

การพัฒนาเว็บแอปพลิเคชันโดยใช้เฟรมเวิร์กแองกูลาร์เจเอสแบบอิงแบบจำลอง

นายวุฒิชัย จันทร์สุวัฒน์

โครงงานมหาบัณฑิตนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต  
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์  
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย  
ปีการศึกษา 2558  
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

.A Model-Driven Development of Web-Based Applications  
Using AngularJS Framework

Mr Wutthichai Chansuwath

A Master Project Submitted in Partial Fulfillment of the Requirements  
for the Degree of Master of Science in Software Engineering  
Department of Computer Engineering  
Faculty of Engineering  
Chulalongkorn University  
Academic Year 2015  
Copyright of Chulalongkorn University

หัวข้อโครงงานมหัศจรรย์

การพัฒนาเว็บแอปพลิเคชันโดยใช้เฟรมเวิร์กแองกูลาร์เจเอสแบบ  
อิงแบบจำลอง

โดย

นายวุฒิชัย จันทร์สุวัฒน์

ภาควิชา

วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษา

รศ. ดร.ทวีชัย เสนีวงศ์ ณ อยุธยา

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้  
โครงงานมหัศจรรย์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต

.....  
(ผศ. ดร. ณัฐวุฒิ หนูไพโรจน์)

หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์

.....  
(รศ. ดร.ทวีชัย เสนีวงศ์ ณ อยุธยา)

อาจารย์ที่ปรึกษา

วุฒิชัย จันทรสุวัฒน์ : การพัฒนาเว็บแอปพลิเคชันโดยใช้เฟรมเวิร์กแองกูลาร์เจเอสแบบอิงแบบจำลอง (A Model-Driven Development of Web-Based Applications Using AngularJS Framework) อาจารย์ที่ปรึกษา : รศ. ดร.ทวีติย์ เสนิงค์ ณ อยุธยา, จำนวน 75 หน้า

แองกูลาร์เจเอสเป็นจาวาสคริปต์เฟรมเวิร์กประเภทหนึ่งซึ่งเป็นที่นิยม สำหรับการพัฒนาเว็บแอปพลิเคชันแบบซิงเกิลเพจซึ่งสนับสนุนวิวัฒนาการแบบไดนามิก เพื่อเป็นการช่วยเหลือนักพัฒนาเว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจเอส งานวิจัยชิ้นนี้จึงได้นำหลักการของการพัฒนาแบบอิงแบบจำลองเข้ามาช่วยในการพัฒนาเว็บแอปพลิเคชันซึ่งใช้แองกูลาร์เจเอส งานวิจัยชิ้นนี้ได้นำเสนออยู่เอ็มแอลโพรไฟล์สำหรับเฟรมเวิร์กแองกูลาร์เจเอส สำหรับใช้ในการสร้างแบบจำลองของเว็บแอปพลิเคชัน และได้มีการนำเสนอกฎการแปลงสำหรับการแปลงแบบจำลองไปเป็นโค้ดเทมเพลต นักพัฒนาต้องทำการเติมโค้ดบางส่วนลงในเทมเพลตเพื่อให้เว็บแอปพลิเคชันสามารถทำงานได้อย่างสมบูรณ์ พร้อมกันนี้งานวิจัยชิ้นนี้ยังได้สร้างเครื่องมือที่ช่วยทำการแปลงแบบจำลองเป็นโค้ดเทมเพลต งานวิจัยชิ้นนี้ยังได้ใช้เว็บแอปพลิเคชันกรณีศึกษาเพื่อประเมินอัตราการแปลงโค้ด ผลการประเมินพบว่าวิธีการที่ได้นำเสนอไปนั้นมีสัดส่วนการแปลงโค้ดได้ประมาณ 87% ของโค้ดที่ทำงานได้ จึงสามารถลดระยะเวลาในการพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสได้

ภาควิชา วิศวกรรมคอมพิวเตอร์.....ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมซอฟต์แวร์.....ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา .....2558

# # 5770968021 : MAJOR SOFTWARE ENGINEERING

KEY WORD : AngularJS, MDA, UML Profile, Transformation Rules

WUTTHICHAJ CHANSUWATH : A MODEL-DRIVEN DEVELOPMENT OF WEB-BASED APPLICATIONS USING ANGULARJS FRAMEWORK.

MASTER PROJECT ADVISOR : ASSOC.PROF. DR.TWITTIE SENIVONGSE, 75 pp.

AngularJS is one of the widely used frameworks for modern single-page web application development which is designed to support dynamic views in the applications. To further assist AngularJS developers, this research proposes how the concept of model-driven development can be applied to AngularJS-based development. We propose a UML profile for AngularJS for building a model of an AngularJS web application, and a set of transformations that transform the model into a code template. The developer can then fill in the template to make a complete workable web application. Also, a transformation tool is developed to assist in constructing the code template. Using a case study application, the evaluation in terms of transformation rate shows that the automatically generated code covers 87% of the complete code of the case study, which means it could greatly help reduce development time.

Department of Computer Engineering..... Student's signature.....

Field of study Software Engineering..... Advisor's signature.....

Academic year.....2015

## กิตติกรรมประกาศ

ขอกราบขอบพระคุณ รศ. ดร.ทวีติย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาโครงการ เป็นอย่างยิ่งที่ได้สละเวลาให้คำปรึกษา คำแนะนำ และแนวทางสำหรับการทำโครงการมหำบัณฑิต รวมทั้งเป็นผู้ประสานงานให้ความช่วยเหลือแก่นิสิตที่ทำโครงการทุกคน

ขอกราบขอบพระคุณ รศ.ดร.สมชาย ประสิทธิ์จูตระกูล รศ.ดร.พรศิริ หมั่นไชยศรี และรศ. ดร.ทวีติย์ เสนีวงศ์ ณ อยุธยา คณะกรรมการคุมสอบโครงการมหำบัณฑิตเป็นอย่างยิ่ง ที่ได้กรุณาแนะนำแนวทาง รวมถึงการตรวจสอบและแก้ไขโครงการมหำบัณฑิตนี้

ขอขอบคุณ คณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้คำแนะนำ ความรู้และแนวทางการทำโครงการ

ขอขอบคุณ เพื่อนๆ หลักสูตรวิศวกรรมซอฟต์แวร์ สำหรับกำลังใจและคำแนะนำในการจัดทำโครงการมหำบัณฑิต

ขอขอบคุณ ภริยา ที่สนับสนุนและคอยเป็นกำลังใจ ในทุก ๆ เรื่องจนสามารถผ่านพ้นไปได้

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา มารดา รวมถึงสมาชิกในครอบครัวที่ให้การสนับสนุน และให้กำลังใจที่ดีเสมอมา

วุฒิชัย จันทร์สุวัฒน์

## สารบัญ

	หน้า
บทคัดย่อภาษาไทย.....	ง
บทคัดย่อภาษาอังกฤษ.....	จ
กิตติกรรมประกาศ .....	ฉ
สารบัญ .....	ช
สารบัญตาราง .....	ญ
สารบัญภาพ .....	ฎ

## บทที่

1. บทนำ.....	1
1.1 ความเป็นมาและความสำคัญของปัญหา .....	1
1.2 วัตถุประสงค์ของโครงการ .....	2
1.3 ขอบเขตของโครงการ .....	2
1.4 ขั้นตอนและวิธีการดำเนินโครงการ .....	3
1.5 ประโยชน์ที่คาดว่าจะได้รับ .....	3
1.6 ผลงานที่ตีพิมพ์จากโครงการมหาบัณฑิต.....	3
2. ทฤษฎี และ งานวิจัยที่เกี่ยวข้อง.....	4
2.1 ทฤษฎีที่เกี่ยวข้อง.....	4
2.1.1 สถาปัตยกรรมอิงแบบจำลอง .....	4
2.1.2 ยูเอ็มแอล.....	5
2.1.3 ยูเอ็มแอลโปรไฟล์ .....	5
2.1.4 แอ่งกูลาร์เจเอส.....	6
2.1.5 การพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแอ่งกูลาร์เจเอส .....	7
2.2 งานวิจัยที่เกี่ยวข้อง.....	7
2.2.1 Transformation from Web PSM to Code.....	7
2.2.2 A Component-Centric UML Based Approach for Modeling the Architecture of Web Applications.....	8
2.2.3 Visual Modeling for Web 2.0 Applications Using Model Driven Architecture Approach .....	9
2.2.4 A Model-Driven Development of Web-Based Applications on Google App Engine Platform.....	9
3. ยูเอ็มแอลโปรไฟล์แอ่งกูลาร์เจเอส .....	11
3.1 การสร้างเมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแอ่งกูลาร์เจเอส .....	12
3.2 การสร้างโปรไฟล์แอ่งกูลาร์เจเอส.....	13
3.2.1 สถาปัตยกรรมไคลเอนต์ด้วยยูเอ็มแอล (Client Architecture with UML) .	14

