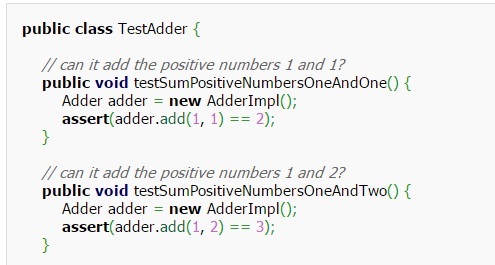
**บทที่ 2**

**ทฤษฎีที่เกี่ยวข้อง**

ในบทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับการพัฒนาโปรแกรม โดยผู้จัดทำได้ศึกษารวบรวมข้อมูลและนำความรู้เหล่านี้ไปประยุกต์ใช้ในการพัฒนาโปรแกรม ซึ่งทฤษฎีที่เกี่ยวข้องที่ได้รวบรวมไว้ มีดังนี้

**2.1 Unit Testing**

Unit Testing [1] คือ กระบวนการทดสอบซอฟต์แวร์รูปแบบหนึ่งซึ่งใช้ทดสอบซอร์สโค้ด เมท็อด หรือเซตของโมดูลในโปรแกรม ที่มีข้อมูลหรือกระบวนการทำงานที่เกี่ยวเนื่องกัน ว่าทำงานได้ถูกต้องตามหน้าที่ ที่ซอร์สโค้ดนั้นถูกเขียนขึ้นมาหรือไม่ หรือกล่าวง่ายๆก็คือ unit testing นั้นคือการทดสอบส่วนที่เล็กที่สุดเท่าที่จะทดสอบได้ในโปรแกรม ซึ่งอาจจะเป็นโมดูลการทำงานหนึ่ง หรือเพียงแค่เมท็อดๆหนึ่งก็ได้. โดยในการโปรแกรมเชิงวัตถุหรือ Object-oriented programming นั้น Unit testing มักจะทดสอบเมท็อดในคลาสหรือ ทดสอบการทำงานของทั้งคลาส. ซึ่งการทำ Unit testing นั้นจะมีการเขียน Unit test ขึ้นมาโดยโปรแกรมเมอร์ซึ่งส่วนมากมักจะเป็น White box testers. การเขียน Unit test นั้นส่วนมากจะถูกเขียนขึ้นระหว่างกระบวนการพัฒนาโปรแกรม โดย test ที่เขียนขึ้นมาเพื่อทดสอบส่วนๆหนึ่งจะเรียกว่า test case ซึ่งในแต่ละฟังก์ชั่น หรือ เมท็อดหนึ่งจะถูกทดสอบด้วย Test case เดียวหรือหลายๆ test case ก็เป็นไปได้. การเขียน test case นั้นโดยส่วนมากจะมีการสร้าง Test class ที่ประกอบไปด้วย Test case จำนวนหนึ่งเอาไว้ใน class นี้เพื่อเป็นการบ่งบอกและจัดกลุ่มให้ชัดเจนว่า class นี้เป็น class ที่เอาไว้ใช้ทดสอบฟังก์ชั่นใด ยกตัวอย่างเช่น ดังรูปที่ 2.1 คือ Test class ซึ่งในที่นี้ใช้ทดสอบการทำงานของ class ที่ชื่อ Adder ที่ใช้ในการบวกเลข



**รูปที่ 2.1** TestAdder Test class

จากรูปที่ 2.1 class TestAdder นั้นประกอบไปด้วย 2 test case คือ

1.testSumPositiveNumbersOneAndOne เป็นการทดสอบเมท็อด add ของ class Adder โดยการส่ง input 2 ตัวคือ 1 และ 1 โดยคาดหวังผลลัพธ์ว่าต้องได้เป็น 2

2.testSumPositiveNumbersOneAndTwo เป็นการทดสอบเมท็อด add ของ class Adder โดยการส่ง input 2 ตัวคือ 1 และ 2 โดยคาดหวังผลลัพธ์ว่าต้องได้เป็น 3

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

คุณสมบัติของ Unit testing [2] 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 เมื่อมีการเพิ่มเติมฟังก์ชั่นการทำงานของเมท็อดหนึ่ง แล้วต้องการทดสอบว่าเมท็อดนั้นยังทำงานของฟังก์ชั่นเดิมได้ถูกต้องหรือไม่ 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 ชื่อเมท็อดจะต้องอ่านแล้วเข้าใจ และต้องมีการ document ตัวอย่างของข้อมูลเอาไว้เสมอ เพื่อช่วยทำความเข้าใจ

