**บทที่ 2**

**ทฤษฎีที่เกี่ยวข้อง**

ในบทนี้จะกล่าวถึงความรู้ และทฤษฎีที่เกี่ยวข้องกับการพัฒนาโปรแกรม โดยผู้จัดทำได้ศึกษารวบรวมข้อมูลและนำความรู้เหล่านี้ไปประยุกต์ใช้ในการพัฒนาโปรแกรม ซึ่งทฤษฎีที่เกี่ยวข้องที่ได้รวบรวมไว้ มีดังนี้

1. การทดสอบแบบถดถอย (Regression Testing)

การทดสอบแบบถดถอย หรือ Regression Testing [] คือ การทดสอบซอฟต์แวร์ประเภทหนึ่ง ที่ใช้ในการทดสอบเพื่อค้นหาข้อผิดพลาดใหม่หรือ ข้อผิดพลาดเดิมที่เคยเกิดขึ้นแล้วและซ่อมไปแล้วในระบบที่อาจะเกิดขึ้นหลังจากที่ระบบมีการเปลี่ยนแปลง มีการอัพเดทเพิ่มประสิทธิภาพ หรือมีการแก้ข้อผิดพลาดใหม่ของระบบ

โดยจุดประสงค์ของ regression testing นั้นก็เพื่อที่จะทำการทดสอบเพื่อยืนยันให้แน่ใจว่าการเปลี่ยนแปลงที่เกิดขึ้นกับระบบตามที่กล่าวไว้ข้างต้นนั้น จะต้องไม่ทำให้เกิดข้อผิดพลาดใหม่ในระบบขึ้นมา โดยหนึ่งในเหตุผลสำคัญสำหรับการทำ regression คือการตรวจสอบว่าการเปลี่ยนแปลงในส่วนใดส่วนหนึ่งของซอฟต์แวร์ไปกระทบกับส่วนอื่นหรือไม่

สำหรับวิธีที่ใช้ทั่วไปสำหรับการทำ regression testing ก็คือการกลับไปทำการทดสอบทุกฟังก์ชั่น หรือพฤติกรรมในระบบเพื่อดูว่ามีข้อผิดพลาดเกิดขึ้น หรือข้อผิดพลาดที่เคยแก้ไขไปแล้ว ปรากฏขึ้นอีกครั้งหรือไม่ ซึ่งอาจเป็นการเสียเวลามากหากไม่มีการพัฒนาระบบให้มีความเป็นอัตโนมัติ ดังนั้นวิธีที่นิยมในการทำ regression testing อย่างมีประสิทธิภาพก็คือ การทำ impact analysis หรือการวิเคราะห์ถึงผลกระทบที่เกิดขึ้นได้ในการเปลี่ยนแปลงอะไรบางอย่างในระบบ เพื่อลดขอบเขตการทดสอบลง ให้เหลือน้อยที่สุด ไม่ใช่การทดสอบทุกฟังก์ชั่นที่มีในระบบ

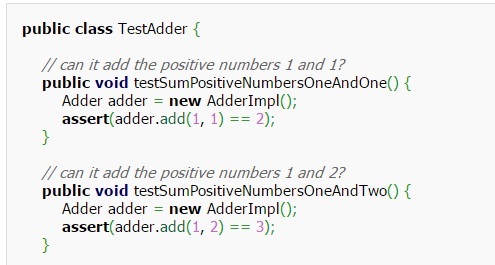
1. การทดสอบอัตโนมัติ (Automated Testing)

Automated Testing [] คือ การทดสอบซอฟต์แวร์ด้วยคอมพิวเตอร์แบบอัตโนมัติแทนการทดสอบโดยมนุษย์ โดยจะใช้เครื่องมือซึ่งช่วยในการรันชุดคำสั่งที่ใช้ทดสอบซอฟต์แวร์ที่ต้องการ และช่วยเปรียบเทียบผลลัพธ์จริงที่ได้จากระบบกับผลลัพธ์ที่ควรจะได้รับ เพื่อตรวจสอบว่าซอฟต์แวร์ที่ทดสอบนี้ทำงานได้ถูกต้องหรือไม่ โดยปกติในการทดสอบด้วยมือนั้น ผู้ทดสอบ 1 คนสามารถทดสอบได้ทีละ 1 เทสเท่านั้น ทำให้ต้องใช้เวลานานหรือใช้คนทดสอบจำนวนมาก ซึ่งจุดประสงค์หลักในการทำ Automated Testing ก็เพื่อลดจำนวนการทดสอบที่ต้องใช้คนลง เพื่อประหยัดเวลาและต้นทุนที่ใช้ในการทดสอบในแต่ละครั้ง โดยใช้เครื่องมือในการรันเทสอย่างอัตโนมัติ ไม่จำเป็นต้องมีคนคอยควบคุมขณะรัน และยังสามารถรันเทสหลายเทสได้พร้อมกันในเวลาเดียว ผู้ทดสอบเพียงแค่สั่งรันเทสแล้วรอผลลัพธ์ของการทดสอบ ซึ่งอยู่ในรูปของรายงานสรุปผลการทดสอบ

* 1. ลักษณะของเทสเคสที่เหมาะสมในการทำ Automated Testing [2]

1. เทสเคสที่มีการทดสอบซ้ำๆ และไม่ค่อยมีการเปลี่ยนแปลง
2. เทสเคสที่ทดสอบด้วยมือได้ยากหรือน่าเบื่อหน่าย
3. เทสเคสที่ต้องทดสอบเป็นเวลานาน
4. เทสเคสที่มีความสำคัญมากและมีความเสี่ยงสูง ถ้าทดสอบด้วยมืออาจจะผิดพลาดได้
5. เทสเคสที่มีการทดสอบฟังก์ชันเดิมๆ แต่ข้อมูลที่ใช้ทดสอบมีหลายชุด เช่น ฟังก์ชันล็อกอิน ฟังก์ชันค้นหา
   1. ข้อดีของ Automated Testing [] มีดังนี้
6. ประหยัดเวลาและต้นทุนในการทดสอบซอฟต์แวร์ ช่วยให้รันเทสหลายเทสพร้อมกันได้อย่างอัตโนมัติ สามารถรันทิ้งไว้แล้วรอผลลัพธ์ได้เลย
7. เพิ่มความแม่นยำการทดสอบซอฟต์แวร์ การทดสอบแบบอัตโนมัติช่วยให้มั่นใจได้ว่าการทดสอบระบบเป็นไปอย่างถูกต้อง ไม่มีปัญหาที่เกิดจากความผิดพลาดของผู้ทดสอบ
8. การทดสอบมีความครอบคลุมมากขึ้น เนื่องจากในการทดสอบด้วยมือนั้น ผู้ทดสอบอาจจะไม่ได้ทดสอบครบทุกกรณี เช่น การทดสอบซอฟต์แวร์ที่ต้องทดสอบทุก platform การนำ Automated Testing มาใช้จะช่วยให้การทดสอบเป็นไปอย่างครอบคลุม ครบถ้วนทุกกรณี
9. ช่วยให้ทดสอบในกรณีที่การทดสอบด้วยมือไม่สามารถทำได้หรือทำได้ยาก เช่น การทดสอบระบบในกรณีที่มีผู้ใช้จำนวนมากเข้าใช้ระบบในเวลาเดียว ซึ่งการทำ Automated Testing สามารถทำได้ โดยรันเทสบนคอมพิวเตอร์หลายๆ เครื่อง พร้อมๆ กัน
10. ลดเวลาในการค้นหาและแก้ไขข้อผิดพลาดของซอฟต์แวร์ โปรแกรมเมอร์สามารถรันเทสด้วยตัวเองได้ทันทีหลังจากที่เขียนโปรแกรมเสร็จ ทำให้สามารถรู้ถึงข้อผิดพลาดที่เกิดขึ้นได้อย่างรวดเร็ว และยังรู้ว่าข้อผิดพลาดที่เกิดนั้นเกิดขึ้นจากส่วนไหน
11. ผู้ทดสอบระบบได้ทำงานที่มีความท้าทายและเป็นประโยชน์แก่องค์กรมากขึ้น งานส่วนที่ต้องทดสอบซ้ำๆ หรือสามารถให้คอมพิวเตอร์ทดสอบแทนได้ ก็ใช้คอมพิวเตอร์ทดสอบแทน ทำให้ผู้ทดสอบได้ทำงานในส่วนที่มีความซับซ้อนหรือจำเป็นต้องทดสอบด้วยมือ สร้างประโยชน์ให้องค์กรมากขึ้น
12. Unit Testing

Unit Testing [] คือ กระบวนการทดสอบซอฟต์แวร์รูปแบบหนึ่งซึ่งใช้ทดสอบซอร์สโค้ด, เมทอด หรือเซตของโมดูลในโปรแกรม ที่มีข้อมูลหรือกระบวนการทำงานที่เกี่ยวเนื่องกัน ว่าทำงานได้ถูกต้องตามหน้าที่ ที่ซอร์สโค้ดนั้นถูกเขียนขึ้นมาหรือไม่ หรือกล่าวง่ายๆก็คือ Unit Testing นั้นคือการทดสอบส่วนที่เล็กที่สุดเท่าที่จะทดสอบได้ในโปรแกรม ซึ่งอาจจะเป็นโมดูลการทำงานหนึ่ง หรือเพียงแค่เมทอดๆหนึ่งก็ได้ โดยในการโปรแกรมเชิงวัตถุหรือ Object-oriented programming นั้น Unit testing มักจะใช้ในการทดสอบเมทอดหรือ ทดสอบการทำงานของทั้งคลาส ซึ่งการทำ Unit testing นั้นจะมีการเขียน unit test ขึ้นมาโดยโปรแกรมเมอร์ซึ่งส่วนมากมักจะเป็น white box testers การเขียน Unit test นั้นมักจะถูกเขียนขึ้นระหว่างกระบวนการพัฒนาโปรแกรม โดย test ที่เขียนขึ้นมาเพื่อทดสอบส่วนๆหนึ่งจะเรียกว่า test case ซึ่งในแต่ละฟังก์ชั่น หรือ เมทอดหนึ่งจะถูกทดสอบด้วย test case เดียวหรือหลายๆ test case ก็เป็นไปได้ การเขียน test case นั้นจะมีการสร้าง test class ขึ้นมาซึงประกอบไปด้วย test case จำนวนหนึ่งเอาไว้ใน class นี้เพื่อเป็นการบ่งบอกและจัดกลุ่มให้ชัดเจนว่า class นี้เป็น class ที่เอาไว้ใช้ทดสอบฟังก์ชั่นใด ยกตัวอย่างเช่น ดังรูปที่ 2.x คือ test class ซึ่งในที่นี้ใช้ทดสอบการทำงานของ class ที่ชื่อ Adder ที่ใช้ในการบวกเลข



1. TestAdder Test class

จากรูปที่ 2.x class TestAdder นั้นประกอบไปด้วย 2 test case คือ

1. testSumPositiveNumbersOneAndOne เป็นการทดสอบเมทอด add ของ class Adder โดยการส่ง input 2 ตัวคือ 1 และ 1 โดยคาดหวังผลลัพธ์ว่าต้องได้ค่าออกเป็น 2
2. testSumPositiveNumbersOneAndTwo เป็นการทดสอบเมทอด add ของ class Adder โดยการส่ง input 2 ตัวคือ 1 และ 2 โดยคาดหวังผลลัพธ์ว่าต้องได้ค่าเป็น 3

จะเห็นได้ว่าทั้ง 2 test case นั้นเป็นการทดสอบเมทอด add ของ class Adder ทั้งคู่แต่ต่างกันที่ input ที่ใส่เข้าไป

* 1. คุณสมบัติของ Unit testing [2]

โดยคุณสมบัติของ Unit testing นั้นแบ่งออกได้เป็น 4 คุณสมบัติคือ

1. Isolate คือการแยก test case ออกจากกันอย่างชัดเจน มีเป้าหมายอย่างเดียวเท่านั้น โดยการแยกส่วนการทำงานต่างๆ ออกจากกันอย่างชัดเจน จำเป็นต้องใช้เทคนิคต่างๆทางโปรแกรมมิ่งเพื่อช่วยในการแบ่งการทดสอบออกอย่างชัดเจน เช่น Domain-Driven Design, Dependency Injection และ Single Responsibility เป็นต้น
2. Repeatable คือการกลับมาทดสอบอีกครั้งนั้นสามารถทำได้ง่าย ไม่จำเป็นต้องรู้การทำงานของซอร์สโค้ด สามารถรัน test case ได้เลยในทุก environment ยกตัวอย่างเช่น เมื่อมีการเพิ่มเติมฟังก์ชั่นการทำงานของเมทอดหนึ่ง แล้วต้องการทดสอบว่าเมทอดนั้นยังทำงานของฟังก์ชั่นเดิมได้ถูกต้องหรือไม่ developer ต้องสามารถกลับมารัน test case นี้ได้เลยทันที ไม่จำเป็นต้องติดต่อกับส่วนอื่นๆ
3. Fast คือ test case นั้นต้องสามารถรันได้อย่างรวดเร็ว ใช้เวลาน้อย เนื่องจากเวลานั้นมีค่าหากการ run test case ใช้เวลานานจะเสียค่าใช้จ่ายสูง ดังนั้นการเขียน test case นั้นจะต้องคำนึงถึงการใช้งานทรัพยากรที่เสียเวลามากซึ่งอาจแก้ไขโดยการใช้เทคนิค mocking หรือ stubbing เข้าช่วยในการลดเวลาในส่วนนี้
4. Self-documenting คือ ซอร์สโค้ดของ unit test ที่เขียนขึ้นนั้นจะต้องชัดเจน อ่านง่าย สามารถทำความเข้าใจได้ทันทีว่า test case นั้นๆทำการทดสอบอะไร และอย่างไร และทำให้เขียนเอกสารน้อยที่สุดเท่าที่จำเป็น ดังนั้นการเขียน test case นั้น ชื่อเมทอดจะต้องอ่านแล้วเข้าใจ และต้องมีการ document ตัวอย่างของข้อมูลเอาไว้เสมอ เพื่อช่วยทำความเข้าใจ
   1. ข้อดีของการทำ Unit test
5. ช่วยให้หาข้อผิดพลาดของระบบได้รวดเร็วโดยการทำ unit testing นั้นจะช่วยแก้ปัญหาทั้งข้อผิดพลาดที่เกิดจากการเขียน code ผิดพลาดของ developer เอง และยังช่วยในการตรวจสอบความครบถ้วนว่า ซอร์สโค้ดที่เขียนขึ้นสามารถทำงานได้ตาม requirement หรือไม่
6. ช่วยในการแก้ไขปรับปรุงซอร์สโค้ด เนื่องจากการทำ unit testing เราจะมี test class ใช้ในการกำกับการทำงานของแต่ละฟังก์ชั่นในระบบ ดังนั้นเมื่อมีการปรับแต่งแก้ไขฟังก์ชั่นใดๆ developer ก็จะมี test class นั้นเป็นตัวกำกับเพื่อทำให้มั่นใจได้ว่า การปรับแต่งใดๆที่เกิดขึ้นกับฟังก์ชั่นนั้นจะไม่ไปทำให้ซอร์สโค้ดเดิมเกิดข้อผิดพลาด โดยอาศัยการตรวจสอบว่า เมื่อมีการปรับแต่งเกิดขึ้น test case เดิมทั้งหมดใน test class จะต้องรันผ่าน
7. ช่วยทำตัวเป็นเอกสาร เนื่องจากปกติแล้วการเขียน test case จะต้องมีการเขียน example หรือตัวอย่างการใช้งานเอาไว้ด้วย ดังนั้น unit test จึงช่วยทำหน้าที่เป็นเอกสาร สำหรับคนที่ต้องการจะศึกษาว่าเมทอดหรือคลาสใดๆนั้น ทำงานอย่างไร มีหน้าที่อะไร ก็สามารถมาดูได้ที่ test class ได้
8. JUnit