## สารบัญ (ต่อ)

บทที่	หน้า
3.2.2	สถาปัตยกรรมความสัมพันธ์ด้วยยูเอ็มแอล (Association Architecture with UML) ..... 16
3.3	การตรวจสอบโปรไฟล์แองกูลาร์เจเอส..... 17
4.	กฎการแปลงแบบจำลองเป็นโค้ด ..... 19
4.1	กฎสำหรับแปลงแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์เป็นโค้ด ..... 19
4.1.1	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ WebPage..... 19
4.1.2	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ ScriptLanguage .. 20
4.1.3	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ CSS..... 21
4.1.4	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ AngularApp ..... 22
4.1.5	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Filter ..... 23
4.1.6	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Service ..... 24
4.1.7	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ AppRouter ..... 25
4.1.8	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ AppView..... 26
4.1.9	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Controller..... 27
4.1.10	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Scope ..... 28
4.1.11	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Template..... 29
4.1.12	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ MarkupLanguage 30
4.1.13	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Directive..... 31
4.1.14	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Directive Controller..... 32
4.1.15	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Directive Isolate Scope..... 33
4.1.16	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Directive Scope . 34
4.1.17	กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Directive Template ..... 35
4.2	การตรวจสอบกฎการแปลง ..... 36
5.	การออกแบบและพัฒนาเครื่องมือ ..... 39
5.1	ภาพรวมการพัฒนา ..... 39
5.2	การออกแบบหน้าที่การทำงานของเครื่องมือ ..... 40
5.3	การออกแบบเชิงแนวคิด..... 40



## สารบัญ (ต่อ)

บทที่	หน้า
5.4 ลำดับการทำงานของคลาสต่าง ๆ ของเครื่องมือ .....	42
5.5 ขั้นตอนการแปลงค่าจากไฟล์ XMI เป็นชุดของออบเจกต์ CommonXMIOject.....	44
5.6 การออกแบบส่วนต่อประสานของเครื่องมือ .....	45
5.6.1 การออกแบบส่วนต่อประสานสำหรับข้อมูลนำเข้า.....	46
5.6.2 การออกแบบส่วนต่อประสานสำหรับการแสดงผลลัพธ์ .....	46
5.7 เครื่องมือที่ใช้ในการพัฒนาระบบ .....	46
5.7.1 ฮาร์ดแวร์ที่ใช้ในการพัฒนาระบบ .....	46
5.7.2 ซอฟต์แวร์ที่ใช้ในการพัฒนาระบบ .....	46
5.8 การพัฒนาระบบ .....	47
5.8.1 ขั้นตอนการพัฒนาระบบ .....	47
5.8.2 ตัวอย่างการใช้งานระบบ .....	47
6. การทดสอบและประเมินผล .....	54
6.1 การพัฒนาระบบดูข้อมูลพื้นฐานของหุ่นออนไลน์ .....	54
6.2 การประเมินผล .....	62
7. บทสรุปโครงการและข้อเสนอแนะ .....	64
7.1 สรุปผลโครงการมหาบัณฑิต .....	64
7.2 ข้อจำกัดในการทำโครงการ .....	64
7.3 ข้อเสนอแนะ .....	65
รายการอ้างอิง .....	66
ภาคผนวก .....	68
ภาคผนวก ก แผนภาพและตารางคำอธิบายยูสเคส .....	69
ประวัติผู้เขียนโครงการมหาบัณฑิต .....	75

## สารบัญตาราง

	หน้า
ตารางที่ 3.1 รายละเอียดแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์ .....	14
ตารางที่ 3.2 รายละเอียดแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์ .....	17
ตารางที่ 4.1 แม่พิมพ์ต้นแบบและแบบจำลองของ WebPage .....	20
ตารางที่ 4.2 แม่พิมพ์ต้นแบบและแบบจำลองของ ScriptLanguage .....	21
ตารางที่ 4.3 แม่พิมพ์ต้นแบบและแบบจำลองของ CSS.....	21
ตารางที่ 4.4 แม่พิมพ์ต้นแบบและแบบจำลองของ AngularApp .....	22
ตารางที่ 4.5 แม่พิมพ์ต้นแบบและแบบจำลองของ Filter .....	23
ตารางที่ 4.6 แม่พิมพ์ต้นแบบและแบบจำลองของ Service .....	24
ตารางที่ 4.7 แม่พิมพ์ต้นแบบและแบบจำลองของ AppRouter .....	25
ตารางที่ 4.8 แม่พิมพ์ต้นแบบและแบบจำลองของ AppView .....	26
ตารางที่ 4.9 แม่พิมพ์ต้นแบบและแบบจำลองของ Controller.....	27
ตารางที่ 4.10 แม่พิมพ์ต้นแบบและแบบจำลองของ Scope .....	28
ตารางที่ 4.11 แม่พิมพ์ต้นแบบและแบบจำลองของ Template.....	29
ตารางที่ 4.12 แม่พิมพ์ต้นแบบและแบบจำลองของ MarkupLanguage.....	30
ตารางที่ 4.13 แม่พิมพ์ต้นแบบและแบบจำลองของ Directive.....	31
ตารางที่ 4.14 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveController .....	33
ตารางที่ 4.15 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectivesolateScope .....	34
ตารางที่ 4.16 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveScope .....	35
ตารางที่ 4.17 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveTemplate .....	36
ตารางที่ 4.18 แบบสำรวจรายการสำหรับตรวจสอบความครบถ้วนของกฎการแปลง .....	36
ตารางที่ 5.1 การเพิ่มเติมโค้ดเทมเพลตให้เป็นโค้ดที่สามารถทำงานได้ .....	51
ตารางที่ 6.1 ผลการประเมินการแปลงแบบจำลองเป็นโค้ด .....	63
ตารางที่ ก. 1 คำอธิบายยูสเคสการนำเข้าแบบจำลองยูเอ็มแอลโปรไฟล์ในรูปแบบไฟล์ XMI .....	69
ตารางที่ ก. 2 คำอธิบายยูสเคสการเลือกโพลเดอร์สำหรับจัดเก็บโค้ด.....	70
ตารางที่ ก. 3 คำอธิบายยูสเคสการกำหนดไลบรารีของแองกูลาร์เจเอส .....	71
ตารางที่ ก. 4 คำอธิบายยูสเคสการแปลงแบบจำลองเป็นโค้ด.....	72
ตารางที่ ก. 5 คำอธิบายยูสเคสการดูโค้ดหลังจากแปลงแบบจำลองเสร็จ .....	73
ตารางที่ ก. 6 คำอธิบายยูสเคสการดูโพลเดอร์ของโค้ดที่ได้จากการแปลงแบบจำลอง .....	74

## สารบัญภาพ

### หน้า

ภาพที่ 2.1	ขั้นตอนหลักในกระบวนการพัฒนาแบบเอ็มดีเอ .....	4
ภาพที่ 2.2	ตัวอย่างส่วนประกอบของยูเอ็มแอลโปรไฟล์ .....	6
ภาพที่ 2.3	แองกูลาร์เจเอส (AngularJS).....	7
ภาพที่ 3.1	แนวคิดการพัฒนาเครื่องมือสำหรับแปลงแบบจำลองเฟรมเวิร์กแองกูลาร์เจเอสไปเป็นโค้ด .....	12
ภาพที่ 3.2	เมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส .....	13
ภาพที่ 3.3	แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมโคเลนต์.....	16
ภาพที่ 3.4	แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์.....	17
ภาพที่ 4.1	การแปลงป้ายระบุหรือคุณสมบัติของแบบจำลองเป็นโค้ด .....	19
ภาพที่ 4.2	กฎการแปลงป้ายระบุแบบจำลอง WebPage เป็นโค้ด.....	20
ภาพที่ 4.3	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง WebPage ไปใช้ .....	20
ภาพที่ 4.4	กฎการแปลงป้ายระบุแบบจำลอง ScriptLanguage เป็นโค้ด .....	21
ภาพที่ 4.5	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง ScriptLanguage ไปใช้.....	21
ภาพที่ 4.6	กฎการแปลงป้ายระบุแบบจำลอง CSS เป็นโค้ด.....	22
ภาพที่ 4.7	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง CSS ไปใช้ .....	22
ภาพที่ 4.8	กฎการแปลงป้ายระบุแบบจำลอง AngularApp เป็นโค้ด .....	22
ภาพที่ 4.9	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง AngularApp ไปใช้ .....	23
ภาพที่ 4.10	กฎการแปลงป้ายระบุแบบจำลอง Filter เป็นโค้ด .....	23
ภาพที่ 4.11	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Filter ไปใช้.....	24
ภาพที่ 4.12	กฎการแปลงป้ายระบุแบบจำลอง Service เป็นโค้ด .....	25
ภาพที่ 4.13	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Service ไปใช้.....	25
ภาพที่ 4.14	กฎการแปลงป้ายระบุแบบจำลอง AppRouter เป็นโค้ด.....	25
ภาพที่ 4.15	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง AppRouter ไปใช้.....	26
ภาพที่ 4.16	กฎการแปลงป้ายระบุแบบจำลอง AppView เป็นโค้ด.....	26
ภาพที่ 4.17	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง AppView ไปใช้.....	27
ภาพที่ 4.18	กฎการแปลงป้ายระบุแบบจำลอง Controller เป็นโค้ด .....	28
ภาพที่ 4.19	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Controller ไปใช้.....	28
ภาพที่ 4.20	กฎการแปลงป้ายระบุแบบจำลอง Scope เป็นโค้ด .....	29
ภาพที่ 4.21	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Scope ไปใช้.....	29
ภาพที่ 4.22	ไฟล์เทมเพลตหลังจากผ่านการนำแบบจำลอง Template ไปใช้.....	30
ภาพที่ 4.23	กฎการแปลงป้ายระบุแบบจำลอง MarkupLanguage เป็นโค้ด .....	31
ภาพที่ 4.24	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง MarkupLanguage ไปใช้ .....	31
ภาพที่ 4.25	กฎการแปลงป้ายระบุแบบจำลอง Directive เป็นโค้ด .....	32
ภาพที่ 4.26	โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Directive ไปใช้ .....	32