ข้อดีของการทำ Unit test นั้นแบ่งออกเป็น 3 ข้อ ได้แก่

1. ช่วยให้หาข้อผิดพลาดของระบบได้รวดเร็วโดยการทำ unit testing นั้นจะช่วยแก้ปัญหาทั้งข้อผิดพลาดที่เกิดจากการเขียน code ผิดพลาดของ developer เอง และยังช่วยในการตรวจสอบความครบถ้วนว่า ซอร์สโค้ดที่เขียนขึ้นสามารถทำงานได้ตาม requirement หรือไม่
2. ช่วยในการแก้ไขปรับปรุงซอร์สโค้ด เนื่องจากการทำ unit testing เราจะมี test class ใช้ในการกำกับการทำงานของแต่ละฟังก์ชั่นในระบบ ดังนั้นเมื่อมีการปรับแต่งแก้ไขฟังก์ชั่นใดๆ developer ก็จะมี test class นั้นเป็นตัวกำกับเพื่อทำให้มั่นใจได้ว่า การปรับแต่งใดๆที่เกิดขึ้นกับฟังก์ชั่นนั้นจะไม่ไปทำให้ซอร์สโค้ดเดิมเกิดข้อผิดพลาด โดยอาศัยการตรวจสอบว่า เมื่อมีการปรับแต่งเกิดขึ้น test case เดิมทั้งหมดใน test class จะต้องรันผ่าน
3. ช่วยทำตัวเป็นเอกสาร เนื่องจากปกติแล้วการเขียน test case จะต้องมีการเขียน example หรือตัวอย่างการใช้งานเอาไว้ด้วย ดังนั้น unit test จึงช่วยทำหน้าที่เป็นเอกสาร สำหรับคนที่ต้องการจะศึกษาว่าเมท็อดหรือคลาสใดๆนั้น ทำงานอย่างไร มีหน้าที่อะไร ก็สามารถมาดูได้ที่ test class ได้

**2.2 JUnit**

Junit [3] คือ framework สำหรับช่วยในการทำ Unit testing ด้วยภาษา Java. โดย Junit นั้นถูกพัฒนาเมื่อปี 1990 โดย Erich Gamma และ Kent Beck ซึ่ง JUnit เป็น Open Source ในการ test โปรแกรมที่เขียนโดยภาษา JAVA ซึ่งวิธีการทำงานของ Junit ประกอบไปด้วยการ Write และ Run โดยสามารถทำการรัน test ซ้ำๆ หลายๆ ครั้งได้.



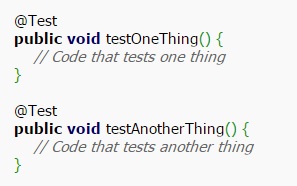
**รูปที่ 2.2** สัญลักษณ์ของ Junit Framework

สำหรับการพัฒนา unit test ด้วย Junit นั้นจะสามารถทำได้โดยการนำ jar file ของ Junit เข้ามาในโปรแกรม หลังจากนั้น test class ที่จะใช้งาน Junit ให้ทำการ import org.junit.\* ดังรูปที่ 2.3



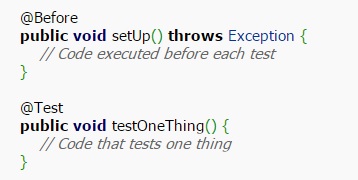
**รูปที่ 2.3** library ของ Junit

ส่วน test case หรือ test method นั้นให้ทำการกำกับคำว่า @Test เอาไว้เพื่อเป็นการบอก JUnitว่าเมท็อดนั้นเป็น test method ดังรูปที่ 2.4 ซึ่งจะมีประโยชน์ในการแยกเมท็อดธรรมดาและเมท็อดทดสอบ



**รูปที่ 2.4** @Test Annotation

โดย Junit มาพร้อมกับ บรรณนิทัศน์ (Annotation) หลายแบบซึ่งมีหน้าที่ต่างๆกันเพื่อช่วยเพิ่มประสิทธิภาพในการทำ unit test เช่น @Before เมื่อนำไปกำกับเมท็อด เมท็อดนั้นจะถูกรันก่อนที่จะทำ test case ทุกกรณี ดังเช่นรูปที่ 2.5 ถ้าเราสั่งรันเมท็อด testOneThing เมท็อด setup จะทำงานก่อนเสมอ.