Junit [] คือ framework สำหรับช่วยในการทำ Unit testing ด้วยภาษา Java โดย Junit นั้นถูกพัฒนาเมื่อปี 1990 โดย Erich Gamma และ Kent Beck ซึ่ง JUnit เป็น Open Source ที่ใช้ช่วยในการทดสอบโปรแกรมที่เขียนขึ้นด้วยภาษา JAVA ซึ่งวิธีการทำงานของ Junit ประกอบไปด้วยการเขียน test และการรัน test โดยสามารถทำการรัน test ซ้ำหลายๆ ครั้งได้เพื่อใช้ในการทดสอบซ้ำในกรณีที่ซอร์สโค้ดมีการเปลี่ยนแปลง



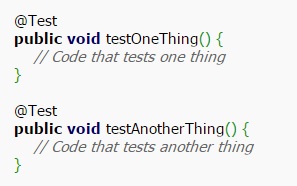
1. สัญลักษณ์ของ Junit Framework

สำหรับการพัฒนา unit test ด้วย Junit นั้นจะสามารถทำได้โดยการนำ jar file ของ Junit เข้ามาในโปรแกรม หลังจากนั้น test class หรือ Class ในภาษา Java ที่ถูกสร้างขึ้นเพื่อรวบรวม test method ต่างๆที่ใช้ทดสอบโปรแกรม จะใช้งาน Junit จะต้องทำการ import org.junit.\* ดังรูปที่ 2.x



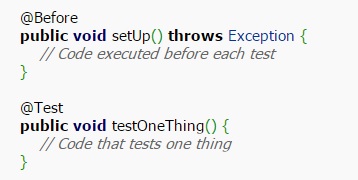
1. library ของ Junit

ส่วน test case หรือ test method ที่ใช้สำหรับทดสอบส่วนขอโปรแกรม นั้นให้ทำการกำกับด้วยคำว่า @Test เอาไว้เพื่อเป็นการบอก Junit ว่าเมทอดนั้นเป็น test method ดังรูปที่ 2.x ซึ่งจะมีประโยชน์ในการแยกเมทอดธรรมดาและเมทอดทดสอบ โดยเมื่อเราทำการสั่งทดสอบเมทอดทั้งหมดใน Test Class ของเรา JUnit จะทำการรันเฉพาะเมทอดที่มี @Test กำกับไว้เท่านั้น



1. @Test Annotation

โดย Junit มาพร้อมกับ บรรณนิทัศน์ (Annotation) หลายแบบซึ่งมีหน้าที่ต่างๆกันเพื่อช่วยเพิ่มประสิทธิภาพในการทำ unit test เช่น @Before เมื่อนำไปกำกับเมทอด เมทอดนั้นจะถูกรันก่อนที่จะทำ test case ทุกกรณี ดังเช่นรูปที่ 2.x ถ้าเราสั่งรันเมทอด testOneThing เมทอด setup จะทำงานก่อนเสมอ

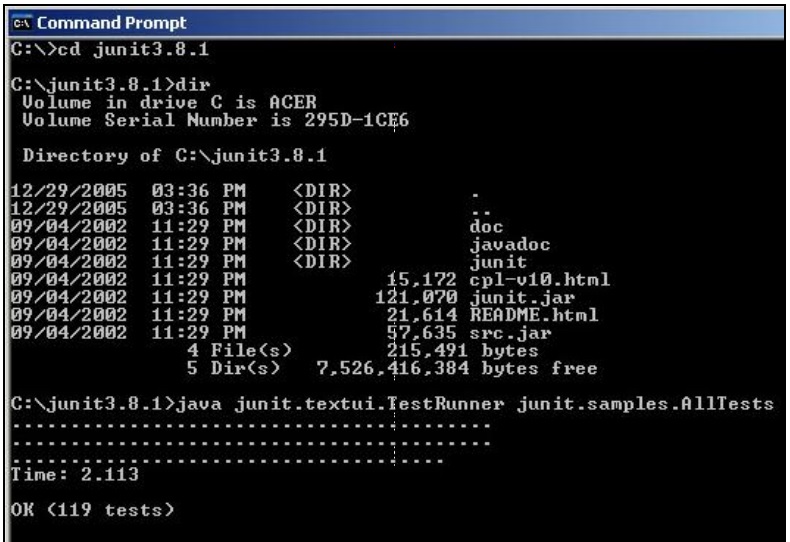


1. @Before Annotation

โดย Junit นั้นจะมีการรัน test อยู่ 2 รูปแบบ [4] คือ Batch test runner และ Graphical test runner โดย Batch test runner นั้นหมายถึงการสั่งงานการรัน test จะทำด้วย command line ส่วน Graphical test runner นั้นสามารถแบ่งออกได้เป็นแบบ awt-based และ swing-based ซึ่ง Graphical test runner จะสามารถสั่งรัน test ผ่านส่วนแสดงผลผู้ใช้ และสามารถแสดงรายงานออกมาได้อย่างชัดเจนถึงผลของการทดสอบ โดยวิธีการใช้งาน Junit จะกล่าวดังต่อไปนี้

* 1. Batch test runner

ให้ไปที่ command prompt และพิมพ์คำสั่ง java junit.textui.TestRunner junit.samples.TestClass โดยที่ junit.textui.TestRunner คือ library ของ Junit ส่วน junit.samples.TestClass นั้นคือ test class ที่มี test case อยู่ จากนั้นเมื่อรันจะแสดงผลดังรูปที่ 2.6

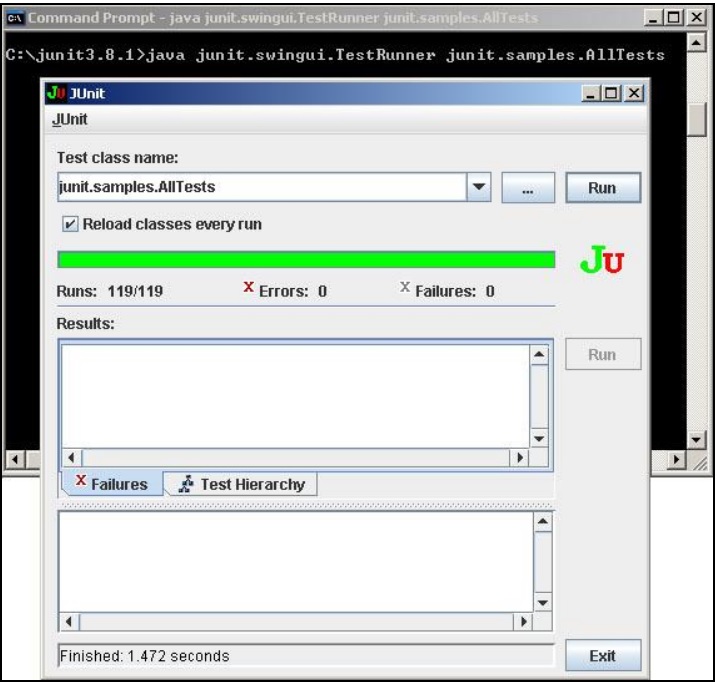


1. หน้าจอแสดงผล Batch test runner

จากภาพแสดงจำนวนจุด ( . ) มีจำนวน 119 module. Time: 2.113 หมายถึงเวลาที่ใช้ในการ Test. OK <119 tests> หมายถึงการ Test ผ่านการทดสอบหมด ไม่เกิด Error ใดๆ ขึ้นมา ดังนั้นหมายความว่า Code นี้ไม่มี error และผ่านการทดสอบโดยวิธี batch test

* 1. Graphical test runner

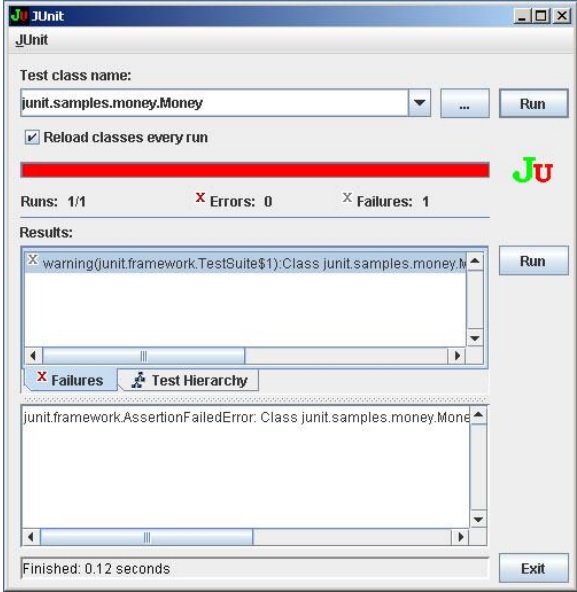
ให้ไปที่ command prompt และพิมพ์คำสั่ง Java junit.awtui.TestRunner junit.samples.TestClass หรือ Java junit.swingui.TestRunner junit.samples.TestClass โดยทั้ง 2 คำสั่งต่างกันตรงที่ graphic library ของ java ว่าจะใช้ Swing หรือ Awt คำสั่งทั้งสองนี้จะไปทำการรัน test ภายใน Test Class และแสดงผลออกมาในรูปกราฟิกดังรูปที่ 2.x



1. หน้าจอแสดงผล Graphical test runner

โดยรูปที่ 2.7 นั้นเป็นหน้าจอโปรแกรม JUnit ที่กำลังรันไฟล์ AllTests โดยขณะที่กำลังรัน แถบสีเขียวจะวิ่งจากด้านซ้ายไปด้านขวา และเมื่อรันเสร็จแล้วจะได้ดังภาพคือ แถบจะเป็นสีเขียวเมื่อการทดสอบเสร็จสิ้น โดย จากรูปภาพ 2.7 แสดงผลการ Run ดังนี้

Runs: 119 มายถึง การ Run จำนวน 119 test case, Errors: 0 หมายถึง ไม่พบ error, Failures: 0 หมายถึง ไม่พบ Failure, Finished: 1.182 Seconds หมายถึง เวลาที่ใช้ในการ Run test case ทั้งหมด ส่วนในกรณีที่มี error เกิดขึ้น แถบสีจะเป็นสีแดงและมีข้อความแสดง error ที่เกิดขึ้นดังรูปที่ 2.8



1. หน้าจอแสดงผลกรณี Junit ผิดพลาด
2. Test-driven development ( TDD )

Test-driven development หรือ TDD คือ [] กระบวนการพัฒนาซอฟต์แวร์รูปแบบหนึ่งซึ่งใช้หลักการคือ มีวงจรการพัฒนาที่สั้น แต่เน้นการทำซ้ำไปเรื่อยๆ โดยวงจรการพัฒนาซอฟต์แวร์แบบ TDD นั้นจะเป็นดังรูปที่ 2.x



1. วงจรการพัฒนาแบบ TDD

โดยวงจรการพัฒนาซอฟต์แวร์แบบ TDD นั้นจะเริ่มจากการที่ developer ทำการเขียน test case ซึ่ง test case นี้เป็น unit test ที่อธิบายฟังก์ชั่นใหม่ หรือ ความต้องการของระบบ ซึ่งเมื่อทำการรัน case ที่เขียนขึ้นก็จะพบว่าผลลัพธ์คือ test fail เนื่องจากยังไม่มีส่วน implementation ของโปรแกรม หลังจากนั้นขั้นตอนต่อมาคือการ พัฒนาโปรแกรม หรือเขียนซอร์สโค้ดขึ้นมาโดยมีจุดมุ่งหมายหลักคือ การทำให้ test case ที่เขียนขึ้นมาในขั้นตอนแรก รันผ่าน ซึ่งซอร์สโค้ดในขั้นตอนที่ 2 นี้ควรจะพยายามเขียนให้เล็กที่สุดโดยมีจุดประสงค์เพียงเพื่อให้ test case รันผ่านเท่านั้น หลักจากรันผ่านแล้ว ขั้นตอนสุดท้ายในวงจรการพัฒนาแบบ TDD คือการ refactor หรือการปรับปรุง จัดแต่ง ซอร์สโค้ดที่เขียนขึ้นให้สวยงาม อ่านง่าย ทำการ extract method หรือตั้งชื่อตัวแปรตาม standard เมื่อทำการ refactor เสร็จแล้วจะถือว่าจบ วงจรการพัฒนา 1 รอบหลังจากนั้นก็จะทำการวนรอบการพัฒนาไปเรื่อยๆจนโปรแกรมเสร็จสมบูรณ์.

โดยการนำ Test-driven development ไปประยุกต์ใช้พัฒนาโปรแกรมนั้น ไม่มีข้อจำกัดทางด้านภาษาโปรแกรมที่ใช้ในการพัฒนา เนื่องจาก TDD นั้นไม่ยึดติดกับภาษา และในส่วนของการเขียน test case ซึ่งเป็น unit test นั้น ได้มีผู้พัฒนา framework ที่ใช้ช่วยเขียน unit test สำหรับภาษาต่างๆไว้มากมาย โดย framework ที่เป็นนิยมใช้ในการพัฒนา unit test นั้น ยกตัวอย่างเช่น Junit สำหรับ java PHPUnit สำหรับภาษา PHP และ COBOLUnit สำหรับภาษา COBOL เป็นต้น

* 1. ข้อดีของการพัฒนาซอฟต์แวร์ด้วย Test-driven development [6]

1. ง่ายต่อการทดสอบระบบ เนื่องจากมีการเขียน test ตลอดเวลาในขั้นตอนการพัฒนาระบบ ทำให้สามารถใช้ test เหล่านั้นได้ทันที เมื่อต้องการทดสอบระบบ
2. ระบบที่ถูกพัฒนาด้วย TDD จะมีความเป็นโมดูล (Modularity) เนื่องจากการทำ unit test ให้กับแต่ละฟังก์ชั่นในการพัฒนา จะช่วยให้ระบบที่ค่อยๆพัฒนานั้นแบ่งออกเป็นส่วนย่อยๆโดยปริยาย ซึ่งเป็นข้อดีที่ช่วยให้การแก้ไข ทำได้ง่าย การแก้ไขส่วนใดไม่ไปกระทบส่วนอื่น
3. Test case ที่เขียนขึ้นนั้นเป็นการทำ document ไปในตัวถึงฟังก์ชั่นต่างๆในระบบ
4. ช่วยทำให้เจอ Error ที่เกิดขึ้นในการพัฒนาได้เร็ว ซึ่งเป็นส่วนที่สำคัญในการพัฒนาซอฟต์แวร์เนื่องจาก Error ที่เจอในตอนท้ายๆของการพัฒนานั้น การที่จะแก้ไขจะใช้ทรัพยากรมากกว่า Error ที่เจอตั้งแต่เนิ่นๆ
5. เป็นการทำให้ requirement มีความชัดเจน การเขียน test ของ TDD เป็นสิ่งที่กำหนด requirement ไปในตัว ดังนั้นการทำให้ test ทั้งหมดทำงานได้ ก็เป็นสิ่งบ่งชี้ว่าระบบของเราสามารถทำงานตาม requirement ได้ครบถ้วน และไม่มีการสร้าง code หรือฟังก์ชั่นที่ไม่จำเป็นในระบบ ซึ่งจะทำให้ระบบใหญ่เกินไป ซึ่งทำให้ระบบช้า และดูแลยาก
6. การพัฒนาระบบด้วย TDD สามารถนำไปสู่การทำ automated test เพื่อทดสอบระบบแบบอัตโนมัติได้
7. Behavior driven development (BDD)