## สารบัญภาพ (ต่อ)

### หน้า

ภาพที่ 4.27 กฎการแปลงป้ายระบุแบบจำลอง DirectiveController เป็นโค้ด.....	33
ภาพที่ 4.28 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง DirectiveController ไปใช้.....	33
ภาพที่ 4.29 กฎการแปลงป้ายระบุแบบจำลอง DirectivelsolateScope เป็นโค้ด.....	34
ภาพที่ 4.30 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง DirectivelsolateScope ไปใช้.....	34
ภาพที่ 4.31 กฎการแปลงป้ายระบุแบบจำลอง DirectiveScope เป็นโค้ด.....	35
ภาพที่ 4.32 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง DirectiveScope ไปใช้.....	35
ภาพที่ 4.33 ไฟล์เทมเพลตหลังจากผ่านการนำแบบจำลอง DirectiveTemplate ไปใช้.....	36
ภาพที่ 5.1 ภาพรวมแนวคิดการทำงานของเครื่องมือ .....	39
ภาพที่ 5.2 แผนภาพยูสเคสของเครื่องมือสำหรับการแปลงแบบจำลองเป็นโค้ด .....	40
ภาพที่ 5.3 แผนภาพคลาสของเครื่องมือสำหรับการแปลงแบบจำลองเป็นโค้ด .....	42
ภาพที่ 5.4 ลำดับการทำงานของคลาสต่าง ๆ ของเครื่องมือ.....	43
ภาพที่ 5.5 ลำดับการทำงานของคลาสต่าง ๆ ของเครื่องมือ (ต่อ) .....	43
ภาพที่ 5.6 ลำดับการทำงานของคลาสต่าง ๆ ของเครื่องมือ (ต่อ) .....	43
ภาพที่ 5.7 Tag ต่าง ในไฟล์ XMI .....	45
ภาพที่ 5.8 Tag ต่าง ในไฟล์ XMI (ต่อ).....	45
ภาพที่ 5.9 การออกแบบส่วนต่อประสานของเครื่องมือ.....	45
ภาพที่ 5.10 การระบุข้อมูลนำเข้า.....	48
ภาพที่ 5.11 การแสดงผลเมื่อการทำงานของเครื่องมือเสร็จสิ้น .....	49
ภาพที่ 5.12 การแสดงผลลัพธ์ของส่วนเปิดคู่มือผลลัพธ์ของเครื่องมือ .....	50
ภาพที่ 5.13 การแสดงผลลัพธ์ของการดูโค้ดหลังจากแปลงเสร็จ .....	50
ภาพที่ 5.14 การแสดงผลลัพธ์ของการเปิดไฟล์เดอร์ที่เก็บโค้ด .....	51
ภาพที่ 5.15 การเลือกเปิด Command ของ Window .....	52
ภาพที่ 5.16 การลง Package ของ NPM.....	52
ภาพที่ 5.17 การเริ่มทำงานของเว็บเซิร์ฟเวอร์จำลอง .....	52
ภาพที่ 5.18 เว็บแอปพลิเคชันถูกเปิดโดยอัตโนมัติ.....	53
ภาพที่ 6.1 แผนภาพยูสเคสของระบบข้อมูลพื้นฐานของหุ่นออนไลน์.....	54
ภาพที่ 6.2 ตัวอย่างภาพหน้าจอของระบบข้อมูลพื้นฐานของหุ่นออนไลน์ .....	55
ภาพที่ 6.3 ยูเอ็มแอลโปรไฟล์ของแองกูลาร์เจเอสจากโปรแกรม MagicDraw .....	56
ภาพที่ 6.4 แบบจำลองแผนภาพคลาสของกรณีศึกษาจากโปรแกรม Magicdraw .....	57
ภาพที่ 6.5 แบบจำลองแผนภาพคลาสส่วนโครงสร้างหลักของแองกูลาร์เจเอสเว็บแอปพลิเคชัน.....	57
ภาพที่ 6.6 แบบจำลองแผนภาพคลาสส่วนโครงสร้างของแองกูลาร์เจเอส View.....	58
ภาพที่ 6.7 แบบจำลองแผนภาพคลาสส่วนโครงสร้างของแองกูลาร์เจเอส Directive .....	58
ภาพที่ 6.8 แบบจำลองแผนภาพคลาสส่วนโครงสร้างของ MarkupLanguage.....	59
ภาพที่ 6.9 แบบจำลองแผนภาพคลาสส่วน Header ของ Table.....	60

ภาพที่ 6.10 แบบจำลองแผนภาพคลาสส่วน Body ของ Table.....	61
ภาพที่ 6.11 แบบจำลองแผนภาพคลาสส่วนแถวสุดท้ายของ Table .....	62

# บทที่ 1

## บทนำ

### 1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันความซับซ้อนของระบบซอฟต์แวร์มีเพิ่มมากขึ้น ดังนั้นจึงต้องใช้ระยะเวลาในการพัฒนาระบบซอฟต์แวร์ที่มากขึ้น รวมถึงมีโอกาที่ทำให้เกิดข้อผิดพลาดจากการดำเนินการสูงขึ้น เช่นกัน ประกอบกับปัจจุบันการแข่งขันทางธุรกิจมีความรุนแรงสูงมาก ดังนั้นในแต่ละองค์กรต่างก็ต้องการมองแนวทางในการลดระยะเวลาในการพัฒนาซอฟต์แวร์ลง และส่งมอบงานที่มีคุณภาพให้กับลูกค้า เพื่อเพิ่มโอกาสทางการแข่งขันให้กับองค์กร

จากปัญหาที่กล่าวมาแล้วข้างต้นนั้นทาง Object Management Group (OMG) [1] ซึ่งเป็นองค์กรที่สร้าง มาตรฐาน (Standard) ต่าง ๆ ออกมาและเป็นองค์กรที่ไม่แสวงหาผลกำไร ได้กำหนดมาตรฐานหนึ่งที่เรียกว่า สถาปัตยกรรมอิงแบบจำลอง (Model Driven Architecture: MDA) [2] ซึ่งเป็นมาตรฐานที่ต่อยอดมาจากยูเอ็มแอล (Unified Modeling Language: UML) [3] โดย MDA นั้นจะมีการแปลง Model เป็นหลายระดับโดยเริ่มจาก Model ที่มุ่งเน้นที่ฟังก์ชันทางธุรกิจโดยไม่ขึ้นกับแพลตฟอร์ม เรียกว่า Platform-Independent Model (PIM) ถัดมาเมื่อได้ PIM ออกมา ก็ต้องมีการแปลง PIM ที่ได้นั้นให้เฉพาะเจาะจงกับแพลตฟอร์มที่จะนำไปพัฒนาต่อไปเรียกว่า Platform Specific Model (PSM) ซึ่งในที่นี้จะพิจารณา PSM เป็นหลัก จากหัวใจหลักของทาง MDA นั้นคือการแปลงแบบจำลอง (Model Transformation) ดังนั้นวิศวกรซอฟต์แวร์สามารถแปลง PSM ออกมาให้เป็นโค้ด (Code) ได้โดยการสร้างกฎการแปลง (Transformation Rules) ขึ้นมาและใช้เครื่องมือสำหรับการแปลง (Transformation Tool) ที่รวมกฎการแปลงอยู่นั้น ทำการแปลง PSM ออกมาเป็นโค้ดแบบอัตโนมัติที่เฉพาะเจาะจงกับแพลตฟอร์ม แต่การแปลงเป็นโค้ดจะได้ผลลัพธ์คือโค้ดเทมเพลต (Code Template) ซึ่งผู้พัฒนายังต้องเติมตรรกะทางธุรกิจ (Business Logic) เพื่อให้โปรแกรมทำงานได้ จากการที่ได้โค้ดเทมเพลตมาทำให้สามารถลดระยะเวลาในการเขียนโค้ดลงได้ เมื่อเทียบกับการให้ผู้พัฒนาเขียนโค้ดเองทั้งหมดและผลที่ตามมาทำให้สามารถลดข้อผิดพลาดได้

ในปัจจุบันบริษัทต่าง ๆ ได้มีการสร้างเว็บแอปพลิเคชันขึ้นมาไม่ว่าจะเป็นการสร้างเพื่อนำเสนอข้อมูลของทางบริษัทเองหรือสร้างเพื่อให้ลูกค้าใช้งาน และเทคโนโลยีของการสร้างเว็บแอปพลิเคชันนั้นก็ได้มีการพัฒนาไปมาจากแต่ก่อน เพื่อที่จะทำให้เว็บแอปพลิเคชันมีความใกล้เคียงกับเดสก์ท็อปแอปพลิเคชัน ที่ผู้ใช้งานมีความคุ้นเคยและมีการตอบสนองที่รวดเร็ว ในปัจจุบันนี้มีเทคนิคในการสร้างเว็บแอปพลิเคชันที่เป็นแบบซิงเกิลเพจแอปพลิเคชัน (Single Page Application) [4] เพื่อตอบสนองความต้องการดังที่กล่าวไปแล้ว สำหรับเว็บแอปพลิเคชันแบบซิงเกิลเพจแอปพลิเคชันนั้นการทำงานต่าง ๆ นั้นจะอยู่ที่ฝั่งไคลเอนต์ (Client) ซึ่งผลที่ตามมาคือ ทำให้การใช้งานเว็บแอปพลิเคชันนั้นรวดเร็วขึ้นมาก และหนึ่งในเฟรมเวิร์กที่ได้รับความนิยมในการสร้างเว็บแอปพลิเคชันแบบซิงเกิล

เฟรมแอปพลิเคชันนั้น คือ แองกูลาร์เจส (AngularJS) [5] [10] [11] ซึ่งพัฒนาโดยทาง Google และในปัจจุบันมีการใช้งานโดยบริษัทที่หลากหลาย [5]