**รูปที่ 2.5** @Before Annotation

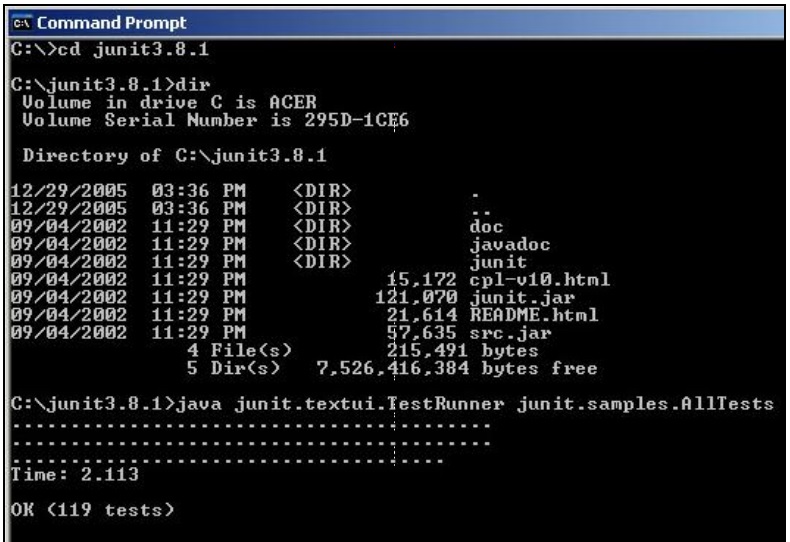
โดย Junit นั้นจะมีการรัน test อยู่ 2 รูปแบบ [4] คือ Batch TestRunner และ Graphical TestRunner โดย Batch TestRunner นั้นหมายถึงการสั่งงานการรัน test จะทำด้วย command line ส่วน Graphical TestRunner นั้นสามารถแบ่งออกได้เป็นแบบ awt-based และ swing-based ซึ่ง Graphical TestRunner จะสามารถสั่งรัน test ผ่านส่วนแสดงผลผู้ใช้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

1. Batch TestRunner

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



**รูปที่ 2.6** หน้าจอแสดงผล batch TestRunner

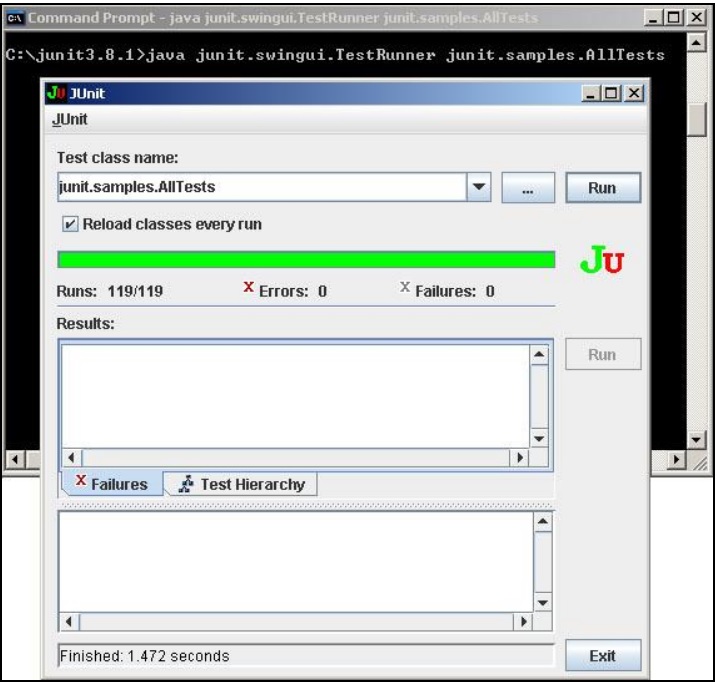
จากภาพแสดงจำนวนจุด ( . ) มีจำนวน 119 module. Time: 2.113 หมายถึงเวลาที่ใช้ในการ Test. OK <119 tests> หมายถึงการ Test ผ่านการทดสอบหมด ไม่เกิด Error ใดๆ ขึ้นมา ดังนั้นหมายความว่า Code นี้ไม่มี error และผ่านการทดสอบโดยวิธี batch test

1. Graphical TestRunner

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 TestClass และแสดงผลออกมาในรูปกราฟฟิกดังรูปที่ 2.7



**รูปที่ 2.7** หน้าจอแสดงผล graphical TestRunner

โดยรูปที่ 2.7 นั้นเป็นหน้าจอโปรแกรม JUnit ที่กำลังรันไฟล์ AllTests โดยขณะที่กำลังรัน