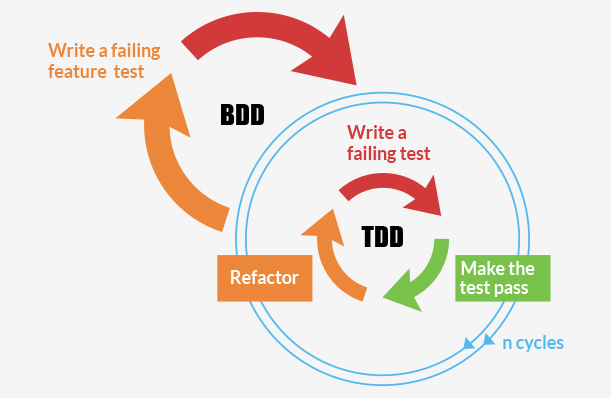
Behavior Driven Development (BDD) [4] เป็นวิธีการพัฒนาซอฟต์แวร์รูปแบบหนึ่ง ถูกพัฒนาโดย Dan North เพื่อแก้ปัญหาของ TDD ซึ่งมีดังนี้

1. ผู้ทดสอบไม่ทราบว่าควรจะเริ่มทดสอบจากตรงไหนก่อน
2. ผู้ทดสอบสร้างเทสเคสไม่ถูกต้อง ไม่ครอบคลุม หรือมากเกินความจำเป็น เนื่องจากไม่รู้พฤติกรรมของผู้ใช้
3. ในการทดสอบแต่ละครั้งต้องทดสอบละเอียดขนาดไหน
4. เทสเคสของ TDD เป็นภาษาที่มีความสอดคล้องกับ user story น้อย ทำให้ผู้ที่ไม่ใช่เทสเตอร์หรือโปรแกรมเมอร์เข้าใจได้ยาก
5. เมื่อเทสเคสรันไม่ผ่าน จะเข้าใจได้ยากว่าสาเหตุเกิดจากอะไร

ปัญหาที่เกิดขึ้นมาจากความเข้าใจที่ไม่ตรงกันของ development team และ business team เนื่องจากเทสเคสของ TDD นั้น จะเขียนด้วยภาษาที่โปรแกรมเมอร์เข้าใจ แต่ผู้ที่ไม่ค่อยมีความรู้ในการเขียนโปรแกรมมากนักจะทำความเข้าใจได้ยาก ไม่สามารถบอกพฤติกรรมของระบบและขอบเขตในการทดสอบได้ Dan North จึงได้คิดค้น BDD [5] ขึ้นมาเพื่อแก้ปัญหาการสื่อสารนี้ โดยการนำแนวคิดของ Acceptance Testing มาประยุกต์เข้ากับ TDD โดยเพิ่ม automated acceptance test เข้ามาเพื่อใช้ในการอธิบายพฤติกรรมของระบบในมุมมองของลูกค้า ซึ่งอยู่ในภาษาและรูปแบบที่เข้าใจได้ง่าย เพื่อให้ทุกคนในทีมสามารถอ่านเทสแล้วเข้าใจตรงกันว่าระบบสามารถทำอะไรได้บ้าง ถูกต้องตามความต้องการของลูกค้าหรือไม่ และต้องทดสอบอะไรบ้างลูกค้าจึงจะพอใจ โดยเราสร้าง acceptance test นี้ขึ้นมาเพื่อยืนยันความถูกต้องกับทาง stakeholders ก่อนที่จะลงมือพัฒนาระบบขึ้นมาจริงๆ และใช้ทดสอบระบบแบบ automated testing ช่วยให้ทีมที่ทำหน้าที่ในพัฒนาซอฟต์แวร์และทีมที่ทำหน้าที่ในการบริหารจัดการสื่อสารกันได้อย่างมีประสิทธิภาพ เข้าใจตรงกันมากขึ้น สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

* 1. ขั้นตอนการพัฒนาระบบแบบ BDD

ขั้นตอนการพัฒนาระบบแบบ BDD [6] จะเป็นการพัฒนาโปรแกรมแบบทำซ้ำเป็นรอบๆ โดยจะเริ่มจากการเขียน automated acceptance test ขึ้นมาก่อน เพื่อเป็นการกำหนดพฤติกรรมของระบบที่เราจะพัฒนาขึ้นมา ว่าสามารถทำอะไรได้บ้าง ซึ่งเทสเคสที่เราจะเขียนใน acceptance test เราจะเขียนด้วยภาษาที่ทุกคนสามารถทำความเข้าใจและสามารถเขียนได้ง่าย ซึ่งภาษาที่ใช้เขียนเป็นภาษาโปรแกรมระดับสูง โดยผู้ที่ทำหน้าที่ในการเขียน acceptance test สามารถเป็นใครก็ได้ ไม่จำเป็นต้องเป็นโปรแกรมเมอร์หรือเทสเตอร์ หลังจากเราที่เขียน acceptance test ขึ้นมาแล้ว ขั้นตอนต่อไปก็เป็นหน้าที่ของโปรแกรมเมอร์ในการเขียน unit test ขึ้นมาเพื่อรองรับเทสเคสแต่ละเทสเคสที่เราเขียนไว้ใน acceptance test หลังจากที่เขียน unit test แล้ว โปรแกรมเมอร์ก็จะทำการพัฒนาซอร์สโค้ดเพื่อให้โปรแกรมรันผ่านเทสที่เขียนขึ้นมา จากนั้นจึงทำการปรับปรุงซอร์สโค้ดและเทสต่างๆ ให้มีประสิทธิภาพมากยิ่งขึ้น ซึ่งเป็นขั้นตอนสุดท้ายในการพัฒนาระบบแบบ BDD ในแต่ละรอบ



1. วงจรการพัฒนาแบบ BDD [6]

ตัวอย่าง Automated Acceptance Test ที่เขียนด้วยภาษา Gherkin [7]

**Title:** Customer withdraws cash  
**As a**customer,  
**I want** to withdraw cash from an ATM,  
**so that** I don’t have to wait in line at the bank.

**Scenario:** Account is in credit+  
**Given** the account is in credit  
**And** the card is valid  
**And** the dispenser contains cash  
**When** the customer requests cash  
**Then** ensure the account is debited  
**And** ensure cash is dispensed  
**And** ensure the card is returned

**Scenario:** Account is overdrawn past the overdraft limit+  
**Given** the account is overdrawn  
**And** the card is valid  
**When** the customer requests cash  
**Then** ensure a rejection message is displayed  
**And** ensure cash is not dispensed  
**And** ensure the card is returned

* 1. คุณสมบัติและข้อดีของ Behavior-Driven Development [7]

1. เป็นการนำแนวคิดของ Acceptance Testing กับ Unit Testing มารวมเข้าด้วยกัน ซึ่งการทำ Acceptance Testing จะช่วยให้รู้ว่าควรเทสละเอียดขนาดไหนลูกค้าจึงจะพอใจ
2. ชื่อเทสเคสอยู่ในรูปประโยคที่เข้าใจง่าย แทนที่จะเป็นเพียงแค่ชื่อ class และชื่อ method
3. รูปแบบในการตั้งชื่อเทสจะใช้คำว่า should เพื่อเป็นการบอกพฤติกรรมของแต่ละเทสว่าทำหน้าที่อะไร ช่วยให้แบ่งพฤติกรรมหรือหน้าที่การทำงานของแต่ละเทสได้อย่างชัดเจน
4. คำนึงถึงพฤติกรรมของระบบว่าระบบควรจะทำอะไรได้บ้าง มากกว่าแค่การทดสอบว่าแต่ละคลาสทำงานถูกต้อง
5. Requirement ที่ได้รับมาจากลูกค้าคือสิ่งที่ระบุพฤติกรรมของระบบ และเป็นสิ่งที่นำมาเขียนเทสเพื่อทดสอบว่าระบบมีพฤติกรรมหรือการทำงานที่ถูกต้องหรือไม่
6. เทสอยู่ในรูปแบบ user story ซึ่ง business team คุ้นเคย ส่วน acceptance criteria จะเขียนอยู่ในรูปของ scenario หรือตามสถานการณ์ที่ต้องการจะทดสอบ
7. Acceptance criteria สามารถใช้รันเพื่อทดสอบระบบแบบ Automated Testing ได้ ทำให้ automated acceptance test เป็นทั้งเอกสารที่ใช้อธิบายพฤติกรรมของระบบและสามารถรันเพื่อทดสอบระบบได้ (living documentation)
8. ภาษาจาวา (Java)

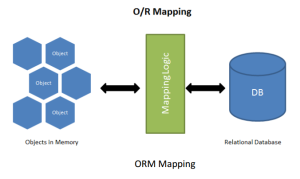
Java หรือ Java programming language [7] คือภาษาโปรแกรมเชิงวัตถุ พัฒนาโดย เจมส์ กอสลิง และวิศวกรคนอื่นๆ ที่บริษัท ซัน ไมโครซิสเต็มส์ ภาษานี้มีจุดประสงค์เพื่อใช้แทนภาษาซีพลัสพลัส C++ โดยรูปแบบที่เพิ่มเติมขึ้นคล้ายกับภาษาอ็อบเจกต์ทีฟซี (Objective-C) แต่เดิมภาษานี้เรียกว่า ภาษาโอ๊ก (Oak) ซึ่งตั้งชื่อตามต้นโอ๊กใกล้ที่ทำงานของ เจมส์ กอสลิง แล้วภายหลังจึงเปลี่ยนไปใช้ชื่อ "จาวา" ซึ่งเป็นชื่อกาแฟแทน จุดเด่นของภาษา Java อยู่ที่ผู้เขียนโปรแกรมสามารถใช้หลักการของ Object-Oriented Programming มาพัฒนาโปรแกรมของตนด้วย Java ได้

ภาษา Java เป็นภาษาสำหรับเขียนโปรแกรมที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ ( OOP : Object-Oriented Programming) โปรแกรมที่เขียนขึ้นถูกสร้างภายในคลาส ดังนั้นคลาสคือที่เก็บ เมทอด (Method) หรือพฤติกรรม (Behavior) ซึ่งมีสถานะ (State) และรูปพรรณ (Identity) ประจำพฤติกรรม (Behavior)



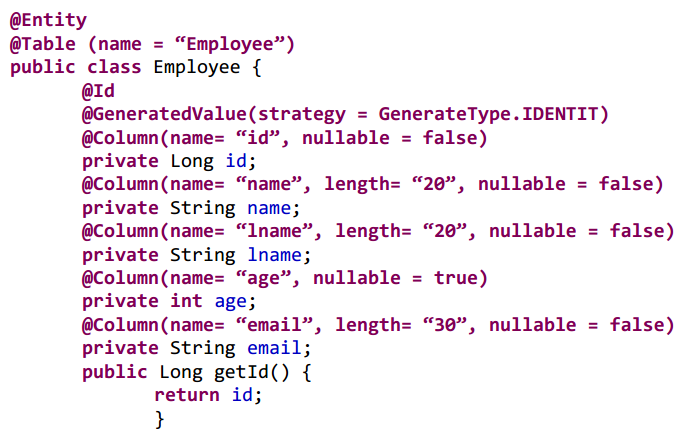
1. สัญลักษณ์ภาษา Java
   1. ข้อดีของภาษาจาวา
2. ภาษาจาวาเป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุแบบสมบูรณ์ ซึ่งเหมาะในการพัฒนาระบบที่มีความซับซ้อน
3. โปรแกรมที่เขียนขึ้นโดยใช้ภาษาจาวาสามารถทำงานได้ในระบบปฏิบัติการที่แตกต่างกัน
4. ภาษาจาวามีการตรวจสอบข้อผิดพลาดทั้งตอนคอมไพล์และเวลารันทาให้ลดข้อผิดพลาดที่อาจเกิดขึ้นในโปรแกรม และช่วยให้แก้จุดบกพร่องของโปรแกรมได้ง่าย
5. ภาษาจาวามีความซับซ้อนน้อยกว่าภาษา C++ ทำให้ใช้งานได้ง่ายกว่าและลดความผิดพลาดได้มากขึ้น
6. ภาษาจาวาถูกออกแบบมาให้มีความปลอดภัยสูงตั้งแต่แรก ทำให้โปรแกรมมีความปลอดภัยมากกว่าโปรแกรมที่เขียนขึ้นด้วยภาษาอื่น
7. มี IDE แอพพลิเคชันเซิร์ฟเวอร์ และไลบลารีมากมายสาหรับจาวาที่สามารถใช้งานได้โดยไม่ต้องเสียค่าใช้จ่าย
   1. ข้อเสียของภาษาจาวา
8. ทำงานได้ช้ากว่า Native code เพราะว่าโปรแกรมที่เขียนขึ้นด้วยภาษาจาวาจะถูกแปลงเป็นภาษากลางก่อน หลังจากนั้นจึงจะถูกเปลี่ยนเป็นภาษาเครื่องซึ่ง Native code อยู่ในรูปของภาษาเครื่องอยู่แล้ว
9. Tools ที่ใช้ในการพัฒนาโปรแกรมจาวามักไม่ค่อยครอบคลุม ทาให้หลายอย่างโปรแกรมเมอร์จะต้องเป็นผู้พัฒนา
10. JPA (Java Persistence API)

JPA [] คือ API มาตรฐานตัวหนึ่งของ Java ที่ใช้ในการทำ ORM (Object Relation Mapping) โดย ORM คือ เครื่องมือที่ช่วยให้เราสามารถมองข้อมูลภายในฐานข้อมูลให้เป็น Object และสามารถใช้ Class ในภาษา Java ในการทำการเชื่อมโยงข้อมูลในฐานข้อมูล เช่น ตาราง คอลัมน์ ข้อมูลในแต่ละแถว และความสัมพันธ์ต่างๆระหว่างตาราง ออกมาในรูปของ Class Object ในภาษา Java ดังนั้นถ้าเราแก้ไขข้อมูล Object ที่ทำหน้าที่เชื่อมโยงกับตารางก็จะทำให้ข้อมูลในฐานข้อมูลจริงมีการเปลี่ยนแปลง เหมือนกับการใช้ SQL Command ในการจัดการตรงๆ ซึ่งการทำ ORM จะช่วยให้เราสามารถจัดการกับข้อมูลในตารางได้ง่ายกว่าด้วยภาษาโปรแกรมมิ่ง โดยแผนภาพแสดงการทำงานของกระบวนการ object relation mapping จะแสดงดังรูปที่ 2.12



