**(PSC) Classic Framework Training (Webform)  
#1 07 September 18**- OOP  
- Pattern & Framework (camel, store procedure)   
- Ajax (Asynchronous JavaScript and XML)   
- Validation (client, server)   
- Transaction Scope  
- Error Exception, Debugging

**Objective**  
1. เพื่อให้สามารถเข้าใจการพัฒนา Software โดยใช้หลักการ OOP  
2. เข้าใจ Software Layer เพื่อนำไปออกแบบและพัฒนาได้  
3. สามารถนำ Pattern & Framework ไปใช้ในการพัฒนา Software ได้  
4. สามารถเข้าใจหลักการของ Ajax เพื่อนำไปพัฒนาใน Project ได้ (Update Panel)  
5. สามารถเข้าใจหลักการของ Transaction Scope เพื่อนำไปพัฒนาใน Project ได้  
6. สามารถเข้าใจประโยชนของการ Try Catch & Debugging เพื่อนำไปพัฒนาใน Project ได้

**ความหมายของ OOP**

OOP (ย่อจาก Object Oriented Programming การเขียนโปรแกรมแบบวัตถุวิธี) เป็นรูปแบบ (Paradigm) หรือแนวคิดอย่างหนึ่ง อันมีจุดมุ่งหมายเพื่อการสร้างซอฟท์แวร์ คำว่า Object ในที่นี้หมายถึงวัตถุ หรือสิ่งของที่จับต้องได้ ไม่ได้หมายถึงวัตถุประสงค์ จุดมุ่งหมาย หรือกรรม (ผู้ถูกกระทำ อย่างประธาน กริยา และกรรม ในรูปประโยค)

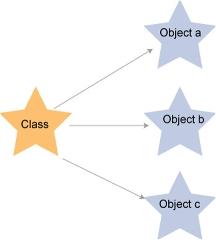
มีผู้ตีความว่าแนวคิดนี้คือ “การเขียนโปรแกรมโดยมุ่งที่เป้าหมาย (มิได้เน้นที่กระบวนการ)” ผู้เขียนเห็นว่าการตีความดังกล่าวไม่สู้จะถูกต้อง เพราะการเขียนโปรแกรมตามลัทธินี้มีการสร้าง object ขึ้นจริง แม้จะเป็น object อันจับต้องไม่ได้ เพราะอยู่ในสภาพซอฟท์แวร์ แต่ก็มีเจตนาจะเลียนแบบ object ที่เป็นรูปธรรมอย่างเต็มที่

เรื่อง OOP เป็นเรื่องของ object ดังนั้นสิ่งแรกที่ต้องรู้คือความหมายของคำว่า object นิยามของ object คือ “หน่วยหนึ่งของโปรแกรมซึ่งมีหน้าที่การทำงานอันเฉพาะเจาะจง และถูกกำหนดปฏิสัมพันธ์กับโปรแกรมหน่วยอื่นๆ ไว้อย่างแน่ชัด” ในภาษา C# เราสร้าง object จากคลาส

**คลาสกับ object**

คลาสกับ object เป็นสองสิ่งที่มีความเกี่ยวข้องกันโดยตรง คลาสคือโค้ดที่เราเขียนขึ้นเพื่อทำหน้าที่เป็นพิมพ์เขียวของ object การสร้าง object จากคลาสเรียกว่าการทำ instantiation การสร้าง object จากคลาส เทียบได้กับการทำขนมครกสิงคโปร์ เตาขนมจะมีหลุมหลายหลุม แต่ละหลุมมีลวดลายไม่เหมือนกัน ขนมครกที่ได้จากแต่ละหลุมจึงมีรูปร่างต่างๆ กัน คลาสคือหลุมหนึ่งหลุม ขนมครกที่ได้คือ instance ของคลาส หรือ object

เราเรียก object หนึ่ง object ว่าหนึ่ง instance เราสามารถสร้าง object ได้หลายๆ instance จากคลาสเพียงคลาสเดียว จากตัวอย่างขนมครกสิงคโปร์ ในวันหนึ่งๆ แม่ค้าจะทำขนมครกได้เป็นจำนวนมากจากหลุมแต่ละหลุม ส่วนในภาษา C# หากเราสร้างคลาสหนึ่งคลาส ยกตัวอย่างเช่น เป็นคลาสเพื่อนิยาม node หลังจากนั้นเราอาจเขียนโปรแกรมสร้าง binary tree ซึ่งขณะทำงานมันอาจจะสร้าง object จากคลาส node ได้หลายล้าน instance ภายในหนึ่งวินาที

