วมกับบราวเซอร์ ระบบปฏิบัติการ และเครื่องที่แตกต่างกันในช่วงเวลาเดียวกัน ทั้งนี้เพื่อให้แน่ใจว่าเว็บแอพพลิเคชั่นที่ถูกทดสอบสามารถทำงานร่วมกับบราวเซอร์และระบบปฏิบัติการที่แตกต่างกันได้
- ต้องการลดเวลาในการประมวลผลการทดสอบ เช่นในกรณีที่กำหนดให้ Selenium Grid รัน 4 การทดสอบในเวลาเดียวกัน ผู้ใช้จะสามารถทดสอบได้เร็วขึ้น 4 เท่านั่นเอง

ตารางที่ 1.2 แสดงความแตกต่างหลัก ๆ ระหว่าง Selenium Grid 1 และ 2

Grid 1	Grid 2
มี remote control ของตัวเองที่แตกต่างไปจาก Selenium RC เซิร์ฟเวอร์	ได้มีการรวบรวม Selenium Server jar ไฟล์ไว้ด้วยกัน
ผู้ใช้จำเป็นต้องติดตั้งและคอนฟิก Apache Ant ก่อนการเรียกใช้ Grid 1	ไม่จำเป็นต้องติดตั้ง Apache Ant ใน Grid 2
สนับสนุนการทำงานเฉพาะคำสั่งหรือสคริปต์ Selenium RC	สนับสนุนการทำงานทั้ง Selenium RC และ WebDriver สคริปต์
ผู้ใช้สามารถรันอัตโนมัติได้เฉพาะหนึ่งบราวเซอร์ต่อหนึ่ง remote control	หนึ่ง remote control สามารถรันอัตโนมัติได้ 5 บราวเซอร์

1.3.4 Selenium IDE

ถูกพัฒนาขึ้นโดยชาวญี่ปุ่นที่ชื่อ Shinya Kasatani ในรูปส่วนต่อขยายของ Firefox ที่สามารถใช้ในการทำงานแบบอัตโนมัติร่วมกับบราวเซอร์ผ่านเทคนิคการทดสอบที่เรียกว่า Record & Playback โดยมีจุดประสงค์เพื่อให้เป็นเครื่องมือทดสอบที่ใช้งานได้ง่าย โดยผู้ใช้ไม่จำเป็นต้องมีพื้นความรู้ทางด้านการโปรแกรมมาก่อน นอกจากนั้นยังเป็นเครื่องมือที่ช่วยเพิ่มความเร็วในการสร้าง Test Case ซึ่งในเวลาต่อมาจึงได้มอบ Selenium IDE ให้กับโครงการพัฒนา Selenium ในปี 2006

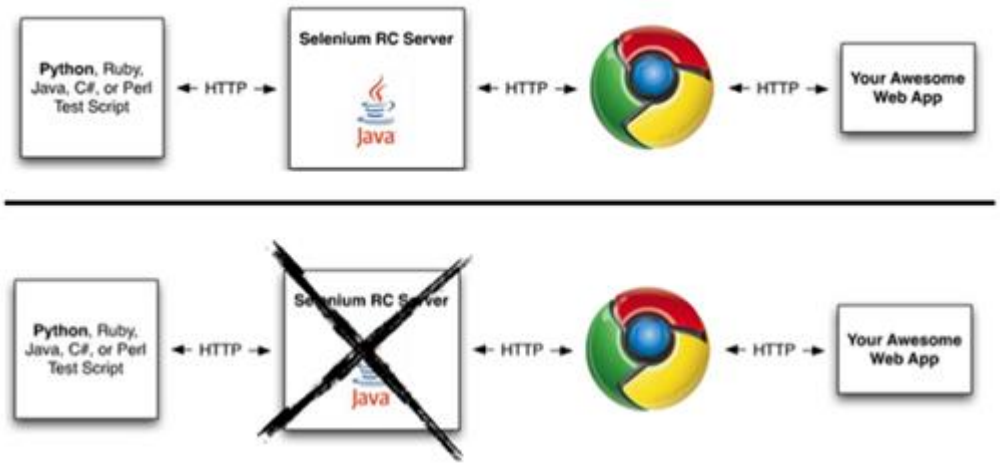
Selenium IDE จัดเก็บสคริปต์สำหรับการทดสอบในรูปของตารางที่สามารถนำกลับมาใช้ใหม่เพื่อการประมวลผลซ้ำได้ในภายหลัง แต่อย่างไรก็ตามตัวโปรแกรมเองยังคงขาดคุณสมบัติที่สำคัญอื่น ๆ ได้แก่ คุณสมบัติในการรันผลลัพธ์การทดสอบหรือความสามารถในการสร้างการทดสอบอัตโนมัติทั้งหมดที่ผู้ใช้ต้องการ นอกจากนั้น Selenium IDE ยังไม่สนับสนุนการทำงานประเภททำซ้ำหรือเงื่อนไขต่าง ๆ สำหรับการใช้งานร่วมกับสคริปต์สำหรับการทดสอบ ทั้งนี้เนื่องมาจากเหตุผลบางส่วนที่เป็นปัญหาทางเทคนิคและบางส่วนจากตัวนักพัฒนาเอง อย่างไรก็ตามเนื่องจากความง่ายต่อการใช้งาน Selenium IDE จึงถูกจำกัดให้เป็นได้เฉพาะเครื่องมือต้นแบบที่สามารถใช้ในการทดสอบแบบพื้นฐานเท่านั้น ส่วนในกรณีที่ต้องการสร้าง Test Case ที่มีการทำงานแบบซับซ้อนจำเป็นต้องเลือกใช้ Selenium RC หรือ WebDriver แทน