1. แผนภาพวิธีการทำงานของ object relation mapping

ซึ่งการทำ ORM จะทำให้เราไม่จำเป็นต้องติดต่อกับฐานข้อมูลโดยตรงด้วยการเขียนคำสั่ง SQL ที่เป็นการติดต่อพื้นฐานของ JDBC (Java database connectivity technology) ซึ่งจะช่วยแก้ปัญหาที่ว่า ถ้าระบบมีขนาดใหญ่ หากซอฟต์แวร์ทำการพัฒนาด้วยการติดต่อกับฐานข้อมูลผ่าน JDBC ก็จะต้องทำการเขียนคำสั่ง SQL command ที่ซับซ้อนและซับซ้อน และกระจัดการจาย เนื่องจากต้องเขียนซอร์สโค้ดจำนวนมากในการทำงานกับข้อมูลในแต่ละส่วน ทำให้การจัดการดูแลทำได้ยาก ยากต่อการค้นหาและแก้ไขเมื่อมีปัญหาเกิดขึ้น โดยการทำ ORM นั้น ยกตัวอย่างเช่น เรามีตารางฐานข้อมูล เก็บข้อมูลลูกค้า หากเรามองตารางนั้นเป็น Object ก็จะทำการเขียน Class ของ object นั้น ดังรูปที่ 2.13



1. ORM Class ของ Java persistence API

ซึ่งใน Class employee จะมี annotation สำคัญๆ เพื่อใช้บ่งบอกถึงค่าต่างๆในตารางดังนี้

1. @Entity ใช้สำหรับบอกว่าเป็น Class ที่ทำหน้าที่เป็นตัวแทนของตารางในฐานข้อมูล
2. @id และ @GeneratedValue ใช้สำหรับกำหนด Field ที่เป็น Primary Key กรณีที่เป็น id ประเภท Auto Generate จะใช้ @GeneratedValue
3. @Column ใช้สำหรับกำหนดรายละเอียดของ column ของฐานข้อมูล เช่น ความยาว ชื่อ column หรือ มีค่าเป็น null ได้หรือไม่

ดังนั้น Class Employee ภาษา Java จากรูปที่ 2.17 นั้น จึงทำหน้าที่เป็นตัวแทนของตารางจริงในฐานข้อมูลที่ชื่อว่า Employee ซึ่งประกอบไปด้วย 5 คอลัมน์ คือ

* Id
* Name
* Lname
* Age
* Email

1. Spring framework

Spring framework หรือ Spring [] คือ framework ตัวหนึ่งของภาษา Java ที่ถูกสร้างขึ้นมาเพื่อช่วยให้การพัฒนาซอฟต์แวร์ด้วยภาษา java ทำได้ง่ายขึ้น ช่วยสนับสนุนในทุกๆแง่มุมโครงสร้างที่ช่วยในการพัฒนาระบบซอฟต์แวร์ขึ้นมา เช่นในส่วนของ Persistence, Transaction management, security และอื่นๆอีกมากมาย ซึ่งส่วนประกอบเหล่านี้เป็นส่วนประกอบพื้นฐาน ที่จำเป็นต้องมีในการพัฒนาระบบขนาดใหญ่หรือ Enterprise software ทุกระบบ โดย Spring framework ได้ทำการรวบรวมส่วนประกอบเหล่านี้เอาไว้ให้เพื่อให้เรียกใช้ หรือนำไปพัฒนาต่อยอด หรือปรับแต่ง ทำให้การพัฒนาซอฟต์แวร์ประเภท Enterprise นั้น ผู้พัฒนาสามารถที่จะมุ่งเน้นไปที่การพัฒนาตัวโปรเซส และbusiness logic ของระบบ ทำให้ไม่ต้องเสียเวลาในการพัฒนาซอฟต์แวร์ขนาดใหญ่ มาพัฒนาส่วนเหล่านี้ ซึ่งเป็นสิ่งที่ Enterprise software ทุกตัวต้องใช้อยู่แล้ว



1. สัญลักษณ์ Spring framework

Spring framework มีจุดเด่นในการเน้นไปที่การใช้ POJO (Plain old java object) หรือ object ธรรมดาของ Java เพื่อใช้ในการพัฒนาระบบทั้งระบบ ทำให้สามารถจัดการคลาสต่างๆในระบบได้ง่าย ซึ่งเป็นจุดเด่นของการใช้ Spring framework เหนือการใช้ EJB (Enterprise Java bean) ที่เน้นทำให้คลาสที่ใช้ทำงานมีขนาดใหญ่และซับซ้อน โดย Spring framework มีหัวใจหลักในการทำงานด้วยสิ่งที่เรียกว่า Ioc (Inversion of control) และ DI (Dependency injection) ซึ่งวิธีการทางซอฟต์แวร์ทั้ง 2 ตัวนี้ทำให้ Spring เป็น framework ที่พัฒนาระบบที่มีการจัดการซอร์สโค้ดอย่างเป็นระบบ ทดสอบได้ง่าย แบ่งระบบออกเป็นส่วนต่างๆได้ง่าย

Spring framework มาพร้อมกับโมดูลย่อยๆประมาณ 20 โมดูลดังรูปที่ 2.15



1. โมดูลของ Spring framework

โดยโมดูลทั้งหมดจะถูกรวบรวมเป็นกลุ่มๆตามหน้าที่เป็นกลุ่มหลักๆคือ 1) Core container 2) Web 3) AOP (Aspect Oriented Programming) 4) Instrumentation 5) Messaging 6) Data Access/Integration และ 7) Test ซึ่งผู้พัฒนาสามารถเลือกนำไปใช้เฉพาะโมดูลที่ตนเองต้องการเพื่อลด overhead ในการนำ spring ทุกโมดูลเข้าไปในการพัฒนา และแต่ละส่วนมีหน้าที่สำคัญในการพัฒนาระบบขนาดใหญ่ การใช้ Spring framework กับโปรเจคจึงช่วยลดต้นทุน ในการพัฒนาได้อย่างมาก

* 1. ข้อดีของ Spring framework []

1. การใช้ Spring framework ทำให้ไม่ต้องพัฒนาระบบ หรือซอฟต์แวร์ตั้งแต่แรกเริ่ม เนื่องจากว่า Spring พัฒนาขึ้นโดยการรวบรวมเทคโนโลยีที่เป็นที่นิยม เช่น ORM frameworks, logging frameworks, JEE, JDK timers, Quartz และอื่นๆ มาทำให้ดีขึ้น ทำให้ผู้พัฒนาระบบไม่จำเป็นต้องไปศึกษา เทคโนโลยีพื้นฐานอื่นๆ เนื่องจาก Spring ได้รวบรวมไว้ให้แล้ว
2. พื้นฐานของ Spring คือ Dependency injection ดังนั้น ระบบที่พัฒนาด้วย Spring จึงมีความสามารถในการทดสอบได้ง่าย การทดสอบระบบที่พัฒนาจาก Spring สามารถทดสอบได้ง่าย เนื่องจาก Dependency injection ช่วยทำให้การพัฒนาระบบเป็นไปแบบแยกส่วน ดังนั้นการทดสอบจึงสามารถทดสอบแบบแยกส่วน แล้วทำการ inject ส่วนที่ไม่ต้องการทดสอบเป็น mock object เข้ามาได้ ซึ่ง Spring สนับสนุนการทำ mocking เป็นอย่างดี
3. Spring มีการจัดการด้วย Inversion of control และ API ต่างๆเป็นอย่างดี ทำให้การจัดการกับ exception ต่างๆที่ Throw มาจาก JDBC, Hibernate และอื่นๆ ทำได้ง่าย
4. การแบ่งส่วนของโปรแกรมโดย Spring เป็นไปอย่างมีระบบ มีการแยก package ของ Class ต่างๆตามหน้าที่ ทำให้เราสามารถรู้ได้ว่า package ไหนที่จะใช้งาน หรือไม่ใช้งาน และสามารถจัดการกับ class ต่างๆที่มีจำนวนมากได้อย่างมีประสิทธิภาพ
5. Spring มี interface สำหรับการทำการจัดการ transaction เป็นอย่างดีทำให้สามารถจัดการกับระบบที่มีขนาดระดับเล็ก ไปจนถึงใหญ่ได้เป็นอย่างดี
   1. ข้อเสียของ Spring framework []
6. การนำเอา jar file หรือ library ของ Spring เข้ามาใช้ในระบบนั้นเป็นการเพิ่ม overhead ให้กับระบบ เนื่องจาก jar file ของ Spring framework นั้นมีขนาดที่ค่อนข้างใหญ่ ประกอบไปด้วย Class อย่างน้อย 3000 Class ดังนั้นหากตัวระบบที่เราจะพัฒนานั้นมีขนาดเล็ก หรือไม่จำเป็นที่จะต้องใช้ framework สำหรับ enterprise การนำเอา Spring มาใช้จะเป็นการเพิ่ม overhead อย่างมากให้กับระบบ
7. หากจุดประสงค์การนำ Spring มาใช้กับระบบของเรา เพียงเพื่อจะต้องการใช้ความสามารถในการทำ Dependency injection ของ Spring เท่านั้น การเลือกใช้ Spring framework อาจจะไม่ใช่ทางเลือกที่ดีที่สุด เนื่องจากยังมี framework สำหรับการทำ Dependency อื่น เช่น Guice, Pico, anecdotal ซึ่งได้อธิบายไว้ว่ามีความเร็วมากกว่าการใช้ Spring
8. การทำงานกับบางส่วนของ Spring นั้นมีความยุ่งยาง เนื่องจากบางคอมโพเนนท์หลักของ Spring มีการเขียนเอกสารอธิบายวิธีใช้ไว้ได้อย่างไม่ดีเพียงพอ ทำให้ทำความเข้าใจ และนำไปใช้ได้ยาก
9. การตั้งค่าต่างๆภายใน Spring framework นั้นส่วนมากจะจัดการด้วย XML ทำให้ยากต่อการเข้าใจ และค้นหาต่างๆ แม้ว่าในภายหลังเริ่มมีการใช้ Class configuration เข้ามาช่วยแต่ยังมีอีกหลายๆส่วนที่ยังคงเป็น XML
10. Spring MVC
    1. Spring MVC Framework [] คืออะไร?

Spring MVC framework [] คือ framework ตัวหนึ่งซึ่งอยู่ภายใต้ ในการสร้างเว็บ ที่รองรับแนวคิดแบบ MVC (Model, View, Controller) นั่นเอง โดยหลักการทำงานของตัวSpring MVC framework จะออกแบบให้การทำงานทุกอย่างขึ้นอยู่กับ Servlet ที่ชื่อว่า DispatcherServlet ยกตัวอย่างเช่น เวลามี Request จาก User Client เข้ามาตัว DispatcherServlet ก็จะต้องทำหน้าที่เป็น Operation รับ Request นั้นๆไว้ก่อน แล้วค่อยส่งต่อไปให้ตัว Controller อื่นๆทำงานต่อไป

* 1. จุดประสงค์ของ Spring MVC Framework?

ตัว Spring MVC framework มีการออกแบบโครงสร้างการเก็บชิ้นส่วนของเว็บ เช่น พวก หน้าเว็บ (ไฟล์ html, ไฟล์ jsp ) ไฟล์ CSS หรือ ไฟล์ Script ต่างๆ ไว้อย่างชัดเจน เช่น หน้า ส่วน Controller ต้องอยู่ภายใน โฟเดอร์ src เท่านั้น ซึ่งทำให้สะดวกต่อการค้นหา แก้ไข เพิ่มเติม หรือ ลบทิ้ง เพราะไม่ว่าจะเป็น developer คนใดในทีมเป็นคนสร้างไฟล์ ก็ต้องถูกเก็บอยู่ตามโครงสร้างที่ Spring MVC Framework กำหนดไว้เท่านั้น โดยตัว Spring MVC framework มีการกำหนด Form การเขียนเว็บในแต่ละส่วน ไม่ว่าจะเป็นส่วน Request , Respond หรือ Controller ต่างๆ ทำให้ไม่ว่าจะเป็น developer คนไหนที่เข้ามาแก้ไขไฟล์ก็สามารถเข้าใจ Code ที่เขียนไว้ได้ง่ายๆ และยังมีตัว Library ที่ไว้ช่วยในการเขียน เพื่อให้ง่ายต่อการทำงานของ developer อีกด้วย

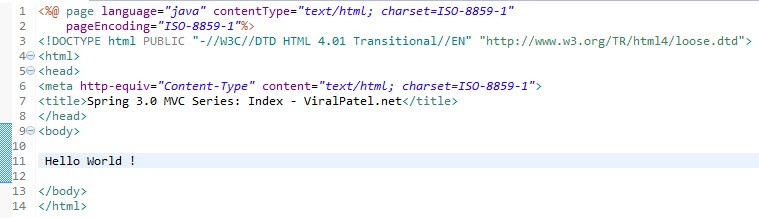
* 1. ตัวอย่างการเขียน Spring MVC framework

รูปที่ 2.16 คือไฟล์ที่มี Dispatcher Servlet



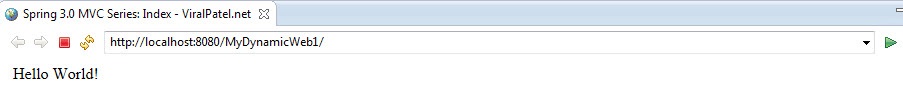
1. ไฟล์ Dispatcher servlet

รูปที่ 2.17 คือไฟล์ View ที่อ้างอิงจาก Dispatcher Servlet



1. ไฟล์ View ของ Spring MVC

หน้าผลลัพธ์ที่แสดงเมื่อส่ง request ผ่าน Dispatcher Servlet และได้รับ View กลับมาแสดงดังรูปที่ รูปที่ 2.18



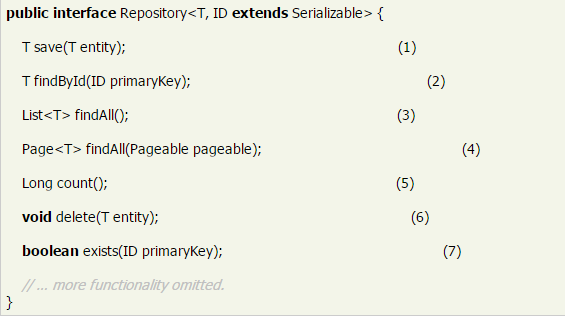
1. หน้าผลลัพธ์ของ Spring MVC
2. Spring Data JPA

Spring Data JPA [] คือ API ซึ่งอยู่ในเซตของ API ใน Spring framework ซึ่งทำหน้าที่ในการช่วยสนับสนุนในการสร้างชั้นส่วนติดต่อของข้อมูลที่เรียกว่า Data access layer ให้ง่ายขึ้น ลดการเขียนซอร์สโค้ดในการเข้าถึงฐานข้อมูลที่ซ้ำซ้อน ซึ่ง Data access layer คือชั้นของการติดต่อกับข้อมูล เพื่อให้ได้ข้อมูลจากฐานข้อมูลมา โดยเรียกใช้ผ่านเมทอดที่เขียนขึ้นอยู่ในคลาสที่เราสร้างขึ้นเพื่อใช้เป็นตัวแทนของเซตของข้อมูลในฐานข้อมูล ซึ่งเมทอดที่เขียนขึ้นสำหรับ Data access layer นั้นมักจะยึดติดกับกับส่วน implementation โดย API ที่ทำหน้าที่ในการสร้าง Data access layer เช่น Java persistence API เป็นต้น