แถบสีเขียวจะวิ่งจากด้านซ้ายไปด้านขวาเมื่อ Run เสร็จแล้วจะได้ดังภาพ

จากรูปภาพ 5.2 แสดงผลการ Run ดังนี้

Runs: 119 หมายถึง การ Run จำนวน 119 test case

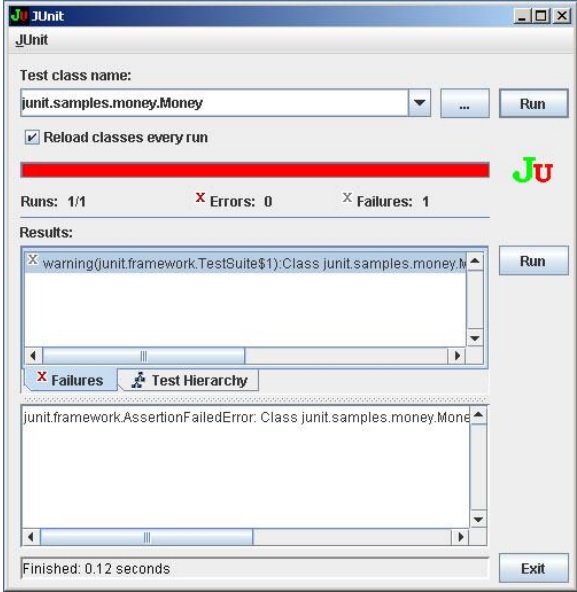
Errors: 0 หมายถึง ไม่พบ error

Failures: 0 หมายถึง ไม่พบ Failure

Finished: 1.182 Seconds หมายถึง เวลาที่ใช้ในการ Run test case ทั้งหมด

ส่วนในกรณีที่มี error เกิดขึ้น แถบสีจะเป็นสีแดงและมีข้อความแสดง error ที่เกิดขึ้น

ดังรูปที่ 2.8



**รูปที่ 2.8** หน้าจอแสดงผลกรณี Junit ผิดพลาด

**2.3 Test-driven development ( TDD )**

Test-driven development หรือ TDD คือ [5] กระบวนการพัฒนาซอฟต์แวร์รูปแบบหนึ่งซึ่งใช้หลักการคือ มีวงจรการพัฒนาที่สั้น แต่เน้นการทำซ้ำไปเรื่อยๆ. โดยวงจรการพัฒนาซอฟต์แวร์แบบ TDD นั้นจะเป็นดังรูปที่ 2.9



**รูปที่ 2.9** 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

ข้อดีของ Test-driven development [6] คือ

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

**2.4 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 ดังนั้นคลาสคือที่เก็บเมทอด (Method) หรือพฤติกรรม (Behavior) ซึ่งมีสถานะ (State) และรูปพรรณ (Identity) ประจำพฤติกรรม (Behavior)



**รูปที่ 2.10** สัญลักษณ์ภาษา Java

**ข้อดีของภาษาจาวา**

1) ภาษาจาวาเป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุแบบสมบูรณ์ ซึ่งเหมาะในการพัฒนาระบบที่มีความซับซ้อน

2) โปรแกรมที่เขียนขึ้นโดยใช้ภาษาจาวาสามารถทำงานได้ในระบบปฏิบัติการที่แตกต่างกัน

3) ภาษาจาวามีการตรวจสอบข้อผิดพลาดทั้งตอนคอมไพล์และเวลารันทาให้ลดข้อผิดพลาดที่อาจเกิดขึ้นในโปรแกรม และช่วยให้แก้จุดบกพร่องของโปรแกรมได้ง่าย

4) ภาษาจาวามีความซับซ้อนน้อยกว่าภาษา C++ ทำให้ใช้งานได้ง่ายกว่าและลดความผิดพลาดได้มากขึ้น

5) ภาษาจาวาถูกออกแบบมาให้มีความปลอดภัยสูงตั้งแต่แรก ทำให้โปรแกรมมีความปลอดภัยมากกว่าโปรแกรมที่เขียนขึ้นด้วยภาษาอื่น

6) มี IDE แอพพลิเคชันเซิร์ฟเวอร์ และไลบลารีมากมายสาหรับจาวาที่สามารถใช้งานได้โดยไม่ต้องเสียค่าใช้จ่าย