จากที่ได้กล่าวมาแล้วข้างต้น หากทางบริษัทที่พัฒนาเว็บแอปพลิเคชันด้วยเฟรมเวิร์กแองกูลาร์เจสได้นำหลักการพัฒนาแบบอิงแบบจำลองหรือ MDA ไปใช้งาน จะทำให้การพัฒนาเว็บแอปพลิเคชันนั้นมีความรวดเร็วในการพัฒนามากยิ่งขึ้น ผู้วิจัยจึงมีแนวคิดในการพัฒนาการออกแบบเว็บแอปพลิเคชันโดยใช้ยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจส ในการสร้างแบบจำลองระดับ PSM ของแอปพลิเคชัน และกำหนดกฎการแปลงและพัฒนาเครื่องมือเพื่อใช้ในการแปลงแบบจำลองระดับ PSM ไปเป็นโค้ดเทมเพลตสำหรับแองกูลาร์เจส เพื่อช่วยลดระยะเวลาและข้อผิดพลาดในการพัฒนา

## 1.2 วัตถุประสงค์ของโครงการ

- 1) เพื่อพัฒนายูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจส
- 2) เพื่อพัฒนากฎการแปลงแบบจำลองไปเป็นโค้ดเทมเพลตจากยูเอ็มแอลโปรไฟล์ที่สร้างขึ้น
- 3) เพื่อพัฒนาเครื่องมือช่วยในการแปลงแบบจำลองเป็นโค้ดเทมเพลตของเว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจส

## 1.3 ขอบเขตของโครงการ

- 1) สร้างยูเอ็มแอลโปรไฟล์ประกอบด้วย สถาปัตยกรรมไคลเอนต์ด้วยยูเอ็มแอล และสถาปัตยกรรมความสัมพันธ์ด้วยยูเอ็มแอล โดยแสดงในรูปแผนภาพซึ่งประกอบด้วยแม่พิมพ์ต้นแบบและนิยาม پایระบุที่เกี่ยวข้อง (ถ้ามี)
- 2) สร้างกฎการแปลงซึ่งประกอบด้วยกฎสำหรับแปลงแบบจำลองที่ออกแบบโดยใช้แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์ และแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์ โดยแสดงในรูปเค้าร่างของกฎตามแนวทางของงานวิจัย [12]
- 3) สร้างเครื่องมือสำหรับแปลงแบบจำลองที่ออกแบบโดยยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชัน บนเฟรมเวิร์กแองกูลาร์เจส โดยเครื่องมือจะถูกพัฒนาด้วยภาษาซีชาร์ป (C#) และมีฟังก์ชันการทำงานดังนี้เป็นอย่างน้อย
  - กำหนดไฟล์ XMI (Locate XMI)
  - กำหนดไลบรารีสำหรับแองกูลาร์เจส (Locate AngularJS library)
  - เลือกไดเรกทอรีสำหรับจัดเก็บโค้ดหลังจากแปลงเสร็จ (Locate Code Folder)
  - แปลงแบบจำลองเป็นโค้ด (Generate Code)
  - ดูโค้ดที่ได้จากการแปลงได้ (View Generated Code)
- 4) เว็บแอปพลิเคชันที่ได้รับจากการแปลงโดยใช้เครื่องมือที่พัฒนาขึ้นเป็นเพียงโค้ดเทมเพลตเท่านั้น
- 5) ทดสอบความถูกต้องของเครื่องมือโดยเครื่องมือต้องสามารถแปลงแบบจำลองเป็นโค้ดเทมเพลตได้อย่างถูกต้องตามกฎการแปลงที่ได้กำหนดไว้ โดยนำมาประยุกต์ใช้กับกรณีศึกษา

ระบบข้อมูลพื้นฐานของหุ่นออนไลน์ โดยจำนวนของกรณีศึกษาต้องมีความครอบคลุมครบทุกกฎการแปลงที่ได้กำหนดไว้

- 6) ประเมินผลการทำงานของเครื่องมือโดยใช้อัตราการแปลง ซึ่งคำนวณจากจำนวนบรรทัดของโค้ด เทมเพลตต่อจำนวนบรรทัดของโค้ดที่เสร็จสมบูรณ์ของกรณีศึกษา โดยโค้ดที่เสร็จสมบูรณ์นั้นจะนับจำนวนบรรทัดของโค้ดโดยใช้ค่าเฉลี่ยจากนักพัฒนา

#### 1.4 ขั้นตอนและวิธีการดำเนินโครงการ

- 1) ศึกษาและนิยามการสร้างยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจเอส และงานวิจัยที่เกี่ยวข้อง
- 2) นิยามกฎการแปลงแบบจำลองไปเป็นโค้ดสำหรับเฟรมเวิร์กแองกูลาร์เจเอส
- 3) เลือกเครื่องมือสำหรับออกแบบระบบที่สามารถปรับแต่งยูเอ็มแอลโปรไฟล์ในการสร้างแบบจำลองและสามารถส่งออกแผนภาพแบบจำลองเป็นไฟล์ XMI ได้
- 4) ออกแบบและนิยามยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจเอส
- 5) พัฒนาเครื่องมือสำหรับการแปลงยูเอ็มแอลโปรไฟล์ไปเป็นโค้ดเทมเพลต
- 6) สร้างระบบกรณีศึกษาและประเมินผลเครื่องมือที่พัฒนาขึ้นเพื่อสนับสนุนแนวคิดในโครงการนี้
- 7) จัดทำบทความทางวิชาการและนำเสนอ
- 8) สรุปผลแนวทางการวิจัย ข้อเสนอแนะและจัดทำเล่มโครงการฉบับสมบูรณ์

#### 1.5 ประโยชน์ที่คาดว่าจะได้รับ

- 1) ได้ยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจเอส
- 2) ได้วิธีการแปลงแบบจำลองเป็นโค้ดแองกูลาร์เจเอสจากยูเอ็มแอลโปรไฟล์ที่ได้สร้างขึ้น
- 3) ได้เครื่องมือที่ช่วยในการแปลงแบบจำลองที่ได้จากยูเอ็มแอลโปรไฟล์ไปเป็นโค้ด ทำให้ลดระยะเวลาในการพัฒนาเว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจเอส

#### 1.6 ผลงานที่ตีพิมพ์จากโครงการมหาบัณฑิต

ส่วนหนึ่งของโครงการมหาบัณฑิตนี้ได้รับการตอบรับเพื่อตีพิมพ์เป็นบทความวิจัยในหัวข้อเรื่อง “A Model-Driven Development of Web Applications Using AngularJS Framework” โดย Wutthichai Chansuwath and Twittie Senivongse ในงานประชุมวิชาการระดับนานาชาติด้านคอมพิวเตอร์และวิทยาการสารสนเทศ ครั้งที่ 15 (15th IEEE/ACIS International Conference on Computer and Information Science: ICIS 2016) ซึ่งจัดขึ้นโดย IEEE Computer Society และ International Association for Computer and Information Science (ACIS) ณ เมืองโอokayama (Okayama) ประเทศญี่ปุ่น ระหว่างวันที่ 26 –29 มิถุนายน 2559



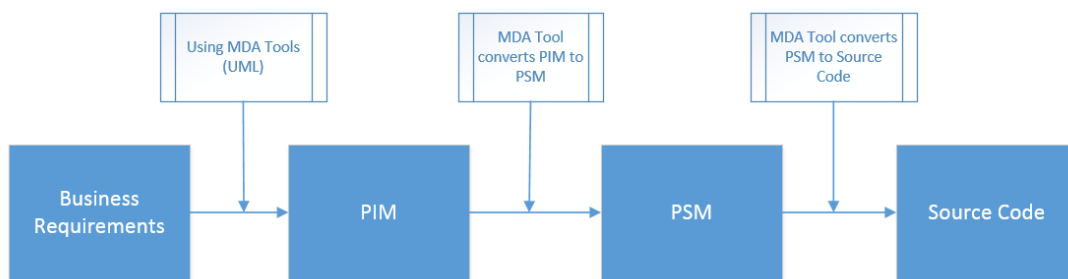
## บทที่ 2

### ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

#### 2.1 ทฤษฎีที่เกี่ยวข้อง

##### 2.1.1 สถาปัตยกรรมอิงแบบจำลอง

สถาปัตยกรรมอิงแบบจำลอง (Model Driven Architecture: MDA) [2] นั้นเป็นสถาปัตยกรรมรูปแบบหนึ่งที่ทางองค์กรไม่แสวงหาผลกำไร Object Management Group (OMG) ได้เผยแพร่ออกมาในปี พ.ศ.2544 เพื่อเป็นกรอบงานสำหรับใช้ในการพัฒนาซอฟต์แวร์ โดยมีหลักการคือ สร้างแบบจำลองขึ้นมาเพื่อเป็นตัวขับเคลื่อนกระบวนการในการพัฒนาซอฟต์แวร์ ซึ่งมีกระบวนการทำงานดังภาพที่ 2.1



ภาพที่ 2.1 ขั้นตอนหลักในกระบวนการพัฒนาแบบเอ็มดีเอ [6]

จากภาพที่ 2.1 นั้นจะเป็นการอธิบายกระบวนการทำงานของสถาปัตยกรรมอิงแบบจำลอง ซึ่งได้แบ่งขั้นตอนการทำงานออกเป็น 4 ส่วนหลัก

- ความต้องการทางธุรกิจ (Business Requirements) ซึ่งในขั้นตอนนี้คือ การระบุความต้องการของผู้ใช้งานซึ่งจะอยู่ในรูปแบบใด ๆ ก็ได้ตามที่ผู้ใช้งานต้องการ