เนื่องจากว่าในการสร้าง Data access layer ขึ้นมานั้นเป็นงานที่ซ้ำซ้อน จำเป็นที่จะต้องเขียนคำสั่งในการทำงานกับฐานข้อมูลที่มักจะเหมือนกันต่างกันเพียงแค่ตารางที่ต้องการ ดังนั้นการใช้ Spring data JPA จะช่วยลดการเขียนซอร์สโค้ดที่ซ้ำซ้อน ช่วยสร้างส่วนติดต่อกับข้อมูลที่จะช่วยประหยัดเวลาในการสร้างชั้น Data access layer อีกทั้งยังทำให้โปรแกรมที่เขียนขึ้นมีความยืดหยุ่น สามารถที่จะเปลี่ยนประเภทของฐานข้อมูล หรือผู้ให้บริการ โดยไม่กระทบกับโปรแกรมและส่วนของการติดต่อข้อมูล

* 1. ส่วนประกอบหลัก []

ส่วนติดต่อกลางซึ่งเป็นส่วนติดต่อพื้นฐานของ Spring data jpa ก็คือ ส่วนติดต่อที่มีชื่อว่า Repository ซึ่งรับค่าเป็น Class หลักที่เป็นตัวแทนของเซตของข้อมูลหรือ Entity class และ Id หรือคีย์หลักของตาราง ซึ่งส่วนติดต่อหลักนี้จะช่วยลดความซับซ้อนในการสร้างเมทอดในการทำการสร้าง, แก้ไข, ลบ หรืออัพเดตข้อมูลสำหรับ Entity โดยส่วนติดต่อ Repository จะแสดงดังรูปที่ 2.12



1. ส่วนติดต่อโปรแกรม Repository

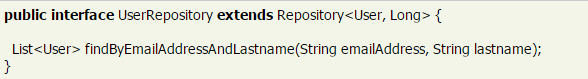
โดยเมทอดที่อยู่ภายใต้ส่วนติดต่อนี้จะมีหน้าที่ดังนี้

1. บันทึกข้อมูล Entity
2. คืนค่า Entity ที่มี Id ตรงกันกับคีย์
3. คืนค่า Entity ทั้งหมดใน domain
4. คืนค่า Page ของ Entity
5. คืนค่าจำนวนของ Entity ทั้งหมด
6. ลบ Entity ที่ระบุ
7. คืนค่า Boolean เพื่อบอกว่า Entity ที่ค้นหามีอยู่หรือไม่

สำหรับเมทอดต่างๆภายในส่วนติดต่อ Repository นั้นเป็นเมทอดพื้นฐานในการกระทำการกับข้อมูลในฐานข้อมูล สำหรับเมทอดพิเศษที่ใช้ค้นหาข้อมูล Entity หรือเมทอด query ที่จำเพาะนั้น การสร้างเมทอดเหล่านั้นด้วย Spring data jpa จะทำได้โดยการใช้ฟังก์ชั่นของ Spring data jpa ซึ่งจะช่วยให้ไม่ต้องทำการเขียนการค้นหาด้วยตัวเอง

* 1. การสร้างเมทอดค้นหา

ในการสร้างเมทอดค้นหา เพื่อค้นหาเฉพาะข้อมูลตามเงื่อนไขที่ต้องการนั้น โดยปกติแล้วจะต้องทำการเขียนคำสั่งเงื่อนไขต่างๆภายในเมทอด เพื่อกระทำการดังกล่าว ซึ่งการค้นหาเหล่านี้มักจะถูกเขียนขึ้นด้วยเงื่อนไขที่ใกล้เคียงกัน ทำให้เกิดการเขียนซอร์สโค้ดซ้ำๆกันเกิดขึ้น ดังนั้น Spring data jpa จึงได้จัดเตรียมส่วนติดต่อสำหรับการสร้างการค้นหาเหล่านี้ โดยไม่จำเป็นต้องเขียนการค้นหาขึ้นด้วยตัวเอง โดยตัวอย่างเมทอดการค้นหาด้วย Spring data jpa จะเป็นดัง รูปที่ 2.13



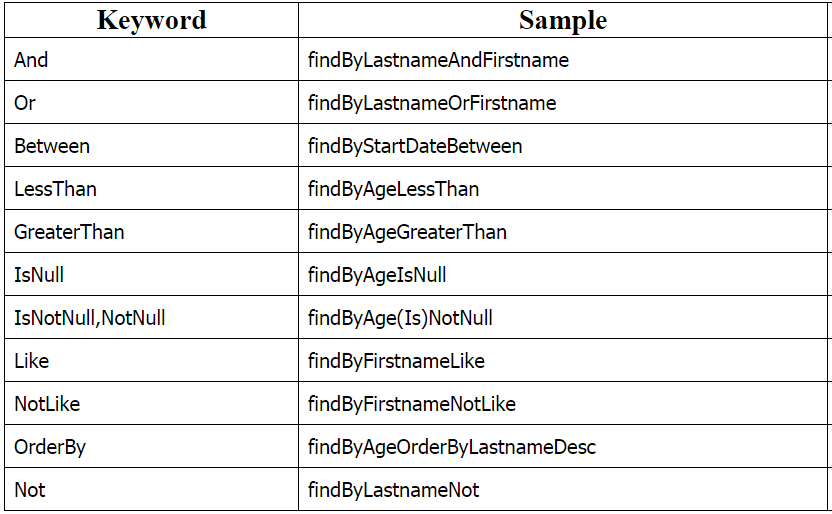
1. ตัวอย่างเมทอดค้นหาของ Spring data jpa

จากรูปจะเห็นได้ว่า เมทอดค้นหาซึ่งสร้างจาก API นี้จะไม่มีซอร์สโค้ดในส่วนของการเขียนคำสั่งค้นหาหรือ implementation ด้วยตัวเองเลย ซึ่งจากรูปนั้นคำสั่งนี้จะมีผลกับการเขียนคำสั่ง JPQL ดังรูปที่ 2.14



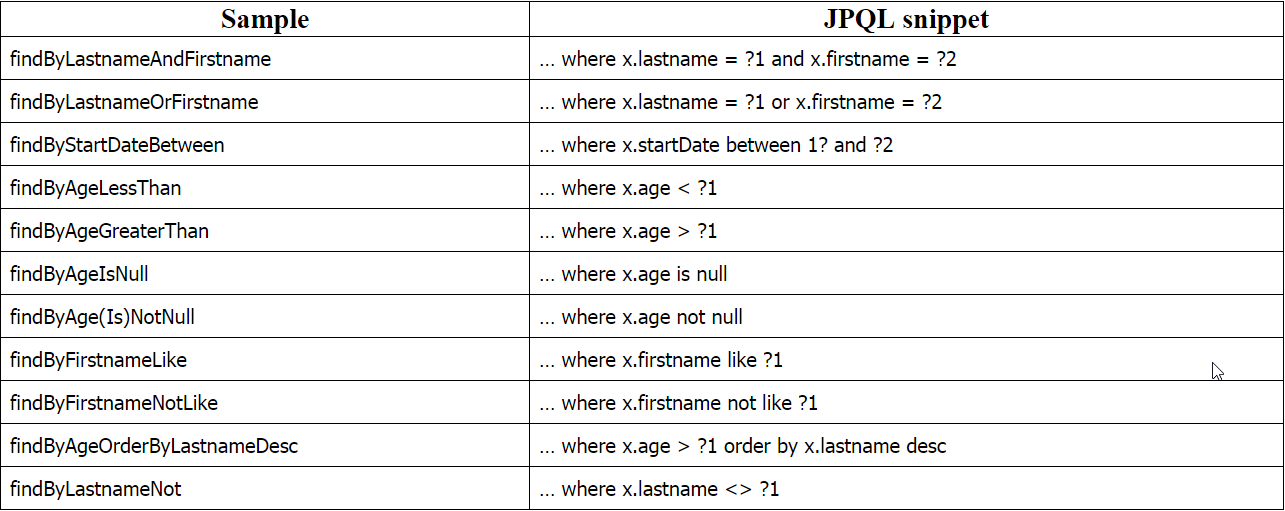
1. คำสั่ง JPQL ของเมทอดค้นหาใน Spring data jpa

ซึ่งมีความหมายว่าให้ทำการค้นหา Entity ของ class User ที่มีค่าของฟิลด์ emailAddress และ lastname ตรงกับที่ต้องการ ซึ่งหากต้องทำการเขียนการ implement ในส่วนนี้เองในทุกๆการค้นหาจะเป็นการเขียนโค้ดที่มีความซ้ำซ้อนและดูแลยาก สำหรับการเขียนเมทอดค้นหาของ Spring data jpa นั้นสามารถดูคีย์เวิร์ดในการเขียนได้จากรูปที่ 2.15



1. รูปการเปรียบเทียบไวยากรณ์การสร้างเมทอดค้นหา

โดยคีย์เวิร์ดในการตั้งชื่อเมทอดที่ใช้ได้จะแสดงดังรูป และสำหรับความหมายของเมทอดเมื่อนำไปเปรียบเทียบกับการเขียนด้วยคำสั่งภาษา JPQL จะมีลักษณะดังรูปที่ 2.16



1. รูปการเปรียบเทียบเมทอดค้นหากับคำสั่ง JPQL
2. Log file (processing log)

ในการประมวลผลด้วยระบบคอมพิวเตอร์นั้น log file [] ก็คือไฟล์ที่ทำหน้าที่ในการบันทึกเหตุการณ์ที่เกิดขึ้นในทั้งระบบปฏิบัติการณ์เอง, เหตุการณ์ที่เกิดขึ้นในซอฟต์แวร์ต่างๆ หรือเหตุการณ์การการรับส่งข้อมูลกันระหว่างคอมพิวเตอร์ผ่านทางการสื่อสารข้อมูล

โดยการเก็บ log ก็คือการบันทึกเอาข้อมูลที่เกิดขึ้น ซึ่งอาจจะประกอบไปด้วยเวลา ข้อความที่แสดงถึงเหตุการณ์ และทำการบันทึกข้อความเหล่านั้นลงไปใน log file ที่ต้องการเพื่อสามารถกลับมาย้อนดูในภายหลังได้

1. Web service

เว็บเซอร์วิส (อังกฤษ: web service บริการบนเว็บ) [] คือระบบซอฟต์แวร์ที่ถูกออกแบบมาเพื่อสนับสนุนการแลกเปลี่ยนข้อมูลกันระหว่างเครื่องคอมพิวเตอร์ ผ่านระบบเครือข่าย. โดยภาษาที่ใช้ในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ คือภาษา เอกซ์เอ็มแอล (XML) หรือ Extensible Markup Language. เว็บเซอร์วิสมีอินเทอร์เฟส ที่ใช้อธิบายรูปแบบข้อมูลที่เครื่องคอมพิวเตอร์ประมวลผลได้ เช่น WSDL. ระบบคอมพิวเตอร์ใช้งานสื่อสารโต้ตอบกับเว็บเซอร์วิสตามรูปแบบที่ได้กำหนดไว้แล้ว โดยการส่งสาสน์ตามอินเตอร์เฟสของเว็บเซอร์วิสนั้น โดยที่สาสน์ดังกล่าวอาจแนบไว้ใน SOAP envelope หรือส่งตามอินเตอร์เฟสในแนวทางของ REST. ปกติแล้วข้อมูลจะถูกส่งโดยอาศัย protocol HTTP และใช้ XML ร่วมกับมาตรฐานเกี่ยวกับเว็บอื่นๆ. โดยโปรแกรมประยุกต์ที่เขียนด้วยภาษาต่างๆ และทำงานบนหลากหลายแพลตฟอร์มต่างกัน สามารถที่จะใช้เว็บเซอร์วิสเพื่อแลกเปลี่ยนข้อมูลผ่านทางเครือข่ายคอมพิวเตอร์ เช่น อินเทอร์เน็ต. ความสามารถในการแลกเปลี่ยนข้อมูลระหว่างระบบที่ต่างกันนี้ (เช่น การแลกเปลี่ยนข้อมูลระหว่าง โปรแกรมที่เขียนโดยภาษาจาวา และโปรแกรมที่เขียนโดยภาษาไพทอน หรือการแลกเปลี่ยนข้อมูลระหว่างโปรแกรมประยุกต์ที่ทำงานบนไมโครซอฟท์วินโดวส์และโปรแกรมประยุกต์ที่ทำงานบนลินุกซ์) เกิดขึ้นได้เนื่องจากการใช้มาตรฐานเปิด โดย OASIS และ W3C เป็นคณะกรรมการหลักในการรับผิดชอบมาตรฐานและสถาปัตยกรรมของเว็บเซอร์วิส

* 1. มาตรฐานหลักของเว็บเซอร์วิส

มาตรฐานหลักของการพัฒนาเว็บเซอร์วิสจะประกอบไปด้วยมาตรฐานต่างๆดังนี้ XML WSDL SOAP และ UDDI รายละเอียดของแต่ละมาตรฐานมีดังนี้

1. Extensible Markup Language (XML)
2. Simple Object Access Protocol (SOAP)
3. Web Services Description Language (WSDL)
4. Universal Description, Discovery and Integration (UDDI)
   1. คุณลักษณะพื้นฐานของเว็บเซอร์วิส
5. เว็บเซอร์วิสเป็นซอฟต์แวร์คอมโพเนนท์ที่ระบุตำแหน่งโดยใช้ URI
6. อินเตอร์เฟสและการติดตั้งของเซอร์วิสจะนิยาม อธิบาย และค้นหาโดยใช้ ภาษาXML
7. เว็บเซอร์วิสสนับสนุนการเรียกใช้จากซอฟต์แวร์ประยุกต์อื่นๆผ่านอินเตอร์เนต
8. เว็บเซอร์วิสใช้เอกสารแบบ XML ในการส่งข้อมูลระหว่างผู้ให้บริการและผู้ใช้
9. เว็บเซอร์วิสช่วยในการเชื่อมโยงโปรแกรมประยุกต์ต่างแพลตฟอร์ม (Cross-platform Integration) ผ่านอินเตอร์เน็ต
10. นักพัฒนาสามารถพัฒนาเว็บเซอร์วิสได้โดยใช้โปรแกรมภาษาคอมพิวเตอร์ต่างๆเช่นJava, C, C# หรือ Visual Basic และสามารถพัฒนาโดยการแปลงซอฟต์แวร์คอมโพเนนท์ที่มีอยู่ให้เป็นเว็บเซอร์วิส
11. เว็บเซอร์วิสจะไม่รวมถึงการจัดการส่วนแสดงผลเหมือน HTML
12. เว็บเซอร์วิสจะเป็นซอฟต์แวร์คอมโพเนนท์แบบ loosely couple ดังนั้นแต่ละคอมโพเนนท์จะเป็นอิสระและมีฟังก์ชันที่สมบูรณ์ในตัว
13. เราสามารถที่จะค้นหาและเรียกใช้เว็บเซอร์วิสจาก registry ที่เป็นแบบ public หรือ private โดยใช้มาตรฐานกลางเช่น UDDI และ ebXML
14. Extensible Markup Language (XML)

เอกซ์เอ็มแอล [] เป็นภาษามาร์กอัพสำหรับการใช้งานทั่วไป พัฒนาโดยองค์กร W3C โดยมีจุดประสงค์เพื่อเป็นสื่อกลาง ที่เอาไว้ช่วยในการติดต่อแลกเปลี่ยนข้อมูลในระบบที่มีความแตกต่างกัน ( เช่นใช้คอมพิวเตอร์มี่มีระบบปฏิบัติการคนละตัว หรืออาจจะเป็นคนละโปรแกรมประยุกต์ที่พัฒนาด้วยภาษาโปรแกรมมิ่งต่างกัน แต่มีความต้องการที่จะสื่อสารข้อมูลถึงกัน) นอกจากนี้ยังเพื่อเป็นพื้นฐานในการสร้างภาษามาร์กอัพเฉพาะทางอีกขั้นหนึ่ง XML พัฒนามาจาก SGML โดยดัดแปลงให้มีความซับซ้อนลดน้อยลง XML ใช้ในแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่แตกต่างกัน และเน้นการแลกเปลี่ยนข้อมูลผ่านอินเทอร์เน็ต

XML ยังเป็นภาษาพื้นฐานให้กับภาษาอื่นๆ อีกด้วย (ยกตัวอย่างเช่น Geography Markup Language (GML), RDF/XML, RSS, MathML, Physical Markup Language (PML), XHTML, SVG, MusicXML และ cXML) ซึ่งอนุญาตให้โปรแกรมแก้ไขและทำงานกับเอกสารโดยไม่จำเป็นต้องมีความรู้ในภาษานั้นมาก่อน.