[](http://tk1.storage.msn.com/x1pGg9EMswqL--qLT8LR13laWKiXSaJy8OONnR7y7vlE5R749pbfJTANgDXFrrFinBakGqNov8QRLAvEBpN3kuHpmACs4U4KxMNtt5e4y-D5jZiDdn8YoiIP1X66uuwL-Od8gQSe2QZN03CY9kpTUQpYP9KI3uMxmB3)

คลาสหนึ่งคลาสใช้สร้าง object ได้จำนวนมาก แม้ object แต่ละตัวจะมีคุณสมบัติและการทำงานเหมือนกันหมด แต่ object แต่ละตัวจะมีสถานะและข้อมูลเป็นของตัวเองโดยไม่ขึ้นกับคลาสหรือ object อื่นๆ

การใช้งานคลาสทำได้สองวิธี วิธีแรกคือการนำไปใช้สร้าง object และใช้งานผ่าน object ดังที่อธิบายไปแล้ว อีกวิธีหนึ่งคือเรียกใช้โดยตรงโดยไม่ต้องสร้าง object เรียกว่า static class แม้การใช้งานคลาสวิธีนี้จะไม่มี object ทำให้ดูเหมือนไม่เป็นหลักการ OOP แต่ในบางสถานการณ์ก็ถือว่ามี**ความเหมาะสม**ดี

ภาษา C# สนับสนุนการสร้าง type ใหม่อย่างพิสดาร เมื่อนิยามแล้ว UDT กับ build-in type จะมีศักดิ์ศรีเสมอกัน (คือ compiler จะปฏิบัติต่อ type ที่เราสร้างในลักษณะ first class เช่นเดียวกับ build-in type).NET Framework library เองก็เป็นแหล่งรวม type หลายพัน type ที่เราสามารถนำมาประกอบเป็น type ใหม่ที่มีการทำงานซับซ้อนได้อย่างสะดวกรวดเร็ว

การเขียนโปรแกรมแบบ OOP เราจะใช้เวลาส่วนมากไปกับการออกแบบและนิยาม type ขึ้นใหม่เพื่อนำไปใช้สร้าง object ต่างๆ เมื่อนำ object ทั้งหมดมาประกอบเข้าด้วยกัน เพื่อให้ทำงานร่วมกัน จะทำให้โปรแกรมของเราสร้างผลลัพธ์ได้ตามความประสงค์

**การใช้งาน object**

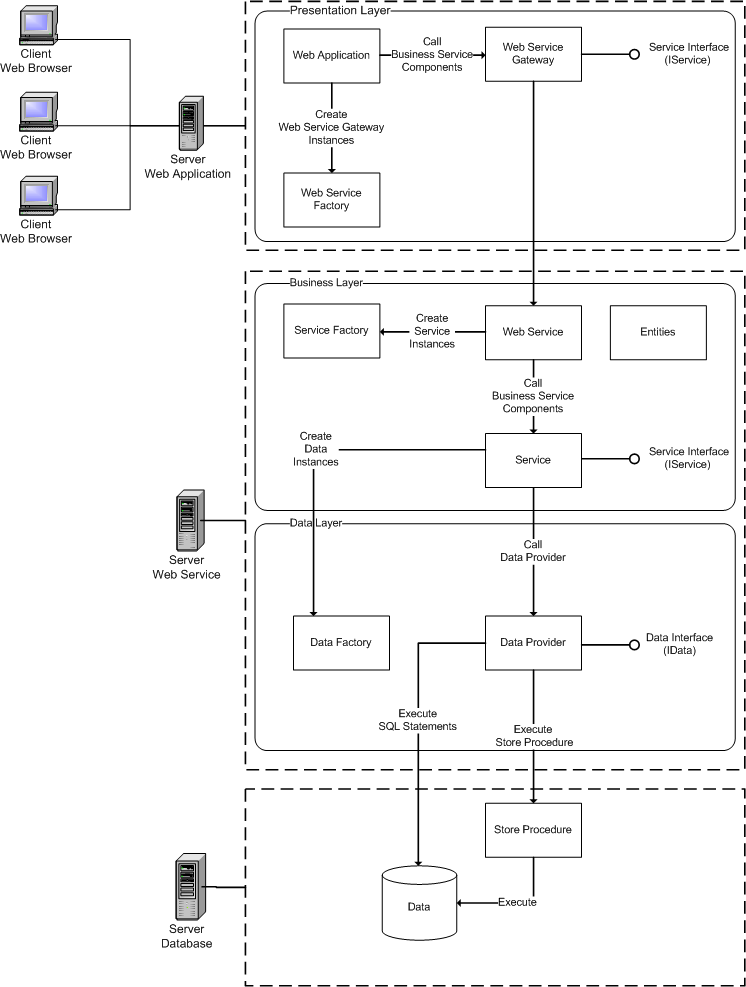
การใช้งาน object ต้องทำได้ง่าย เราอาจมองว่า object เป็น “กล่องดำ” ในทางวิทยาศาสตร์เราจะเรียกอุปกรณ์ที่เราสามารถนำมาใช้ประโยชน์ได้ โดยไม่จำเป็นต้องรู้รายละเอียดการทำงานภายในของมันว่า กล่องดำ ยกตัวอย่างเช่นโทรศัพท์ เรานำมันมาใช้ประโยชน์เพื่อการสื่อสารได้ โดยไม่จำเป็นต้องรู้ว่ามันทำงานได้อย่างไร เช่นเดียวกัน ในภาษา C# เราสามารถนำ object มาใช้งานได้โดยไม่จำเป็นต้องรู้ว่ามันมี source code ภายในเป็นอย่างไร