- แบบจำลองที่อิสระจากแพลตฟอร์ม (Platform Independent Model: PIM) คือแบบจำลองที่มีการแปลงความต้องการของผู้ใช้งานให้อยู่ในรูปของแบบจำลองที่ไม่ขึ้นกับเฟรมเวิร์กหรือแพลตฟอร์ม เพื่อให้เห็นภาพรวมของทั้งระบบ โดยการแปลงความต้องการให้อยู่ในรูปแบบจำลองนั้นจะมีการใช้เครื่องมือเป็นตัวช่วยในสร้างแบบจำลองที่อิสระจากแพลตฟอร์ม เช่น การใช้ UML [3] เป็นเครื่องมือ

- แบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์ม (Platform Specific Model: PSM) คือแบบจำลองที่อ้างอิงกับเฟรมเวิร์กหรือแพลตฟอร์มที่เฉพาะเจาะจงกับผู้พัฒนาซอฟต์แวร์ โดยขั้นตอนการแปลงแบบจำลองที่อิสระจากแพลตฟอร์ม ให้เป็นแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์มนั้นต้องมีการทำผ่านเครื่องมือที่ได้มีการประกาศกฎการแปลง (Transformation Rules) ที่ชัดเจน เช่น UML Profile [7] [8]

- โค้ด (Source Code) คือ ผลลัพธ์ที่ได้หลังจากมีการแปลงแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์ม ตามกฎการแปลงที่ได้ประกาศไว้ ซึ่งการแปลงแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์มให้เป็นโค้ดได้นั้นต้องทำผ่านเครื่องมือที่มีการนำกฎการแปลง เข้าไปรวมอยู่ในเครื่องมือ นั้นด้วยเพื่อให้เครื่องมือชิ้นนั้นทำการแปลงแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์มให้กลายเป็น โค้ด ได้อย่างถูกต้องตรงกับเงื่อนไขของกฎการแปลง

### 2.1.2 ยูเอ็มแอล

ยูเอ็มแอล (Unified Modeling Language: UML) [3] นั้นเป็นสัญลักษณ์มาตรฐานที่ได้รับการออกแบบจาก Object Management Group (OMG) ในปี พ.ศ. 2540 เพื่อให้อธิบายและแสดงรายละเอียดของระบบซอฟต์แวร์ที่จะถูกพัฒนาขึ้นมา เพื่อให้ผู้ที่เกี่ยวข้องมีความเข้าใจ สอดคล้องกันทั้งกระบวนการของพัฒนา ทั้งผู้พัฒนาซอฟต์แวร์และผู้ออกแบบระบบรวมถึงผู้ที่เกี่ยวข้องในโครงการทั้งหมด ในปัจจุบันนี้ยูเอ็มแอลได้รับการปรับปรุงมาถึงรุ่น 2.5 ซึ่งได้รับการเผยแพร่ตั้งแต่เดือนมิถุนายน พ.ศ. 2558 โดยมีแผนภาพหลัก ๆ อยู่ 2 ประเภท คือ

- 1) แผนภาพโครงสร้าง (Structure Diagrams) ซึ่งเป็นแผนภาพที่มีลักษณะเป็นโครงสร้างที่ชัดเจน เช่น แผนภาพคลาส (Class Diagram)

- 2) แผนภาพพฤติกรรม (Behavior Diagrams) ซึ่งเป็นแผนภาพที่มีลักษณะแสดงความเคลื่อนไหวของสิ่งที่อยู่ในแผนภาพ เช่น แผนภาพกิจกรรม (Activity Diagram)

แผนภาพโครงสร้างนั้นทาง OMG ได้เปิดช่องทางให้สามารถเพิ่มเติมความสามารถโดยการปรับแต่งแผนภาพโครงสร้างให้มีความเฉพาะเจาะจงกับเฟรมเวิร์กหรือแพลตฟอร์มที่ต้องการได้ เรียกว่า ยูเอ็มแอลโพรไฟล์

### 2.1.3 ยูเอ็มแอลโพรไฟล์

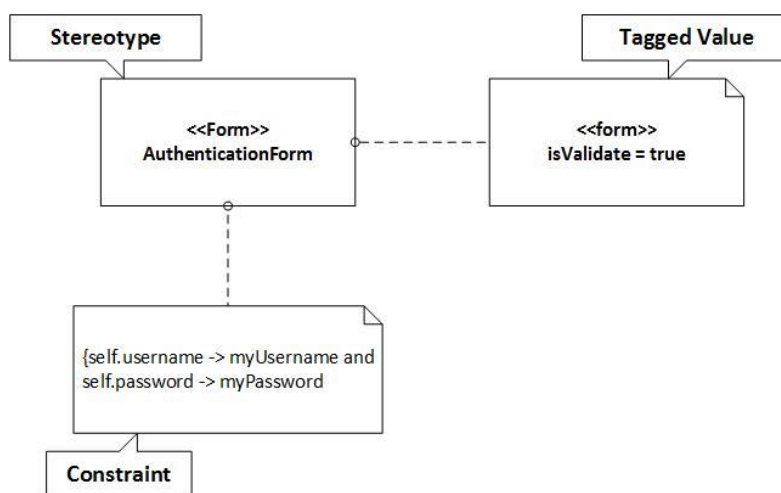
ยูเอ็มแอลโพรไฟล์ (UML Profile) [7] [8] นั้นเป็นการเพิ่มเติมความสามารถของยูเอ็มแอลให้สอดคล้องกับเฟรมเวิร์กหรือแพลตฟอร์มที่ได้ระบุไว้ในการพัฒนาซอฟต์แวร์ โดยผู้ใช้งานนั้นสามารถระบุกฎหรือเงื่อนไขสำหรับการสร้างแบบจำลองให้เหมาะสมกับเฟรมเวิร์กหรือแพลตฟอร์มที่ต้องการ โดยยูเอ็มแอลโพรไฟล์จะมีส่วนประกอบอยู่ 3 ส่วน ดังตัวอย่างในภาพที่ 2.2

- 1) แม่พิมพ์ต้นแบบ (Stereotype) หรือเรียกว่าโพรไฟล์คลาส (Profile Class) เป็นส่วนประกอบหนึ่งของแบบจำลองเพื่อรวมกลุ่มของข้อมูลที่มีลักษณะคล้าย ๆ กันเข้าไว้ด้วยกัน โดยแม่พิมพ์ต้นแบบนั้นไม่สามารถถูกสร้างต่อ (Extend) จากแม่พิมพ์ต้นแบบตัวอื่น ๆ ได้ แม่พิมพ์ต้นแบบนั้นจะใช้สัญลักษณ์ <<>> ในการแสดงผลโดยชื่อของแม่พิมพ์ต้นแบบจะอยู่ตรงกลางระหว่างสัญลักษณ์ <<>> เช่น <<Form>> มีความหมายว่าเป็นหน้าฟอร์มของหน้าเว็บ เป็นต้น

- 2) นิยามป้ายระบุ (Tag Definition) และ ค่าป้ายระบุ (Tagged Value) นั้นจะใช้กำหนดค่าของคุณลักษณะเฉพาะ (Properties) ของแม่พิมพ์ต้นแบบเมื่อมีการนำไปใช้ เช่น ป้ายระบุ isValidate ถูกกำหนดให้มีค่าเป็น true เมื่อนำแม่พิมพ์ต้นแบบ <<Form>> ไปใช้ในการสร้าง

แบบจำลองสำหรับ AuthenticationForm ซึ่งค่าป้อนระบุดังกล่าวแสดงถึงการมีการตรวจสอบฟอร์มของการระบุตัวตนของผู้ที่จะเข้าใช้งาน (Authentication)

3) เงื่อนไขบังคับ (Constraint) เป็นข้อกำหนดที่ระบุถึงเงื่อนไขบังคับการใช้งานแม่พิมพ์ต้นแบบและค่าป้อนระบุ ซึ่งปกติโดยทั่วไปเงื่อนไขบังคับจะใช้ภาษาที่ชื่อว่าโอซีแอล (Object Constraint Language: OCL) [9] ซึ่งถูกประกาศใช้ใน OMG เช่นเดียวกัน โดยจากตัวอย่างภาพที่ 2 เงื่อนไขบังคับนั้น คือ {self.username -> myUsername and self.password -> myPassword} หมายความว่า username ต้องมีค่าเป็น myUsername และ password ต้องมีค่าเป็น myPassword



ภาพที่ 2.2 ตัวอย่างส่วนประกอบของยูเอ็มแอลโปรไฟล์

### 2.1.4 แองกูลาร์เจเอส

แองกูลาร์เจเอส (AngularJS) [10] [11] เป็นจาวาสคริปต์เฟรมเวิร์กประเภทหนึ่งซึ่งเป็นที่นิยม และมีการถูกนำไปใช้งานในหลาย ๆ เว็บแอปพลิเคชัน [5] และสามารถทำเว็บแอปพลิเคชันส่วนหน้า (Front-End Web Application) ได้อย่างสะดวกรวดเร็ว แองกูลาร์เจเอสนั้นจะมีลักษณะที่สำคัญ (Features) คือ

1) การผูกข้อมูลแบบสองทาง (Two Way Data Binding) คือ กระบวนการทำงานหนึ่งในแองกูลาร์เจเอสเฟรมเวิร์ก ที่ถ้าข้อมูลมีการเปลี่ยนแปลงเมื่อใดส่วนการแสดงผลที่ถูกผูกไว้กับข้อมูลนั้น จะถูกเปลี่ยนไปให้ตรงกับข้อมูลที่ได้เปลี่ยนไปโดยอัตโนมัติ