* 1. โครงสร้างของภาษา XML

เอกสารที่สร้างขึ้นด้วยภาษา XML จะประกอบไปด้วย node ต่างๆมาประกอบกันแต่ ทุกเอกสารจะขึ้นต้นด้วย node ในการประกาศ XML ดังรูปที่ 2.24



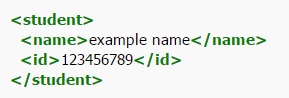
1. Declaration node ของ XML

ซึ่ง Node แต่ละ Node นั้นจะเรียกว่า Element ซึ่งประกอบไปด้วย tag เปิด ข้อมูลภายใน และ tag ปิดดังรูปที่ 2.25



1. Element ของ XML

ในที่นี้ <student> คือ tag เปิด Example\_sudent คือข้อมูล และ </student> คือ tag ปิด โดย tag ปิดนั้นจะต้องมีชื่อเหมือน tag เปิดแต่มีเครื่องหมาย '/' อยู่ด้านหน้า. ซึ่งจะสังเกตได้ว่า XML นั้นคล้ายกับ HTML เป็นอย่างมากแต่สำหรับข้อแตกต่างที่ชัดเจนคือ HTML ได้กำหนด tag ที่ใช้ได้เอาไว้ล่วงหน้าแล้วแต่ XML สามารถกำหนดหรือ define tag ขึ้นมาเองได้. XML นั้นไม่ใช่ภาษาโดยสมบูรณ์แต่เป็นเสมือนมาตรฐานข้อมูล. โดยโปรแกรมประยุกต์จะเป็นผู้กำหนดรูปแบบของตัวเองขึ้นและจะสามารถใช้ได้กับโครงสร้างข้อมูลที่ถูกอนุญาต (เพราะว่ามีรูปแบบของข้อมูลที่เข้ากันได้) XML นั้นเป็นภาษาที่ case sensitive ดังนั้นการที่เราเขียนว่า <student> กับ <Student> จึงถือว่าเป็นคนละ tag กัน นอกจากนี้แล้ว element ใน XML สามารถบรรจุอยู่ใน element อื่นๆได้ยกตัวอย่างเช่น



1. Element ภายใน element ของ XML

โดยภายใน tag Element ของ XML นั้นสามารถมีสิ่งที่เรียกว่า Attribute อยู่ภายใน tag ได้โดยตัวอย่างดังรูปที่ 2.27

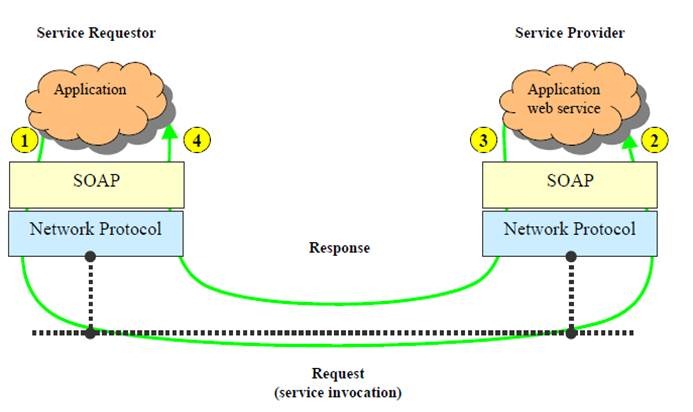


1. Attribute ของ XML
   1. การตรวจสอบความถูกต้องของ XML

สำหรับการตรวจสอบความถูกต้องของเอกสาร XML นั้นแบ่งออกเป็น 2 ระดับ คือ

1. Well-formed เอกสารที่ well-formed คือใช้ syntax ของ XML ถูกต้องตามมาตรฐานทุกอย่างมีการเปิด-ปิด tag ที่สมบูรณ์ แต่ไม่จำเป็นต้องจัดรูปแบบให้สวยงาม เอกสารที่ไม่ well-formed ถือว่าไม่เป็น XML
2. Valid นอกจาก well-formed แล้ว เอกสารที่ valid ยังต้องใช้ tag XML ที่กำหนดเฉพาะใน schema ที่ตกลงกันไว้เท่านั้น ปัจจุบันมี schema ที่นิยม 3 ตัว คือ Document Type Definition (DTD), XML Schema (W3C) (WXS) และ RELAX NG
3. Simple Object Access Protocol (SOAP)

SOAP [] ย่อมาจาก Simple Object Access Protocol คือโปรโตคอลมาตรฐานที่ใช้ใน Web Services เป็นโพรโทคอล (Protocol) ในการติดต่อแลกเปลี่ยนข้อมูลระหว่างกันของ web services เป็นการสื่อสาร ในระดับ Application Layer โดยอาศัยการส่งข้อมูลผ่านอินเทอร์เน็ต protocol ซึ่งอาศัยรูปแบบของภาษา XML ทำให้ Web services สามารถสื่อสารกันได้แม้ว่า จะอยู่บนเครื่องคอมพิวเตอร์คนละแพลตฟอร์ม หรือพัฒนาด้วยภาษาโปรแกรมที่ต่างกันก็ตาม และนิยมใช้ HTTP เป็นโปรโตคอลร่วมสำหรับส่งผ่านข้อมูลบนระบบอินเตอร์เน็ต. โดย SOAP นั้นสามารถทำงานร่วมกับโพรโตคอลได้หลายชนิด เช่น HTTP, SMTP, FTP, IIOP เป็นต้น. โดยลักษณะข้อความที่รับ - ส่ง ผ่านโพรโตคอล SOAP จะเป็นไปตามรูปแบบของ XML โดยจะเรียกว่า SOAP Message.



1. แผนผังการทำงานของ SOAP

โดยการทำงานของ SOAP จะเป็นไปดังภาพที่ 2.xx ซึ่งอธิบายการทำงานได้ดังนี้

* 1. การทำงานของ SOAP protocol

1. ผู้ขอใช้บริการ (Service Requester ) สร้าง SOAP Message เพื่อเรียกใช้บริการของ เว็บเซอร์วิส แล้วส่งผ่านโพรโตคอลเครือข่ายไปยังผู้ให้บริการ ในที่นี้ SOAP message ที่รับ-ส่งไปมานั้น อยู่ในรูปแบบ XML และต้องมีการแปลกลับมาอยู่ในรูปแบบที่โปรแกรมหรือเว็บเซิร์ฟเวอร์เข้าใจ โดยมีโปรแกรมที่ทำหน้าที่แปลความหมายของเอกสาร XML คือ XML Parser
2. ผู้ให้บริการ ( Service Provider ) ได้รับ SOAP Message จากผู้ขอใช้บริการ จากนั้น จึงแปลข้อความนั้นกลับมาอยู่ในรูปแบบที่เว็บเซิร์ฟเวอร์เข้าใจ แล้วตรวจสอบว่า ผู้ใช้บริการต้องการเรียกใช้ เว็บเซอร์วิส ชื่ออะไร เมธอดอะไร และส่งพารามิเตอร์อะไร มาด้วย จากนั้นจึงส่งไปให้แก่คอมโพเนนต์ที่ให้บริการ เว็บเซอร์วิส นั้นๆดำเนินการประมวลผล
3. หลังจากคอมโพเนนต์ที่ให้บริการ เว็บเซอร์วิส ส่งผลลัพธ์กลับมาแล้วผู้ให้บริการก็จะสร้าง SOAP Message ที่มีผลลัพธ์นั้นออกมาด้วย แล้วจึงส่งผ่านทางโพรโตคอลเครือข่ายกลับคืนไปยังผู้ขอใช้บริการ
4. ผู้ขอใช้บริการได้รับ SOAP Message ที่อยู่ในรูปแบบ XML จึงแปลข้อความนั้นกลับมาในรูปแบบที่โปรแกรมของผู้ขอใช้บริการเข้าใจแล้วนำผลลัพธ์ไปใช้งาน เช่น แสดงผล หรือไปทำอย่างอื่น แล้วแต่ว่ามีการเขียนโปรแกรมรองรับไว้ให้ทำอย่างไร
5. และจะมี SOAP Listener ทำหน้าที่คอยรับฟังว่ามีการเรียกใช้ เว็บเซอร์วิส จากผู้ใช้ การบริการของ เว็บเซอร์วิส แต่ละบริการจะมีไฟล์ SOAP Listener จำนวน 1 ไฟล์ เมื่อใดที่มีการเรียกใช้ เว็บเซอร์วิส ไฟล์โปรแกรมที่เป็น SOAP Listener ก็จะไปปลุกให้ เว็บเซอร์วิสทำงาน
   1. ข้อดีของการใช้ protocol SOAP
6. โพรโตคอล SOAP สามารถให้เราเรียกใช้คอมโพเนนต์ หรือ เว็บเซอร์วิส ข้ามเครื่อง ข้าม แพลตฟอร์มหรือข้ามภาษา ได้ โดยอาศัยโพรโตคอลที่มีอยู่เดิมในอินเทอร์เน็ต อย่าง HTTP
7. โครงสร้างข้อมูลของ SOAP เป็นรูปแบบข้อความที่สื่อสารกันด้วยภาษา XML ซึ่งมีลักษณะเป็นข้อความธรรมดาๆปิดล้อมด้วย tag ทำให้เข้าใจได้ในทุกแพลตฟอร์ม
8. โพรโตคอล SOAP สามารถทำงานผ่านระบบ Firewall ได้ง่ายเนื่องจาก SOAP ทำงานอยู่กับ โพรโตคอล HTTP ซึ่งโดยธรรมชาติของ Firewall จะเปิดให้การสื่อสารด้วย HTTP ผ่านได้อย่างสะดวก
9. SOAP สนับสนุนจากหลายค่าย เช่น IBM, MS , SUN
   1. ข้อเสียของการใช้ protocol SOAP
10. เนื่องจากลักษณะของ SOAP message เป็นเอกสาร XML ทำให้เสียเวลาในการแปลกลับมาเป็นรูปแบบที่โปรแกรมเข้าใจ
11. ในกรณีที่ SOAP ทำงานอยู่กับโพรโตคอล HTTP ซึ่งมีสมรรถนะในการรับ-ส่งข้อมูลต่ำกว่าโพรโตคอล DCOM, RMI, หรือ IIOP จึงทำให้โพรโตคอล SOAP มีอัตราการรับ-ส่งข้อมูลต่ำ
12. Representational state transfer (REST)

Representational state transfer หรือ REST [] นั้นถูกพูดถึงครั้งแรกในปี 2000 โดย Roy Thomas Fielding ซึ่ง Representational State Transfer (REST) นั้นเป็น Architecture (สถาปัตยกรรมการสื่อสารข้อมูล) รูปแบบหนึ่งที่ใช้ในการติดต่อสื่อสารข้อมูล เป็นแนวทางใหม่ในการสร้าง Web Service แบบเรียบง่าย โดยเรียกใช้งานผ่านทาง HTTP Method GET / POST / PUT / DELETE และส่งข้อมูลออกมาในรูปของ XML JSON HTML หรือ Text. โดยแนวคิดของ REST [] จะทำการมองเว็บให้เป็นทรัพยากร แล้วทำการจัดเตรียมรูปแบบที่เอาไว้สำหรับติดต่อ เพื่อติดต่อกับทรัพยากรเหล่านี้ โดยการเรียก REST แบบทั่วไปมักจะส่งคำร้องด้วยค่า input มากับ URL เช่น<http://ww.google.com/search?q=currency+exchange> เป็นการส่งคำร้องเพื่อเรียกใช้ REST ที่ทาง google เตรียมเอาไว้ให้ โดยส่งค่า input เข้าไป โดยผลลัพธ์ที่ได้กลับมาจะอยู่ในรูป XML หรือ JSON

โดย Web Service ที่พัฒนาขึ้นด้วยหลักการ REST จะเรียกว่า RESTFul Web Service หรืออีกชื่อคือ RESTFul Web API โดยอาศัย HTTP Method ในการกำหนดคำสั่งต่างๆ โดยตัวอย่าง หน้าที่ของแต่ละ HTTP Method ที่ REST ใช้นั้น จะแสดงดังรูปที่ 2.29



1. RESTful Web Service HTTP Method
2. JavaScript Object Notation (JSON)

เจซัน (JSON) [] คือ ฟอร์แมตสำหรับแลกเปลี่ยนข้อมูลคอมพิวเตอร์ ฟอร์แมต JSON นั้นอยู่ในรูปข้อความธรรมดา (plain text) ที่ทั้งมนุษย์และโปรแกรมคอมพิวเตอร์สามารถอ่านเข้าใจได้มาตรฐานของฟอร์แมต JSON คือ RFC 4627 มี Internet media type เป็น application/json และมีนามสกุลของไฟล์เป็น .json ปัจจุบัน JSON นิยมใช้ในเว็บแอปพลิเคชัน โดยเฉพาะ AJAX โดย JSON เป็นฟอร์แมตทางเลือกในการส่งข้อมูล นอกเหนือไปจาก XML ซึ่งนิยมใช้กันอยู่แต่เดิม สาเหตุที่ JSON เริ่มได้รับความนิยมเป็นเพราะกระชับและเข้าใจง่ายกว่า XML โดย โครงสร้างของ JSON นั้นจะแสดงดังรูปที่ 2.30



1. ตัวอย่างไฟล์ JSON

โดย JSON นั้นไม่ถูกมองว่าเป็นภาษาโปรแกรม กลับถูกมองว่าเป็นภาษาในการแลกเปลี่ยนข้อมูลมากกว่า ในปัจจุบันมีไลบรารีของภาษาโปรแกรมอื่นๆ ที่ใช้ประมวลผลข้อมูลในรูปแบบ JSON มากมาย เนื่องจากมีความสัมพันธ์ของข้อมูลเป็นแบบ key-value ซึ่งทุกภาษาสามารถนำไปเชื่อมโยงกับประเภทข้อมูลของตัวเองได้ โดยในปัจจุบัน JSON นิยมนำไปใช้ในการส่ง-รับข้อมูลผ่าน RESTFul Web Service

1. HyperText Transfer Protocol (HTTP)

HTTP (HyperText transfer protocol) คือโพรโทคอลตัวหนึ่งในชั้นของโปรแกรมประยุกต์ หรือแอพพลิเคชั่นเลเยอร์ ในแบบจำลองการสื่อสารข้อมูล 7 ชั้น ที่ทำหน้าที่ในการ รับ-ส่ง แจกจ่าย และติดต่อสื่อสารข้อมูลสื่อผสมต่างๆ เช่น หน้า HTML, ไฟล์ภาพ, ไฟล์ข้อความ text และอื่นๆ