**ข้อเสียของภาษาจาวา**

1) ทางานได้ช้ากว่า Native code เพราะว่าโปรแกรมที่เขียนขึ้นด้วยภาษาจาวาจะถูกแปลงเป็นภาษากลางก่อน หลังจากนั้นจึงจะถูกเปลี่ยนเป็นภาษาเครื่องซึ่ง Native code อยู่ในรูปของภาษาเครื่องอยู่แล้ว

2) Tools ที่ใช้ในการพัฒนาโปรแกรมจาวามักไม่ค่อยครอบคลุม ทาให้หลายอย่างโปรแกรมเมอร์จะต้องเป็นผู้พัฒนา

**2.5 Spring framework**

Spring framework หรือ Spring [8] คือ framework ตัวหนึ่งของภาษา Java ที่ถูกสร้างขึ้นมาเพื่อช่วยให้การพัฒนาซอฟต์แวร์ด้วยภาษา java ทำได้ง่ายขึ้น ช่วยสนับสนุนในทุกๆแง่มุมโครงสร้างที่ช่วยในการพัฒนาระบบ เช่นในส่วนของ Persistence, Transaction management, security และอื่นๆอีกมากมาย ที่จำเป็นต้องใช้ในการพัฒนาระบบขนาดใหญ่เช่น Enterprise software. Spring framework จะดูแลในส่วนเหล่านี้ ทำให้การพัฒนาซอฟต์แวร์ประเภท Enterprise นั้น ผู้พัฒนาสามารถที่จะมุ่งเน้นไปที่การพัฒนาตัวโปรเซสของระบบ ไม่จำเป็นต้องเสียเวลาในการมาพัฒนาส่วนเหล่านี้ ซึ่งเป็นส่วนที่ Enterprise software ส่วนใหญ่ต้องใช้อยู่แล้ว.



**รูปที่ 2.11** สัญลักษณ์ Spring framework

Spring framework มีจุดเด่นในการเน้นไปที่การใช้ POJO (Plain old java object) หรือ object ธรรมดาของ Java เพื่อใช้ในการพัฒนาระบบทั้งระบบ ทำให้สามารถจัดการคลาสต่างๆในระบบได้ง่าย ซึ่งเป็นจุดเด่นเหนือการใช้ EJB (Enterprise Java bean) ที่คลาสมีขนาดใหญ่และซับซ้อน.

โดย Spring framework มีหัวใจหลักในการทำงานด้วยสิ่งที่เรียกว่า Ioc (Inversion of control) และ DI (Dependency injection) ซึ่งทั้ง 2 ตัวนี้ทำให้ Spring เป็น framework ที่พัฒนาระบบที่มีการจัดการซอร์สโค้ดอย่างเป็นระบบ ทดสอบได้ง่าย แบ่งระบบออกเป็นส่วนต่างๆได้ง่าย

Spring framework มาพร้อมกับโมดูลย่อยๆประมาณ 20 โมดูลดังรูปที่ 2.12



**รูปที่ 2.12** โมดูลของ Spring framework

โดยโมดูลทั้งหมดจะถูกรวบรวมเป็นกลุ่มๆตามหน้าที่เป็นกลุ่มหลักๆคือ 1.Core container 2.Web 3.Aop (Aspect Oriented Programming) 4.Instrumention 5.Messaging 6. Data Access/Integration และ 7.Test ซึ่งผู้พัฒนาสามารถเลือกนำไปใช้เฉพาะโมดูลที่ตนเองต้องการเพื่อลด overhead ในการนำ Spring ทุกโมดูลเข้าไปในการพัฒนา

**2.6 Spring MVC**

**Spring MVC Framework คืออะไร ?**

Spring MVC framework [9] คือ framework 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

**Spring MVC Framework มีไว้ทำไม ?**

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

ตัว Spring MVC framework มีการกำหนด Form การเขียนเว็บในแต่ละส่วน ไม่ว่าจะเป็นส่วน Request , Respond หรือ Controller ต่างๆ ทำให้ไม่ว่าจะเป็น developer คนไหนที่เข้ามาแก้ไขไฟล์ก็สามารถเข้าใจ Code ที่เขียนไว้ได้ง่ายๆ และยังมีตัว Library ที่ไว้ช่วยในการเขียน เพื่อให้ง่ายต่อการทำงานของ developer อีกด้วย

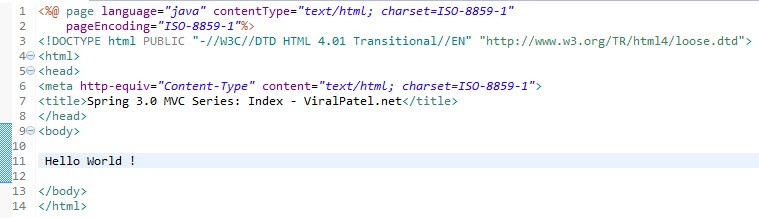
**ตัวอย่างการเขียน Spring MVC framework**