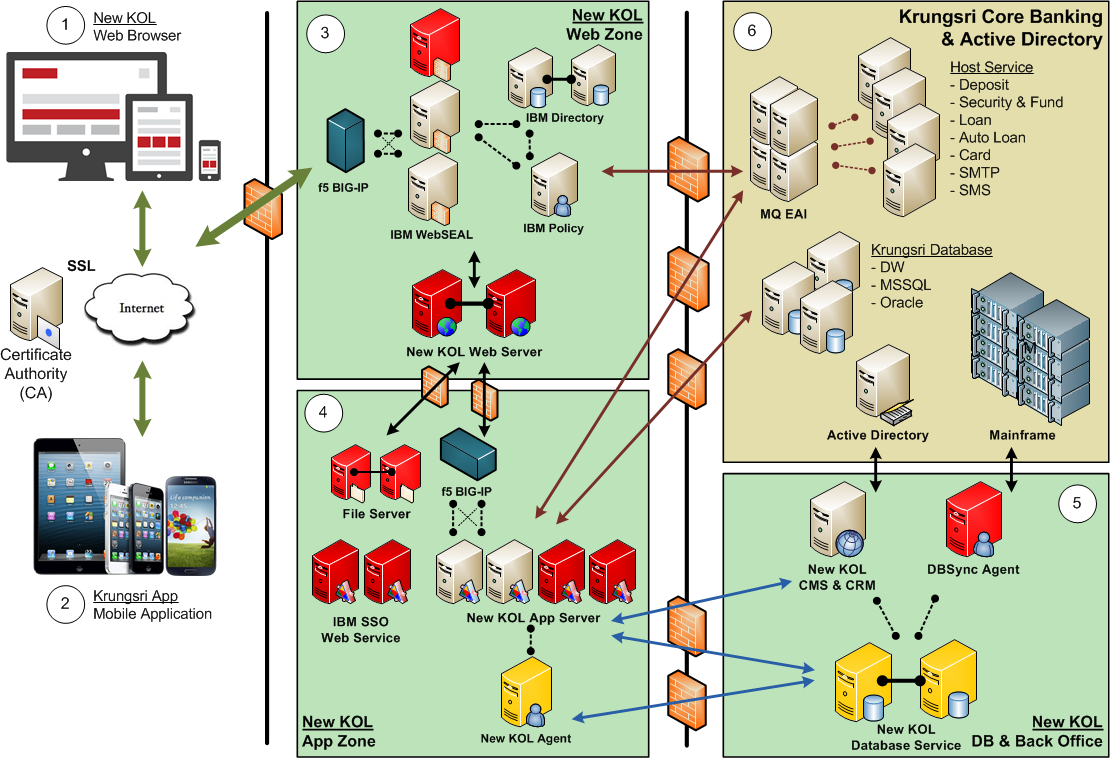
เมื่อเราต้องกาเรียกให้ object ทำงานบางอย่างเราจะเรียก method ของมัน ยกตัวอย่างเช่นโทรศัพท์มี method คือการโทรฯ ออก การวางสาย การพักสาย การบันทึกเลขหมายฯ ในภาษา C# เราจะนิยาม method เหล่านี้โดยการเขียนโค้ดหนึ่งชุด คล้ายการนิยามฟังก์ชันในภาษา C

object จะผนวกกระบวนการ (method) และข้อมูล (information) ไว้ภายในตัวของมันเอง จากตัวอย่างโทรศัพท์เก็บเลขหมายที่เราบันทึกไว้ โดยเราไม่จำเป็นต้องรู้ว่ามันเก็บไว้ที่ไหนอย่างไร เรารู้เพียงวิธีดึงเลขหมายที่บันทึกไว้ออกมาใช้งานก็พอ ในภาษา C# object จะเก็บข้อมูลภายในไว้ใน field (คือตัวแปรท้องถิ่นของคลาส) โปรแกรมที่เรียกใช้ object ไม่จำเป็น (และไม่สามารถ) เข้าถึงหรือเปลี่ยนแปลงข้อมูลใน field ได้โดยตรง แต่สามารถเข้าถึงทางอ้อมได้ผ่านส่วนเชื่อมต่อที่เรียกว่า property