การพัฒนาเอชทีทีพีเป็นการทำงานร่วมกันของเวิลด์ไวด์เว็บคอนซอร์เทียม (W3C) และคณะทำงานเฉพาะกิจด้านวิศวกรรมอินเทอร์เน็ต (IETF) ซึ่งมีผลงานเด่นในการเผยแพร่เอกสารขอความเห็น (RFC) หลายชุด เอกสารที่สำคัญที่สุดคือ RFC 2616 (เดือนมิถุนายน พ.ศ. 2542) ได้กำหนด HTTP/1.1 ซึ่งเป็นรุ่นที่ใช้กันอย่างกว้างขวางในปัจจุบัน อชทีทีพีเป็นมาตรฐานในการร้องขอและการตอบรับระหว่างเครื่องลูกข่ายกับเครื่องแม่ข่าย ซึ่งเครื่องลูกข่ายคือผู้ใช้ปลายทาง (end-user) และเครื่องแม่ข่ายคือเว็บไซต์ เครื่องลูกข่ายจะสร้างการร้องขอเอชทีทีพีผ่านทางเว็บเบราว์เซอร์ เว็บครอว์เลอร์ หรือเครื่องมืออื่น ๆ ที่จัดว่าเป็น ตัวแทนผู้ใช้ (user agent) ส่วนเครื่องแม่ข่ายที่ตอบรับ ซึ่งเก็บบันทึกหรือสร้าง ทรัพยากร (resource) อย่างเช่นไฟล์เอชทีเอ็มแอลหรือรูปภาพ จะเรียกว่า เครื่องให้บริการต้นทาง (origin server) ในระหว่างตัวแทนผู้ใช้กับเครื่องให้บริการต้นทางอาจมีสื่อกลางหลายชนิด อาทิพร็อกซี เกตเวย์ และทุนเนล เอชทีทีพีไม่ได้จำกัดว่าจะต้องใช้ชุดเกณฑ์วิธีอินเทอร์เน็ต (TCP/IP) เท่านั้น แม้ว่าจะเป็นการใช้งานที่นิยมมากที่สุดบนอินเทอร์เน็ตก็ตาม โดยแท้จริงแล้วเอชทีทีพีสามารถ "นำไปใช้ได้บนโพรโทคอลอินเทอร์เน็ตอื่น ๆ หรือบนเครือข่ายอื่นก็ได้" เอชทีทีพีคาดหวังเพียงแค่การสื่อสารที่เชื่อถือได้ นั่นคือโพรโทคอลที่มีการรับรองเช่นนั้นก็สามารถใช้งานได้

โดยปกติเครื่องลูกข่ายเอชทีทีพีจะเป็นผู้เริ่มสร้างการร้องขอก่อน โดยเปิดการเชื่อมต่อด้วยเกณฑ์วิธีควบคุมการขนส่งข้อมูล (TCP) ไปยังพอร์ตเฉพาะของเครื่องแม่ข่าย (พอร์ต 80 เป็นค่าปริยาย) เครื่องแม่ข่ายเอชทีทีพีที่เปิดรอรับอยู่ที่พอร์ตนั้น จะเปิดรอให้เครื่องลูกข่ายส่งข้อความร้องขอเข้ามา เมื่อได้รับการร้องขอแล้ว เครื่องแม่ข่ายจะตอบรับด้วยข้อความสถานะอันหนึ่ง ตัวอย่างเช่น "HTTP/1.1 200 OK" ตามด้วยเนื้อหาของมันเองส่งไปด้วย เนื้อหานั้นอาจเป็นแฟ้มข้อมูลที่ร้องขอ ข้อความแสดงข้อผิดพลาด หรือข้อมูลอย่างอื่นเป็นต้น

ทรัพยากรที่ถูกเข้าถึงด้วยเอชทีทีพีจะถูกระบุโดยใช้ตัวระบุแหล่งทรัพยากรสากล (URI) (หรือเจาะจงลงไปก็คือ ตัวชี้แหล่งในอินเทอร์เน็ต (URL)) โดยใช้ http: หรือ https: เป็นแผนของตัวระบุ (URI scheme)

* 1. ข้อความร้องขอ (HTTP request)

ข้อความร้องขอประกอบด้วยสิ่งต่อไปนี้

1. บรรทัดแรก ขึ้นต้นเป็นคำสั่งร้องขอ และเส้นทางไดเรกทอรีของแฟ้มที่ร้องขอ ตามด้วยรุ่นของ HTTP ตัวอย่างเช่น GET /images/logo.gif HTTP/1.1
2. บรรทัดต่อ ไปที่ไม่ใช่บรรทัดว่าง เรียกว่า ส่วนหัว (header) เป็นเมทาเดตา(metadata) ต่าง ๆ ประกอบการร้องขอ ตัวอย่างเช่น Accept-Language: en หรือ Content-type : application/json เป็นต้น
3. บรรทัดว่าง เพื่อแบ่งแยกระหว่างส่วนหัวกับเนื้อหา
4. บรรทัดต่อๆ ไป เป็นเนื้อหาข้อมูล ซึ่งบางคำสั่งอาจไม่จำเป็นต้องใช้ส่วนนี้

แต่ละบรรทัดจะต้องลงท้ายด้วย CRLF (อักขระปัดแคร่ตามด้วยอักขระป้อนบรรทัด เหมือนการกดปุ่ม Enter ในวินโดวส์) บรรทัดที่ว่างจะมีเพียงแค่ CRLF เท่านั้นโดยไม่มีอักขระช่องว่างอยู่เลย สำหรับรุ่น HTTP/1.1 ส่วนหัว Host: จำเป็นต้องมีเสมอ แต่ส่วนหัวอื่น ๆ ไม่จำเป็นต้องมีก็ได้ บรรทัดคำสั่งที่มีเพียงเส้นทางไดเรกทอรี (ไม่มีชื่อแฟ้ม) ก็เป็นที่ยอมรับโดยเครื่องแม่ข่าย เพื่อรักษาความเข้ากันได้กับโปรแกรมตัวแทนรุ่นเก่าก่อนที่จะมีข้อกำหนดของ HTTP/1.0 ใน RFC 1945 ส่วน HTTP/1.1 ได้กำหนดไว้ใน RFC 2068

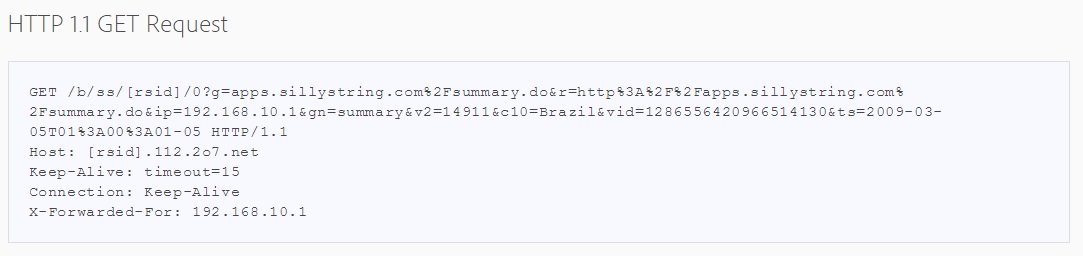
* 1. คำสั่งร้องขอ (HTTP method)

เอชทีทีพีได้กำหนดคำสั่งร้องขอไว้แปดคำสั่ง (หรือเรียกว่าวิธีการร้องขอ บางครั้งอาจเรียกว่าเป็น "กริยา") แสดงการกระทำที่ต้องการ เพื่อที่จะดำเนินการกับทรัพยากรที่ถูกระบุ สิ่งที่ทรัพยากรนั้นนำเสนอ ไม่ว่าเป็นข้อมูลที่มีอยู่ก่อนหรือสร้างขึ้นมาแบบพลวัตก็ตาม จะขึ้นอยู่กับการนำไปใช้ของเครื่องแม่ข่าย ซึ่งบ่อยครั้งทรัพยากรมักจะสอดคล้องกับไฟล์ หรือผลลัพธ์ส่งออกจากโปรแกรมข้างเคียงในเครื่องแม่ข่ายนั้น เครื่องให้บริการเอชทีทีพีจะต้องสามารถใช้คำสั่ง GET และ HEAD ได้เป็นอย่างน้อย โดยประเภทของคำสั่งร้องขอจะแบ่งออกเป็น 8 ประเภท [] ดังนี้

1. OPTIONS เป็นเมทอดของ HTTP ที่มีหน้าที่ในการแสดงถึงเมทอดอื่นๆที่เซอร์วิสของเว็บไซต์ที่เราทำการร้องขอมีไวให้ใช้บริการ โดยปกติแล้วคำร้องของเมทอด OPTIONS จะไม่มีเนื้อหาหรือ entity body เนื่องเจอเป็นเม-ทอดที่เอาไว้ให้บริการเพียงเพื่อบอกถึงเมทอดที่มีให้บริการ โดยข้อความตอบกลับนั้นควรจะมีรหัส 200 และพ่วงมาด้วย header ที่ชื่อว่า Allow : แล้วตามด้วย เมทอดที่เซอร์วิสมีให้บริการ โดยตัวอย่างของข้อความตอบกลับจากเมทอด OPTIONS จะมีลักษณะดังรูปที่ 2.31

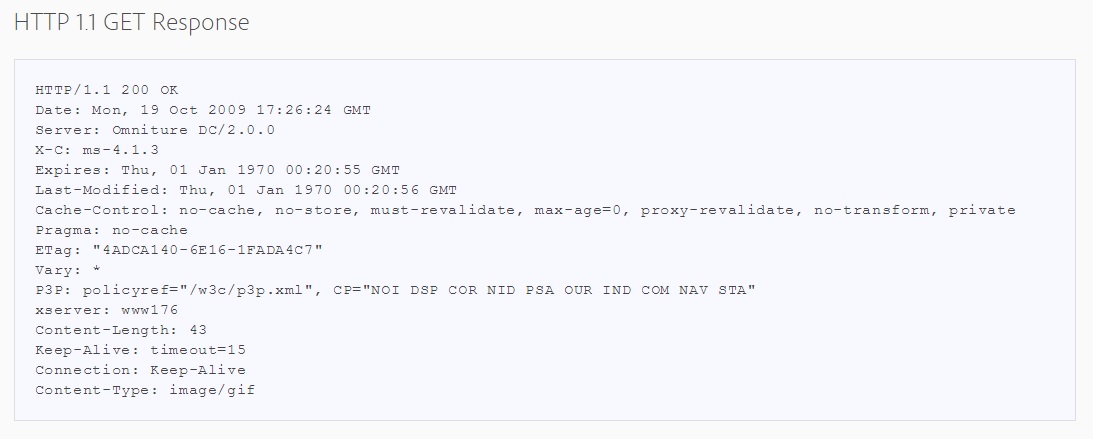


1. ข้อความตอบกลับของเมทอด OPTIONS
2. GET เป็นเมทอดที่หมายถึงการร้องขอข้อมูลที่ URI ปลายทางได้เตรียมเอาไว้ให้ ซึ่งถ้าปลายทางเป็นโปรเซสในการสร้างข้อมูล สิ่งที่จะได้รับกลับมาในข้อความตอบกลับจากการส่งเมทอด GET ก็คือ ข้อมูลที่ถูกสร้างโดยโปรเซสนั้น ซึ่งสิ่งที่ได้รับอาจจะเป็นหน้าเพจ HTML, ไฟล์รูปภาพหรือ อื่นๆก็เป็นได้ ซึ่งเมทอด GET นั้นมีหน้าที่ในการร้องขอข้อมูลเท่านั้น จะไม่มีนิยมให้มีการลบข้อมูล แก้ไขหรือ เพิ่มข้อมูลผ่านเมทอด GET โดยข้อมูลที่ได้รับจะอยู่ในส่วน body ของข้อความตอบกลับ ส่วนข้อมูลที่จะส่งไปกับเมทอด GET นิยมใช้เป็น URI พารามิเตอร์ซึ่งติดไปกับคำร้องขอ ดังรูปที่ 2.32 โดยพารามิเตอร์จะอยู่หลังเครื่องหมาย ? ในท่อนสุดท้ายของ URL ซึ่งหากมีพารามิเตอร์หลายตัวจะขั้นด้วยเครื่องหมาย & สำหรับพารามิเตอร์แต่ละตัว



1. ตัวอย่างคำร้องเมทอด GET

ส่วนข้อความตอบกลับจะมีลักษณะดังรูปที่ 2.33 ซึ่งเราสามารถดูประเภทของไฟล์ที่ได้รับกลับมาจาก header ชื่อว่า Content-type



1. ตัวอย่างข้อความตอบกลับของเมทอด GET
2. HEAD สำหรับเมทอด HEAD นั้นเป็นเมทอดที่มีหน้าการทำงานเหมือนเมทอด GET ทุกประการ แต่ส่วนที่แตกต่างกันก็คือ หากได้รับคำสั่ง HEAD ทางฝั่งเซิฟเวอร์จะต้องไม่ทำการส่งส่วน body กลับมาในข้อความตอบกลับ ซึ่งข้อมูลที่จะได้รับกลับมานั้นจะอยู่ในส่วนของ header ดังนั้น ส่วน header ของเมทอด GET และ HEAD จะต้องเหมือนกันทุกประการ เพียงแต่ว่าเมทอด HEAD จะมีเพียงแค่ header ที่บ่งบอกถึงลักษณะของข้อมูลที่จะได้รับ แต่ไม่ได้รับในส่วนของ body กลับมา
3. POST เป็นเมทอดที่ถูกใช้ในการส่งข้อมูลไปกับ body ของคำร้องไปให้กับเซิฟเวอร์เพื่อจุดประสงค์ในการทำการบันทึกข้อมูลนั้นเข้าสู่ระบบ ซึ่งโดยทั่วไปแล้ว เมทอด POST จะถูกใช้มากในการอัพโหลดไฟล์ หรือทำการส่งข้อมูลที่ผู้ใช้ทำการกรอกข้อมูลในฟอร์มต่างๆบนเว็บ และส่งให้เซิฟเวอร์ ซึ่งต่างกับเมทอด GET ตรงที่ เมทอด GET เป็นเมทอดที่ใช้ในการร้องขอข้อมูลจากทางฝั่งเซิฟเวอร์และสามารถส่งข้อมูลบางอย่างให้กับเซิฟเวอร์ไปพร้อมกับ url ที่เป็นข้อความเท่านั้น ส่วน POST จะสามารถส่งข้อมูลใดๆก็ได้ไปพร้อมกับ body ซึ่งโดยส่วนมากแล้วนั้นส่วน header ของเมทอด POST จะทำหน้าที่ในการกำหนด media type หรือประเภทของข้อมูลที่ผู้ใช้ทำการส่งไปพร้อมกับคำร้องขอ โดยข้อความตอบกลับจากเมทอด POST จะเน้นแสดงถึงสถานะว่า การบันทึกข้อมูลมีผลอย่างไร
4. PUT
5. DELETE
6. TRACE
7. CONNECT
8. การพัฒนาโปรแกรมแบบบูรณาการอย่างต่อเนื่อง (Continuous Integration)