รูปที่ 2.13 คือไฟล์ที่มี Dispatcher Servlet



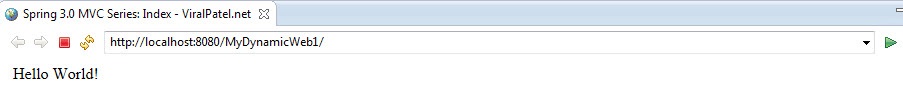
**รูปที่ 2.13** ไฟล์ Dispatcher servlet

รูปที่ 2.14 คือไฟล์ View ที่อ้างอิงจาก Dispatcher Servlet



**รูปที่ 2.14** ไฟล์ View ของ Spring MVC

หน้าผลลัพธ์ที่แสดงเมื่อส่ง request ผ่าน Dispatcher Servlet และได้รับ View กลับมาแสดงดังรูปที่ 2.15

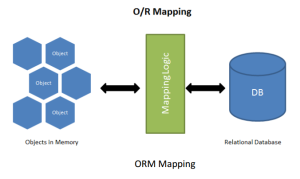


**รูปที่ 2.15** หน้าผลลัพธ์ของ Spring MVC

**2.7 JPA (Java Persistence API)**

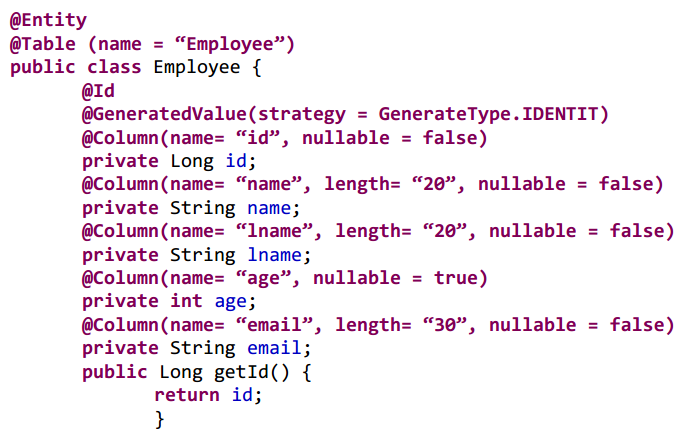
JPA [10] คือ API มาตรฐานตัวหนึ่งของ Java ที่ใช้ในการทำ ORM (Object Relation Mapping)

โดย ORM คือ เครื่องมือที่ช่วยให้เราสามารถมองฐานข้อมูลเป็น Object ถ้าเราแก้ไขข้อมูลใน Object ก็จะทำให้ข้อมูลในฐานข้อมูลมีการเปลี่ยนแปลง เหมือนกับการใช้ SQL Command ตรงๆ ดังเช่นรูปที่ 2.16



**รูปที่ 2.16** ORM Mapping

ซึ่งการทำ ORM จะทำให้เราไม่จำเป็นต้องติดต่อกับฐานข้อมูลโดยตรงด้วยการเขียนคำสั่ง SQL ที่เป็นการติดต่อพื้นฐานของ JDBC (Java database connectivity technology) ซึ่งถ้าระบบมีขนาดใหญ่จะมี SQL Command จำนวนมหาศาล ทำให้ดูแลจัดการยาก ยากต่อการแก้ไขต่างๆ โดยการทำ ORM นั้น ยกตัวอย่างเช่น เรามีตารางฐานข้อมูล เก็บข้อมูลลูกค้า หากเรามองตารางนั้นเป็น Object ก็จะทำการเขียนClass ของ object นั้น ดังรูปที่ 2.17



**รูปที่ 2.17** Class ของ ORM

ซึ่งใน Class จะมี Annotation สำคัญๆเพื่อใช้บ่งบอกถึงค่าต่างๆในตารางดังนี้

1. @Entity ใช้สำหรับบอกว่าเป็น Class ที่ Map กับตารางในฐานข้อมูล
2. @id และ @GeneratedValue ใช้สำหรับกำหนด Field ที่เป็น Primary Key กรณีที่เป็น id ประเภท Auto Generate จะใช้ @GeneratedValue
3. @Column ใช้สำหรับกำหนดชนิดของ column ของฐานข้อมูล เช่น ความยาว และ Nullable