2) เทมเพลต (Templates) คือ กลุ่มของภาษาเอชทีเอ็มแอล (HyperText Markup Language: HTML) ที่ถูกแปลงให้กลายเป็นดอม (Document Object Model: DOM) ซึ่งก็คือโครงสร้างของหน้าเว็บแอปพลิเคชันซึ่ง แองกูลาร์เจเอสเฟรมเวิร์กนั้น สามารถควบคุมเทมเพลตให้ทำงานตามที่คุณพัฒนาต้องการได้

3) เอ็มวีซี (Model-View-Controller: MVC) คือ สถาปัตยกรรมรูปแบบหนึ่งที่แยกการทำงานออกเป็น 3 ส่วนหลัก เพื่อให้ให้นักพัฒนาสามารถแยกการทำงานเป็นอิสระออกจากกัน โดยมี

การแบ่งได้เป็น 1) วิว (Views) สำหรับจัดการส่วนแสดงผล 2) แบบจำลอง (Model) สำหรับเก็บข้อมูลที่จะใช้แสดงบนวิว และ 3) ส่วนควบคุม (Controller) ซึ่งมีไว้จัดการข้อมูลให้เป็นไปตามที่นักพัฒนาต้องการ

4) ดีเพนเดนซีอินเจกชัน (Dependency Injection) คือ การเพิ่มเซอร์วิส (Services) ที่เกี่ยวข้องกับการทำงานของแองกูลาร์เจเอสเฟรมเวิร์ก เพื่อให้ทางแองกูลาร์เจเอสเฟรมเวิร์กทำการโหลดเซอร์วิส ให้เสร็จสิ้นก่อนที่จะทำงานส่วนหลักของไดเรกทีฟ

5) ไดเรกทีฟ (Directives) คือ กลุ่มของเทมเพลตที่ทำงานได้ตามที่นักพัฒนาต้องการ โดยมีการเพิ่มโค้ดเพื่อควบคุมดอม ให้แสดงผลตามที่นักพัฒนาต้องการ



ภาพที่ 2.3 แองกูลาร์เจเอส (AngularJS) [11]

### 2.1.5 การพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส

สำหรับโครงการนี้ได้เสนอการพัฒนาเว็บแอปพลิเคชัน โดยใช้แองกูลาร์เจเอสเฟรมเวิร์ก ซึ่งเป็นเฟรมเวิร์กสำหรับสร้างเว็บแอปพลิเคชันส่วนหน้า ซึ่งหลังจากที่หน้าเว็บถูกโหลดขึ้นมานั้น แองกูลาร์เจเอสเฟรมเวิร์กจะทำการแ่งส่วน (Parse) ดอมและแฟ้มจาวาสคริปต์ (Javascript Files) ที่เกี่ยวข้องทั้งหมดให้อยู่ในภาพที่ทางแองกูลาร์เจเอสเฟรมเวิร์กเข้าใจ ก่อนที่จะทำการแสดงผลให้ผู้ใช้งานเห็น

## 2.2 งานวิจัยที่เกี่ยวข้อง

ผู้วิจัยได้ทำการค้นคว้างานวิจัยเพิ่มเติมที่เกี่ยวข้องกับงานวิจัยที่จะพัฒนาขึ้นนี้ โดยสามารถอธิบายได้ตามหัวข้อย่อยดังต่อไปนี้

### 2.2.1 Transformation from Web PSM to Code [13]

งานวิจัยได้เสนอการแปลงแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์มไปเป็นโค้ดเทมเพลต ซึ่งในงานวิจัยฉบับนี้จะใช้แผนภาพคลาสเป็นตัวแทนแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์ม และมีการแปลงแบบจำลองไปเป็นโค้ดเทมเพลตในรูปแบบของเอ็มวีซี โดยงานวิจัยนี้ได้ใช้ภาษาในการพัฒนาคือ JSP, Servlets และ JAVA โดยใช้ JAVA เป็นภาษาสำหรับการพัฒนาเครื่องมือในการแปลงแบบจำลองไปเป็นโค้ดเทมเพลต

จากแนวคิดของสถาปัตยกรรมอิงแบบจำลองนั้นจะเริ่มมาจากการออกแบบแผนภาพคลาสซึ่งมีการขยายยูเอ็มแอลเพื่อสร้างแม่พิมพ์ต้นแบบ (Stereotypes) และความสัมพันธ์ของ

แม่พิมพ์ต้นแบบ (Association Stereotypes) เพื่อให้ได้เป็นแผนภาพคลาสในระดับ PSM หรือแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์ม จากนั้นทีมนักวิจัยก็ได้สร้างเครื่องมือเพื่อแปลงแบบจำลองไปเป็นโค้ดเทมเพลต ส่วนการประเมินนั้นทางทีมนักวิจัยได้เลือกวิธีการวัดซอฟต์แวร์ที่ประยุกต์มาจากกระบวนการทดสอบซอฟต์แวร์ (Software Testing) ที่เรียกว่าความครอบคลุมของโค้ด (Code Coverage) ด้วยการเทียบอัตราส่วนจำนวนบรรทัดของโค้ด (Line of Code: LOC) เพราะว่าการวัดทำได้ง่ายและสามารถใช้คำนวณแรงงาน (Effort) ที่ถูกนำมาใช้ได้จากจำนวนบรรทัดของโค้ดที่ได้วัดมา ซึ่งงานวิจัยนี้ได้ทำการวัดจำนวนบรรทัดของโค้ดเทมเพลตเปรียบเทียบกับจำนวนบรรทัดของโค้ดที่เสร็จสมบูรณ์ พบว่าอัตราการแปลง (Transformation Rate) อยู่ที่ประมาณ 36–55%

ผู้วิจัยในโครงการมหาบัณฑิต จะนำแนวทางการประเมินประสิทธิภาพการแปลงของงานวิจัยนี้มาปรับใช้ ในส่วนการเปรียบเทียบจำนวนบรรทัดของโค้ดเทมเพลตกับจำนวนบรรทัดของโค้ดที่เสร็จสมบูรณ์

### 2.2.2 A component-centric UML based approach for modeling the architecture of web applications [14]

งานวิจัยนี้ได้กล่าวถึงการจัดหมวดหมู่ (Categorize) ของส่วนประกอบของเว็บแอปพลิเคชัน (Web Applications Component) และความสัมพันธ์ระหว่างส่วนประกอบของเว็บแอปพลิเคชัน โดยการขยายยูเอ็มแอลเพื่อระบุส่วนประกอบของเว็บแอปพลิเคชันโดยใช้แม่พิมพ์ต้นแบบ (Stereotype) โดยงานวิจัยนี้ได้ใช้เทคโนโลยี HTML, ASP, JSP, PHP, Servlet และ JavaBean ในการสร้างเว็บแอปพลิเคชัน งานวิจัยนี้ได้ทำการขยายยูเอ็มแอล โดยมีลำดับขั้นตอนดังต่อไปนี้

- 1) สร้างเมตาโมเดลสำหรับเว็บแอปพลิเคชัน ซึ่งเว็บแอปพลิเคชันหนึ่ง ๆ นั้นจะมีส่วนประกอบอยู่หลายส่วน เช่น ฟอรัม ส่วนประกอบการนำเสนอ หน้าเว็บแบบคงที่ หน้าเว็บแบบไดนามิก และการเชื่อมโยงหลายมิติ เป็นต้น ซึ่งส่วนประกอบเหล่านี้เป็นแบบนามธรรมที่ดึงออกมาจากการพัฒนาเว็บแอปพลิเคชัน การนิยามของส่วนประกอบที่เป็นนามธรรมเหล่านี้จะกำหนดเป็นเมตาโมเดลสำหรับเว็บแอปพลิเคชัน

- 2) กำหนดแม่พิมพ์ต้นแบบสำหรับส่วนประกอบของเว็บแอปพลิเคชัน ซึ่งเป็นขั้นตอนการกำหนดแม่พิมพ์ต้นแบบสำหรับแต่ละส่วนประกอบของเมตาโมเดล

- 3) ส่วนขยายของส่วนประกอบแบบจำลองยูเอ็มแอล เป็นการระบุส่วนประกอบของยูเอ็มแอลซึ่ง ได้แก่ คลาส (Class) คอมโพเนนต์ (Component) แพคเกจ (Package) และความสัมพันธ์ (Association) ระหว่างการสร้างยูเอ็มแอลโปรไฟล์ โดยการกำหนดแม่พิมพ์ต้นแบบสำหรับส่วนประกอบของไคลเอนต์ เซิร์ฟเวอร์ และความสัมพันธ์ของเว็บแอปพลิเคชัน

ผู้วิจัยในโครงการนมหำบัณฑิต จะนำแนวทางการสร้างยูเอ็มแอลโปรไฟล์ของงานวิจัยนี้ มาปรับปรุงเพื่อให้สอดคล้องกับการพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสต่อไป

### 2.2.3 Visual Modeling for Web 2.0 Applications Using Model Driven Architecture Approach [15]

งานวิจัยนี้มีการกล่าวถึงเว็บ 2.0 แมชอัป (Web 2.0 Mashup) [16] ซึ่งได้แก่ เว็บแอปพลิเคชันที่ได้นำการทำงานโดยรวมข้อมูล (Data) และบริการ (Service) จากภายนอกเข้าไว้ด้วยกัน เพื่อเพิ่มความสามารถให้แก่เว็บแอปพลิเคชัน โดยในงานวิจัยนี้ได้นำเสนอการใช้อยูเอ็มแอลโปรไฟล์และนำสถาปัตยกรรมอิงแบบจำลองมาใช้งานซึ่งเรียกว่า Web2.0MUML โดยจะมี 4 ระดับดังนี้