Continuous Integration [1] หรือเทคนิคการพัฒนาโปรแกรมแบบบูรณาการอย่างต่อเนื่อง คือ เทคนิคที่ใช้ในการพัฒนาโปรแกรมที่มีหลักการคือ ให้นักพัฒนาทุกคนในทีมพัฒนาที่ทำงานกับ ซอร์สโค้ดชุดเดียวกันนั้น ทำการรวมซอร์สโค้ดที่ตนเองเขียนขึ้นเข้าสู่ที่เก็บซอร์สโค้ดส่วนกลางหรือ central repository หลายๆครั้งในหนึ่งวัน เพื่อให้ทีมพัฒนาสามารถตรวจพบปัญหาที่เกิดขึ้นได้อย่างรวดเร็ว และต่อเนื่อง ยิ่งการรวมเข้าสู่ซอร์สโค้ดกลางบ่อยเท่าไหร่ ก็จะยิ่งพบปัญหาได้เร็ว และง่ายมากยิ่งขึ้น

* 1. คุณสมบัติของ Continuous Integration

เนื่องจากว่าเราทำการรวมซอร์สโค้ดเข้าสู่ส่วนกลางอย่างต่อเนื่อง ทำให้เราสามารถรู้ได้ว่า ข้อผิดพลาดเกิดขึ้น ณ จุดใด ตอนไหน ซึ่งการทำการบูรณาการอย่างต่อเนื่องนี้ใช้ทรัพยากรน้อย ซึ่งถ้าเราไม่ทำการรวมซอร์สโค้ดบ่อยๆ จะทำให้การตามหาข้อผิดพลาดที่เกิดขึ้นนั้นทำได้ยาก เนื่องจากไม่ทราบว่าข้อผิดพลาดเกิดจากใคร เกิดขึ้นจากอะไร และเมื่อไหร่ ซึ่งเหตุการณ์แบบนี้อาจทำให้โปรเจคที่ทำเกิดการล่าช้าได้ เนื่องจากทุกคนทำงานร่วมกัน

* 1. แนวทางปฏิบัติของ Continuous Integration []

1. ทุกๆ อย่างที่จำเป็นต่อการ build ระบบ ควรอยู่ใน repository เดียวกัน โดยขณะเดียวกันเราต้องสามารถทำการ build บนเครื่องพัฒนาของตนเองได้เลย ส่วน repository ส่วนกลางนั้นให้ใช้ที่เดียวกันเป็น main branch ในการพัฒนาระบบเพียง branch เดียวใช้ร่วมกันในทีม เนื่องจากยิ่งมีมาก ยิ่งเกิดปัญหามาก ดังนั้นถ้ามี branch หลักในการทำงานเพียง branch เดียวแล้วจะทำให้มั่นใจได้ว่า ระบบจะยังคงทำงานได้อยู่เสมอ แล้วหากต้องการอาจจะมี branch อื่นๆ ขึ้นมาเช่น fix bug branch สำหรับการแก้ไขข้อผิดพลาดบางอย่าง
2. การ deploy หรือ run ระบบนั้นมักจะมีกระบวนการที่ซับซ้อนเช่นการ compile การ copy หรือย้ายไฟล์ การสร้างหรือแก้ไข schema databaseรวมทั้งอาจจะมีกระบวนการอื่นๆ อีกมากมายส่วนใหญ่มักจะทำกระบวนการดังกล่าวแบบ manualซึ่งสูญเสียเวลาอันมีค่าไปอย่างมาก ดังนั้นควรเปลี่ยนกระบวนการเหล่านี้ให้ไปเป็นแบบอัตโนมัติหรือ automated ให้หมด โดยในการ build สามารถใช้เครื่องมือที่เรียกว่า build tool มาช่วยได้ เช่น ในภาษา Java สามารถใช้พวก Maven, Gradle มาช่วย ซึ่งสั่งให้ build ได้ด้วยชุดคำสั่งเดียวโดยการเขียนเอาไว้ใน build script ซึ่งจะช่วยให้การ build ระบบทำได้ง่ายมาก และนอกจากนั้น automated build ที่ดีจะต้องสามารถที่จะ build และ run บนเครื่องของผู้พัฒนาเองได้เลย และหากระบบที่พัฒนามีขนาดใหญ่มาก การ build แบบนี้ทุกๆ ครั้งที่มีการเปลี่ยนแปลง อาจจะใช้เวลามากเกินไปวิธีการแก้ไขก็คือให้ทำการ build เฉพาะส่วนที่แก้ไข ซึ่งอาจจะตั้งค่าเอาไว้ใน build server เพื่อให้ทุกคนใช้งานร่วมกัน
3. ในเครื่องพัฒนาจะต้องมีชุดของการทดสอบที่ทำงานแบบอัตโนมัติเพื่อใช้ตรวจสอบ เพื่อหาข้อผิดพลาดต่างๆของระบบ สามารถตรวจสอบด้วยชุดคำสั่งที่สั้นและง่าย และผลการทดสอบนั้น ควรที่จะต้องบอกถึงผลการทดสอบด้วยว่าผ่านและไม่ผ่านเพราะว่าอะไร ดังนั้น จึงควรจะพัฒนาระบบตามแนวคิดของ TDD (Test-Driven Developement) ซึ่งจะช่วยให้เรามีเทสเอาไว้ทดสอบระบบแบบอัตโนมัติในทุกๆจุด ช่วยทำให้ระบบมีความถูกต้องและ ระบบสามารถทดสอบได้แบบอัตโนมัติ
4. ให้ทำการ commit และ push ซอร์สโค้ดไปที่ repository ที่เก็บส่วนกลางเสมอ และบ่อยๆ เพื่อลดเวลาในการ merge รวมซอร์สโค้ดของผู้พัฒนาที่ทำงานกับซอร์สโค้ดชุดเดียวกัน ซึ่งการ commit บ่อยๆจะทำให้แก้ปัญหาได้อย่างรวดเร็ว
5. ทุกๆครั้งที่ repository ส่วนกลางถูกอัพเดท ระบบจะต้องถูก build ใหม่เสมอเพื่อให้รู้ถึงสถานภาพของระบบ ณ ปัจจุบันว่า fail อยู่หรือผ่านอยู่ เพื่อให้เห็นข้อผิดพลาดที่เกิดขึ้นจากแต่ละ commit เพื่อให้เราสามารถปรับปรุงได้ดียิ่งขึ้น ซึ่งทุกๆครั้งก่อนที่นักพัฒนาจะทำการ commit ซอร์สโค้ดของตนเองเข้าสู่ repository ส่วนกลางควรจะต้องรัน unit test ในเครื่องตนเองให้ผ่านทั้งหมดก่อน เพื่อให้เมื่อทำการรวมกับส่วนกลางจะไม่ทำให้ build ระบบ fail เพื่อเป็นการทำให้ build server เป็นการ build เพื่อรัน integration test เท่านั้น และหาก commit ของใครทำให้ระบบ build fail จะต้องทำให้ระบบกลับมา build สำเร็จเสมอก่อน ไม่เช่นนั้นจะทำให้นักพัฒนาคนอื่นไม่สามารถทราบได้ว่า ระบบที่ build fail ณ ตอนนี้เกิดขึ้นจากซอร์สโค้ดของใคร
6. ทำให้การ build integration นั้นเร็ว หากสร้างระบบ continuous integration ที่ทำงานช้าถือว่าเป็นความล้มเหลว เนื่องจากเราต้องใช้เวลามากในการรอผล build ซึ่งทำให้ไปทำงานอื่นต่อได้ช้า ซึ่งบางครั้งหากการ build ทำได้ช้า เราอาจจะต้องดูว่าหากช้าเพราะว่าการรัน test หลายๆแบบรวมกันเช่น BDD, ATDD, UI test, performance test เราควรจะต้องทำ build pipeline หรือการแยก build ออกเป็นส่วนๆ เพื่อความเร็ว ซึ่งจะช่วยให้เราสามารถสังเกตุเฉพาะ pipeline ที่เราสนใจว่า หาก pipeline ของเราผ่านก็ไปทำงานต่อได้
7. สภาพแวดล้อมในการ test นั้นจะต้องเหมือนกับบนเซิฟเวอร์จริงของลูกค้า เพื่อทำให้เรามั่นใจได้ว่าหากทำการ ส่งมอบ ย้ายไปสู่ production จะไม่เกิดปัญหาที่ไม่เคยเจอมาก่อน
8. จะต้องมีไฟล์ระบบงานล่าสุดเตรียมพร้อมเสมอ เช่น WAR, EAR ไฟล์ของระบบจาก build ล่าสุด ไม่ใช่สร้างเมื่อต้องการ
9. ทุกๆคนในทีมต้องสามารถเห็นสิ่งที่คนอื่นทำ เพื่อช่วยในการติดต่อสื่อสาร เมื่อมีปัญหา หรือต้องการจะทำอะไรกับระบบ
10. ควรจะต้องทำ automated deployment คือเมื่อทำการ build เสร็จสามารถ deploy ได้เลย
    1. ข้อดีของการทำการพัฒนาแบบบูรณาการอย่างต่อเนื่อง
11. ไม่ต้องเสียเวลากับการทำการรวมซอร์สโค้ดเอง ซึ่งใช้เวลานาน และยุ่งยาก
12. ช่วยเพิ่มความโปร่งใสในการทำงาน เนื่องจากทราบถึงการทำงานของแต่ละคนอย่างชัดเจน ว่าใครทำอะไรในส่วนไหน ดังนั้นจึงเป็นการสนับสนุนให้ทีมมีการสื่อสารกันได้อย่างดียิ่งขึ้น
13. พบเจอปัญหาได้เร็ว และช่วยให้จัดการกับปัญหานั้นได้ในทันที
14. ช่วยให้ทีมพัฒนาลดเวลาในการที่จะต้องมาทำการหาข้อผิดพลาด แก้ bug และเพิ่มเวลาในการเพิ่มคุณสมบัติอื่นๆให้กับโปรเจคแทน
15. ช่วยให้มั่นใจได้ว่า ซอร์สโค้ดส่วนกลางของโปรเจคที่มีไว้เพื่อส่งมอบให้กับลูกค้านั้นมีความถูกต้องอยู่ตลอดเวลา
16. ไม่ต้องเสียเวลาในการรอการสร้างโปรเจคและรอดูผลในเครื่องซึ่งอาจใช้เวลานาน ทำให้เสียเวลาพัฒนา
17. ช่วยให้การส่งมอบทำได้บ่อยและเร็วขึ้น
18. Jenkins

Jenkins [] คือ ซอฟต์แวร์ open-source ประเภท CI server หรือ continuous integration server ที่เป็นที่นิยมมากที่สุดตัวหนึ่ง สร้างขึ้นด้วยภาษา Java โดย Jenkins มีหน้าที่ช่วยเร่งความเร็วให้กับวงจรการพัฒนาซอฟต์แวร์ ผ่านทางวิธีการอัตโนมัติต่างๆ โดย Jenkins ช่วยในการจัดการและควบคุมวงจรการพัฒนาซอฟต์แวร์ในทุกๆขั้นตั้งแต่การสร้าง, การทำเอกสาร, การทดสอบ, การส่งมอบและ การทำสถิติและวิเคราะห์

Jenkins [] เป็นระบบที่เป็น server-based ซึ่งทำงานได้ภายใต้ servlet container เช่น Apache, Tomcat และทำงานร่วมกับ SCM tool ที่ทำหน้าที่จัดการกับซอร์สโค้ดกลาง ในการให้บริการเพื่อทำงานกับระบบ continuous integration โดยสามารถช่วยสั่ง build โปรเจคที่เป็น maven-based หรือ ant-based โปรเจคผ่านทาง build script หรือผ่านทาง batch command โดย Jenkins ถูกสร้างขึ้นโดย Kohsuke Kawaguchi และถูกปล่อยให้ใช้งานเป็น free software หรือ open-source ภายใต้ MIT License

* 1. feature ของ Jenkins []

1. Easy installation คือ การจะนำ Jenkins ไปใช้งานเพื่อทำ continuous integration นั้น วิธีการติดตั้ง Jenkins สามารถทำได้ง่ายเพียงแค่ทำการรัน java –jar กับไฟล์ jenkins.war ซึ่งเวปไซต์ของ Jenkins เองมีให้ดาวน์โหลดได้โดยตรง หรือสามารถนำไป deploy ผ่าน servlet container เช่น Tomcat หรือภายใต้ IDE ที่มี servlet container ให้บริการเช่น Eclipse spring tool suites ด้วยการนำ jenkins.war ไป deploy ได้โดยตรง ไม่ต้องมีการติดตั้งอื่นๆ หรือลงโปรแกรมฐานข้อมูลใดๆ
2. Easy configuration คือ การตั้งค่าต่างๆ ของ Jenkins เช่นการสร้าง build job, การตั้งค่า build script ของ job หรือการตั้งค่าทั่วไปอื่นๆ สามารถทำได้ง่ายมาก ผ่าน Web UI ของตัว Jenkins เองที่มีไว้ให้ ซึ่งเป็น UI ที่สวยงาม ทำความเข้าใจและใช้งานง่าย ทำให้ไม่ต้องทำการตั้งค่าต่างๆผ่าน XML หรือถ้าหากต้องการทำผ่าน XML อยู่ก็ทำได้
3. Change set support คือ Jenkins สามารถแสดงถึงเซตของ build ที่ถูกเรียกจากการ commit เข้าสู่ repository
4. Permanent links คือ Jenkins ได้ทำการเตรียม URL ต่างๆที่เข้าใจง่าย ตรงตัว สำหรับบริการต่างๆที่ใช้ได้กับ job เช่น สำหรับ job ของเราหากเราเติม /latestBuild เข้าไป Jenkins ก็จะแสดงข้อมูลของ build ล่าสุด หรือ /build ก็จะทำการสั่ง build job นั้นผ่าน REST API ได้
5. RSS/E-mail คือ feature ที่ช่วยให้ Jenkins ทำการส่ง feedback ถึงผลของ build ผ่านทาง RSS feed หรือส่ง E-mail ไปหาบัญชีที่ได้มีการลงทะเบียนเพื่อขอผลของการ build สำหรับ job นั้นเอาไว้ เพื่อให้ผู้พัฒนาหรือผู้ที่เกี่ยวข้องกับโปรเจคสามารถรู้ถึงสถานะของโปรเจค และนักพัฒนาไม่จำเป็นต้องนั่งรอเฝ้าหน้าจอเพื่อรอผลของ build
6. After-the-fact tagging ช่วยให้ทำการ tag build ในภายหลังจากที่ build สำเร็จ
7. JUnit/TestNG test reporting เป็น feature ที่ช่วยในการสรุปรายงานของการทดสอบของ Junit แสดงผลสรุป ประวัติการทดสอบ และช่วยพลอตกราฟถึงผลการทดสอบที่ผ่านมาเพื่อดูแนวโน้มของ build
8. Distributed builds คือ Jenkins สามารถกระจายงาน build ไปที่คอมพิวเตอร์หลายๆตัวได้ ทำให้ไม่จำเป็นต้อง build อยู่บนเครื่องๆเดียว
9. File fingerprinting ช่วยให้ Jenkins สามารถตามได้ว่าแต่ละ build สร้าง jars อะไร เวอร์ชั่นเท่าไหร่ และอื่นๆ
10. Plugin Support คือ feature ที่ช่วยให้ Jenkins สามารถทำงานร่วมกับ third-party อื่นๆได้ รวมถึงเราสามารถเขียน Plugin ขึ้นใช้เองสำหรับทีม และเนื่องจาก Jenkins เป็น open-source โปรเจคขนาดใหญ่ ทำให้มี community ที่คอยสร้าง Plugin ใหม่ๆอยู่ตลอดเวลา