ภาษา C# สนับสนุนหลักการทุกรูปแบบในลัทธิ OOP โดยหลักการนี้ มีเรื่องหลัก 3 หัวข้อที่ท่านจำเป็นต้องเข้าใจ คือ  
• Encapsulation: การห่อหุ้มวิธีการและข้อมูลไว้ภายในหน่วยหนึ่งๆ  
• Inheritance: การสืบทอดคุณสมบัติ  
• Polymorphism: การแปรคุณสมบัติ (หนึ่งรูปหลายพฤติกรรม Overloading, Overriding)

**- Pattern & Framework**





**- Transaction Scope**

public bool DeleteSchemaMapping(string pSchemaMappingID, string pProgramID, string pCurrentUser, string pCompanyCode)

{

try

{

bool Result = false;

using (TransactionScope scope = new TransactionScope(TransactionScopeOption.Required))

{

using (ISchemaVersionData iData = DataFactory.GetSchemaVersionData())

{

Result = iData.DeleteSchemaMapping(pSchemaMappingID,pProgramID,pCurrentUser,pCompanyCode);

}

scope.Complete();

}

return Result;

}

catch (Exception ex)

{

throw new BSLException(this, "3202", "DeleteSchemaVersion method occurs an error.", ex, false);

}

}

**TransactionScope**  
อยู่ภายใน System.Transactions เมื่อเรียกมาใช้งานจะมีรูปแบบการทำงานเป็นบล๊อก คือ จะควบคุมการใช้ Transaction ภายในบล๊อกของตัวเองเท่านั้น และจะทำการ Rollback Transaction อัตโนมัติเมื่อคำสั่งภายในบล๊อกทำงานไม่สำเร็จ   
  
using (TransactionScope scope = new TransactionScope(TransactionScopeOption.Required))  
{  
         //คำสั่งการทำงานกับฐานข้อมูล

1

2

3  
  
         scope.Complete(); //หากการทำงานไม่สำเร็จถึงบรรทัดนี้ จะทำการ Rollback ทันที  
}

ค่าของ TransactionScopeOption จะสามารถระบุได้ 3 ค่า คือ

* Required : หากไม่มี Transaction อื่นอยู่ให้สร้างขึ้นใหม่ หากมีอยู่แล้วให้ใช้ Transaction เดิม
* Requires New : ไม่สนใจว่ามี Transaction อื่นใดอยู่หรือไม่ ให้สร้างใหม่เสมอ
* Suppress : ไม่สร้าง Transaction

**- Error Exception**  
 try

{

….  
}

catch (HttpException ex)

ex)

{

throw new DALException(this, "3209", "ReadRaexFileForValidate method occurs an error.", ex, false);

}

catch (Exception ex)

{

throw new DALException(this, "3209", "ReadRaexFileForValidate method occurs an error.", ex, false);

}

finally

{

if (!string.IsNullOrEmpty(serverName))

{

using (AuthenService authenService = new AuthenService())

{

authenService.LogoutFromShare(serverName, shareName);

}

}

}

- Access Modifiers

•public

•protected

•internal

•private

* Basic Event handler  
  •Page\_Load  
  •Page\_Init  
  •Search\_Click  
  •grdxxxx\_PageIndexChanging  
  •grdxxxx\_RowDataBound

- MasterPage  
- BasePage  
- Themes, .CSS  
- GlobalResources  
- LocalResources  
- WebService  
- Store ProcedureAssignment #1 **ส่งงาน 11/09/2018 - 19:00**

1. ศึกษาโครงสร้าง Project แบบ N tire architecture

2. ทำหน้าจอ   
 - Add/Edit/Delete Account  
 - Search Account and / or amount  
     - validation   
 - Account No format - client  
 - Exist Account No - server

3. ติดต่อด้วย Store Procedure

4. เพิ่ม Class Exception UIL, BSL, DAL

ส่งงาน [panan.pe@hotmail.com](mailto:panan.pe@hotmail.com)