ตารางที่ 1.3 สรุปข้อดีและข้อเสียของการใช้งานร่วมกับ Selenium IDE

ข้อดี	ข้อเสีย
สะดวกต่อการติดตั้งและใช้งาน	ใช้ได้เฉพาะกับบราวเซอร์ Firefox
ไม่จำเป็นต้องมีประสบการณ์ด้านโปรแกรมมาก่อน ใช้ความรู้เฉพาะ HTML และ DOM	ออกแบบมาเพื่อใช้เป็นเครื่องมือต้นแบบเท่านั้น
สามารถส่งออกการทดสอบไปยังรูปแบบที่ใช้ได้กับ Selenium RC และ WebDriver	ไม่สนับสนุนการทำงานแบบทำซ้ำหรือแบบมีเงื่อนไข
มี help และโมดูลที่ถูกสร้างไว้แล้วสำหรับรายงานผลการทดสอบ	การประมวลผลการทดสอบช้าเมื่อเปรียบเทียบกับ selenium RC และ WebDriver

1.3.5 WebDriver

ในปี 2006 Simon Stewart ได้พัฒนา WebDriver ขึ้นโดยเพิ่ม API ที่เป็นทางเลือกสำหรับฟังก์ชันการทำงานที่ไม่ได้รับการสนับสนุนจาก Selenium-RC โดยการทำงานของ WebDriver จะไม่ขึ้นอยู่กับจาวาสคริปต์ที่ฝังตัวอยู่ในบราวเซอร์ ดังนั้นจึงสามารถลดข้อจำกัดที่เกิดขึ้นจากการทำงานของ Selenium ได้ นอกจากนั้น WebDriver ยังมีเป้าหมายหลักในการพัฒนาเครื่องมือทดสอบอัตโนมัติเพื่อให้สามารถทำงานข้ามแพลตฟอร์มได้ รวมถึงความสามารถในการควบคุมการทำงานร่วมกับบราวเซอร์ได้ในระดับระบบปฏิบัติการ สำหรับความแตกต่างระหว่าง Selenium RC และ WebDriver สามารถนำเสนอได้จากรูปดังต่อไปนี้



รูปที่ 1.5 เปรียบเทียบการทำงานระหว่าง Selenium RC และ WebDriver

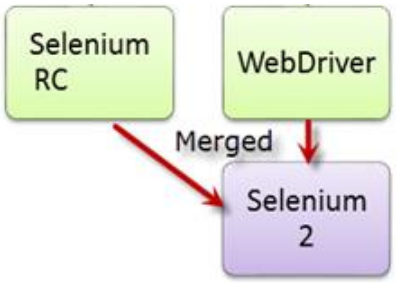
WebDriver ได้ถูกออกแบบมาเพื่อแก้ไขข้อจำกัดต่าง ๆ ที่เพิ่มมากขึ้นจากการใช้จาวาสคริปต์ รวมไปถึงข้อบกพร่องต่าง ๆ ที่เกิดขึ้นทั้งจาก Selenium IDE และ Selenium RC โดยมีชุดคำสั่งที่ถูกออกแบบมาเป็นแบบเชิงวัตถุส่งผลให้การนำไปใช้สามารถทำได้ง่ายขึ้น รวมทั้งยังสนับสนุนการทำงานกับโปรแกรมภาษาต่าง ๆ เช่นเดียวกับ Selenium RC นอกจากนั้นยังมีการทำงานที่เสถียรกว่าในการทดสอบอัตโนมัติร่วมกับบราวเซอร์ชนิดต่าง ๆ

ตารางที่ 1.4 สรุปข้อดีและข้อเสียของการใช้งานร่วมกับ WebDriver

ข้อดี	ข้อเสีย
การติดตั้งง่ายกว่า Selenium RC	การติดตั้งยุ่งยากกว่า Selenium IDE
ติดต่อโดยตรงกับบราวเซอร์	ต้องการความรู้ด้านโปรแกรมมาก่อน
การติดต่อกับบราวเซอร์มีความเสมือนจริง	ไม่สนับสนับสนุนบราวเซอร์รุ่นใหม่ ๆ
ไม่ต้องใช้คอมโพเนนท์อื่น เช่นเดียวกับใน Selenium RC Server	ไม่มีกลไกสำหรับแสดงข้อความเตือนในช่วงเวลารันไทม์และสร้างผลลัพธ์การทดสอบ
ประมวลผลได้เร็วกว่า Selenium IDE และ Selenium RC	

1.3.6 Selenium 2

ในปี 2008 ทีมพัฒนา Selenium ทั้งหมดได้ตัดสินใจรวม WebDriver และ Selenium RC เข้าด้วยกันเพื่อสร้างเครื่องมือชนิดใหม่ที่มีประสิทธิภาพมากขึ้นเรียกว่า Selenium 2 โดยใช้ WebDriver เป็นแกนหลักในการทำงาน ปัจจุบัน Selenium RC ไม่มีการพัฒนาเวอร์ชันใหม่อีกต่อไปแล้ว มีแต่เพียงส่วนของการบำรุงรักษาเท่านั้น ดังนั้นแนวโน้มของการพัฒนาเครื่องมือทดสอบปัจจุบันจะเน้นไปที่ Selenium 2 เสมอ ซึ่งรายละเอียดการทำงานดังกล่าวจะถูกนำเสนอในบทที่ 6



รูปที่ 1.6 Selenium 2 เกิดจากการรวมตัวกันของ Selenium RC และ WebDriver