- 1) การนิยามแบบจำลองของเว็บ 2.0 โดยใช้แผนภาพยูเอ็มแอล
- 2) การนิยามยูเอ็มแอลโปรไฟล์ของเว็บ 2.0 ซึ่งมีการใช้แบบจำลองที่อิสระจากแพลตฟอร์ม โดยการขยายยูเอ็มแอลโดยใช้แม่พิมพ์ต้นแบบ นิยามป้ายระบุ และเงื่อนไขบังคับ ให้สอดคล้องกับโครงสร้างและการทำงานของเว็บ 2.0 ซึ่งประกอบด้วยสถาปัตยกรรมของ เว็บแอปพลิเคชัน 2.0, Web resource, Client และ Relationship
- 3) การนำยูเอ็มแอลโปรไฟล์ของ Web2.0MUML มาใช้สร้างแบบจำลองของระบบการค้นหารถทางหลวงเชิงแผนที่ (Map-Based Highway Traffic Query System: MHTQ)
- 4) นำแบบจำลองของระบบการค้นหารถทางหลวงเชิงแผนที่ มาแปลงเป็นโค้ดโดยใช้ XSLT Style Sheet ทำการแปลงเป็นเอกสาร HTML, KML และ RSS

จากการประเมินผลของยูเอ็มแอลโปรไฟล์ของเว็บ 2.0 ของงานวิจัยนี้จะมีอยู่ 2 ด้านคือ การประเมินความผันแปร (Heterogeneous Evaluation) และการประเมินสมรรถนะ (Performance Evaluation) ของการแปลงเอกสาร XMI [18] ไปเป็นในรูปแบบ HTML (23 โมดูล), KML (21 โมดูล) และ RSS (15 โมดูล) โดยได้ผลลัพธ์เวลาเฉลี่ยดังนี้ 17.8, 17.1 และ 14.3 มิลลิวินาทีตามลำดับ

ผู้วิจัยในโครงการนมหำบัณฑิต จะนำแนวทางการสร้างยูเอ็มแอลโปรไฟล์ของงานวิจัยนี้ มาปรับปรุงเพื่อให้สอดคล้องกับการพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสต่อไป รวมถึงนำเครื่องมือในการสร้างยูเอ็มแอลโปรไฟล์ที่ได้ผลลัพธ์ออกมาในรูปแบบเอกสาร XMI ซึ่งงานวิจัยนี้ใช้เพื่อนำมาเป็นส่วนนำเข้า (Input) ของเครื่องมือที่จะพัฒนาในโครงการ

### 2.2.4 A Model-Driven Development of Web-Based Applications on Google App Engine Platform [12]

งานวิจัยนี้เป็นงานวิจัยเกี่ยวกับการพัฒนาเว็บแอปพลิเคชันบนแพลตฟอร์มกูเกิลแอปเอนจินแบบอิงแบบจำลอง โดยกล่าวถึงการสร้างยูเอ็มแอลโปรไฟล์ตามแนวคิดของเอ็มดีเอ เพื่อแปลงให้ได้

แบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์ม และกำหนดกฎการแปลงเพื่อใช้ในการแปลงแบบจำลองไปเป็นโค้ดเทมเพลต งานวิจัยนี้ยังทำการประเมินประสิทธิภาพของการแปลงให้เป็นโค้ดเทมเพลต โดยการวัดสัดส่วนของการแปลงเป็นโค้ดเทมเพลตเทียบกับโค้ดที่สมบูรณ์ ซึ่งได้ผลประมาณ 45 – 79% ของโค้ดที่สมบูรณ์ในการวัดโดยใช้กรณีศึกษา

ผู้วิจัยในโครงการมหาบัณฑิต จะนำวิธีการประเมินของงานวิจัยนี้มาปรับใช้โดยจะทำการวัดสัดส่วนของการแปลงเป็นโค้ดเทมเพลตเทียบกับโค้ดที่สมบูรณ์และเทียบกับกรณีศึกษา รวมถึงแนวทางในการปรับแต่งยูเอ็มแอลโปรไฟล์ของงานวิจัยนี้มาปรับใช้กับการสร้างยูเอ็มแอลโปรไฟล์บนเฟรมเวิร์กแองกูลาร์เจเอสอีกด้วย

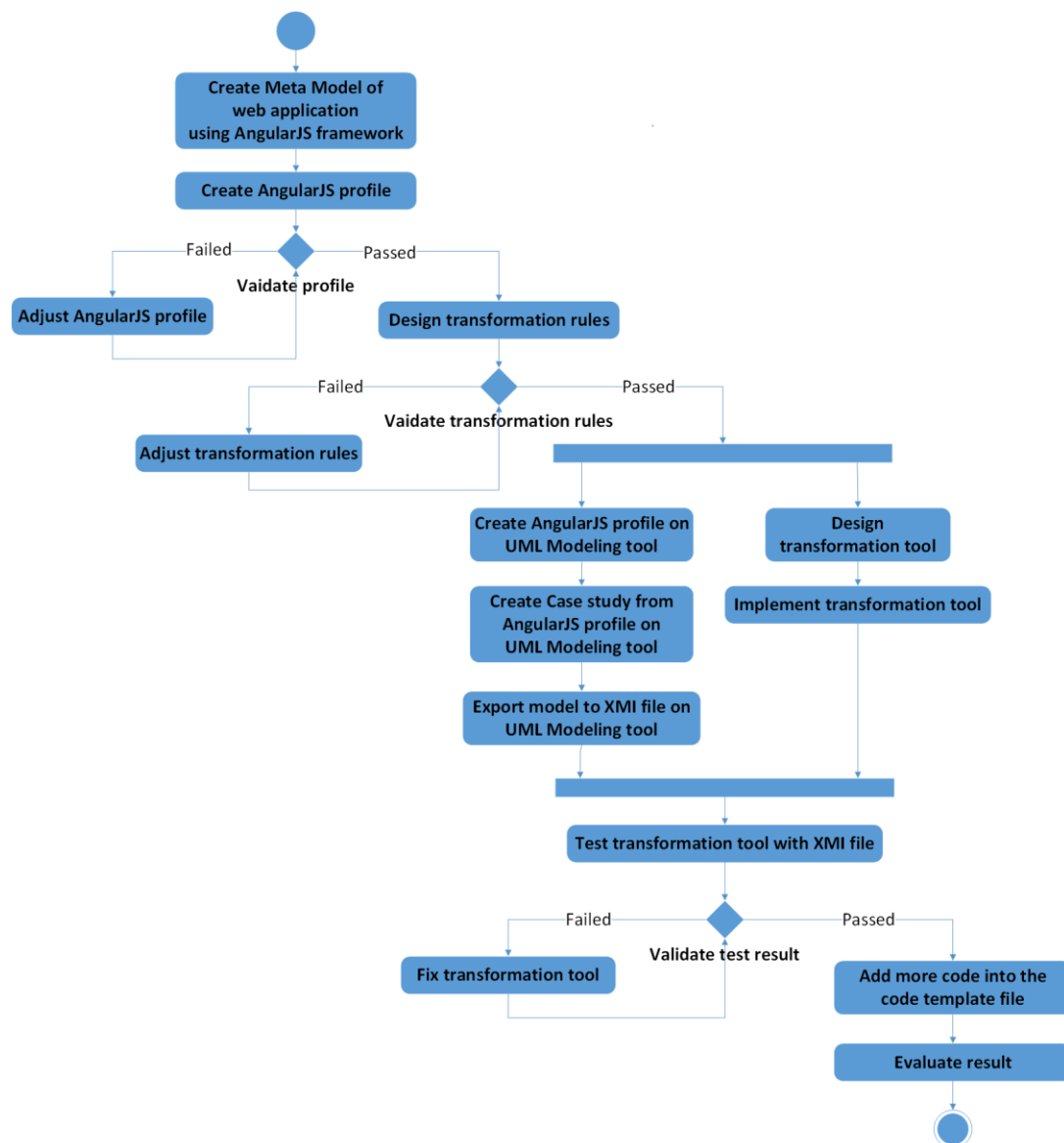
### บทที่ 3

#### ยูเอ็มแอลโปรไฟล์แองกูลาร์เจเอส

ในบทนี้จะเริ่มต้นจากการนำเสนอแนวคิดการพัฒนาเครื่องมือสำหรับแปลงแบบจำลองของแองกูลาร์เจเอสเฟรมเวิร์กไปเป็นโค้ด หลังจากนั้นจะอธิบายถึงการออกแบบยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส เริ่มด้วยการสร้างเมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส การสร้างโปรไฟล์แองกูลาร์เจเอส และการตรวจสอบโปรไฟล์แองกูลาร์เจเอส

แผนภาพกิจกรรมแสดงภาพรวมของการทำโครงการมหาบัณฑิตซึ่งรวมถึงการสร้างโปรไฟล์แองกูลาร์เจเอส แสดงดังภาพที่ 3.1 ซึ่งเริ่มต้นจากการสร้างเมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส ผลลัพธ์ที่ได้คือเมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส แล้วจึงทำการสร้างโปรไฟล์แองกูลาร์เจเอส โดยใช้เมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสที่ได้ออกแบบไว้เป็นข้อมูลนำเข้า ผลลัพธ์ที่ได้คือโปรไฟล์แองกูลาร์เจเอส และทำการตรวจสอบโปรไฟล์ที่สร้างขึ้น หลังจากนั้นจึงทำการออกแบบกฎการแปลงสำหรับแปลงแบบจำลองไปเป็นโค้ด โดยใช้โปรไฟล์แองกูลาร์เจเอส เป็นข้อมูลนำเข้า ผลลัพธ์ที่ได้คือกฎการแปลงสำหรับแปลงแบบจำลองไปเป็นโค้ด สำหรับการประยุกต์ใช้กฎ จะทำการพัฒนาเครื่องมือสำหรับการแปลงแบบจำลองเป็นโค้ด จากนั้นทำการพัฒนาระบบกรณีศึกษาโดยใช้โปรไฟล์แองกูลาร์เจเอส เป็นข้อมูลนำเข้า ผลลัพธ์ที่ได้คือแผนภาพคลาสของการทำงานของเว็บแอปพลิเคชัน ต่อไปก็ทำการทดสอบการทำงานของเครื่องมือ โดยการใช้แผนภาพคลาสของเว็บแอปพลิเคชันซึ่งอยู่ในรูปไฟล์ XMI เป็นข้อมูลนำเข้า ผลลัพธ์ที่ได้คือโค้ดเทมเพลต หากเครื่องมือทำงานไม่ถูกต้อง ต้องมีการปรับปรุงให้เครื่องมือสามารถทำงานได้ตรงตามฟังก์ชันงานที่ได้กำหนดไว้ จากนั้นทำการเพิ่มเติมโค้ดเทมเพลตให้เป็นโค้ดที่สามารถทำงานได้ และสำหรับขั้นตอนสุดท้ายเป็นการประเมินผลการทำงานของเครื่องมือ โดยการวัดสัดส่วนการแปลงซึ่งใช้โค้ดเทมเพลตและโค้ดที่สามารถทำงานได้เป็นข้อมูลนำเข้า จะได้ผลลัพธ์เป็นผลการประเมินของการแปลงแบบจำลองเป็นโค้ด รายละเอียดของขั้นตอนต่าง ๆ มีดังนี้





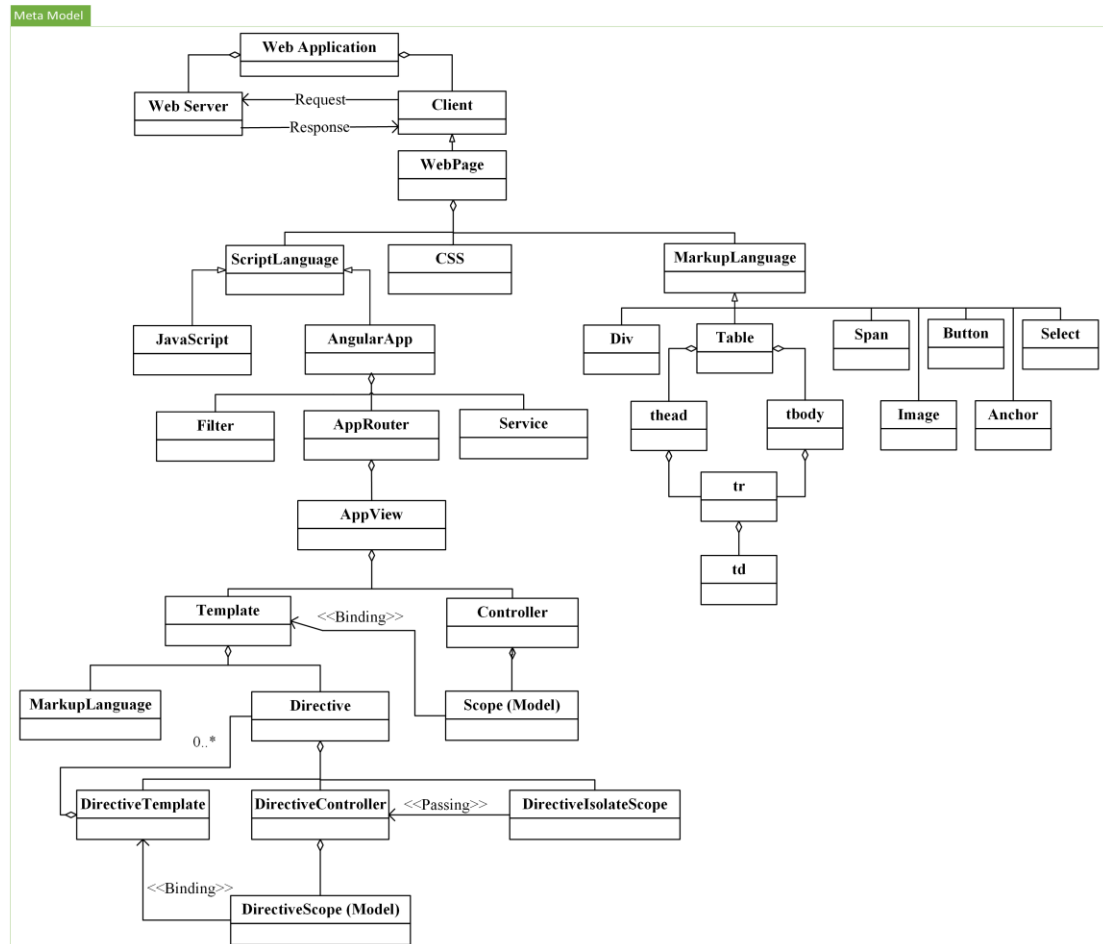
ภาพที่ 3.1 แผนภาพกิจกรรมแสดงภาพรวมของการทำโครงการมหาบัณฑิต

### 3.1 การสร้างเมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส

ในขั้นตอนนี้เป็นการศึกษาและทำความเข้าใจเกี่ยวกับรายละเอียดของโครงสร้างและส่วนประกอบของเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส ซึ่งเฟรมเวิร์กแองกูลาร์เจเอสนั้นทำงานที่ฝั่งไคลเอนต์ (Client) ทั้งหมด ดังนั้นเซิร์ฟเวอร์ (Server) จึงทำหน้าที่แค่เก็บไฟล์ (File) ที่เกี่ยวข้องสำหรับเว็บแอปพลิเคชันเท่านั้น คือ HTML (Hypertext Markup Language), JavaScript, CSS (Cascading Style Sheets) เท่านั้น และส่งไฟล์เหล่านี้กลับมาที่ฝั่งไคลเอนต์ เมื่อฝั่งไคลเอนต์ได้รับไฟล์กลับมาตัวเฟรมเวิร์กแองกูลาร์เจเอส จะทำการอ่านไฟล์ HTML และ JavaScript และแปลงเป็นโค้ดที่ทางเฟรมเวิร์กแองกูลาร์เจเอสเข้าใจ เว็บแอปพลิเคชันที่ใช้เฟรมเวิร์กแองกูลาร์เจเอสนั้นสามารถมีส่วนต่อประสานผู้ใช้ (UI Element) เช่น div, span, Checkbox, Select, Table, Image

ได้เป็นต้นและเฟรมเวิร์กแองกูลาร์เจเอสทำให้ผู้พัฒนาสามารถควบคุมส่วนต่อประสานผู้ใช้ได้ เพื่อให้มีการตอบสนองทันทีทันใด

เมตาโมเดลสำหรับความต้องการในการระบุโครงสร้างการทำงานและส่วนประกอบของเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสสามารถแสดงได้ดังภาพที่ 3.2



ภาพที่ 3.2 เมตาโมเดลสำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส (ปรับจาก [12])

### 3.2 การสร้างโปรไฟล์แองกูลาร์เจเอส

การสร้างโปรไฟล์ทำโดยใช้เมตาโมเดลในขั้นตอนก่อนหน้าเป็นข้อมูลตั้งต้นเพื่อสร้างส่วนขยายยูเอ็มแอลสำหรับการออกแบบเว็บแอปพลิเคชัน ซึ่งเรียกว่ายูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส แบ่งโปรไฟล์ออกเป็น 2 ส่วน ดังนี้

#### 3.2.1 สถาปัตยกรรมไคลเอนต์ด้วยยูเอ็มแอล (Client Architecture with UML)

สถาปัตยกรรมไคลเอนต์ใช้สำหรับอธิบายการทำงานฝั่งไคลเอนต์ (Client-Side) ของเว็บแอปพลิเคชัน จากงานวิจัย [12] [15] นำเสนอการใช้งานยูเอ็มแอลโปรไฟล์ที่สามารถเป็นตัวแทนของสถาปัตยกรรมสำหรับเว็บแอปพลิเคชัน โดยมีการกำหนดแม่พิมพ์ต้นแบบสำหรับส่วนประกอบของ

ไคลเอนต์ จึงสามารถนำแนวคิดดังกล่าวมาพัฒนาต่อเป็นความต้องการในการพัฒนาเว็บแอปพลิเคชัน ในส่วนของแบบจำลองส่วนแสดงผล (Presentation Model) ซึ่งแม่พิมพ์ต้นแบบของไคลเอนต์เป็นส่วนขยายมาจากเมตาคลาสชื่อว่า Artifact และ Class ในเมตาโมเดลของยูเอ็มแอล แสดงดังภาพที่ 3.3 และสามารถอธิบายรายละเอียดส่วนขยายของยูเอ็มแอลซึ่งประกอบด้วยแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์ได้ ดังตารางที่ 3.1

ตารางที่ 3.1 รายละเอียดแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์ (ปรับจาก [12,15])

แม่พิมพ์ต้นแบบ	ประเภทของ ยูเอ็มแอล	ความหมาย
<<AngularApp>>	Class	นิยามตัวแทนของแองกูลาร์เจเอสเว็บแอปพลิเคชันทั้งหมด
<<AppRouter>>	Class	นิยามหน้า View ที่แองกูลาร์เจเอสเว็บแอปพลิเคชันจะไปเปิดเป็นหน้าหลัก
<<AppView>>	Class	นิยามลักษณะหน้าตาของแองกูลาร์เจเอสเว็บหนึ่งหน้า
<<Binding>>	Association	เป็นความสัมพันธ์ระหว่าง Scope และ Template ซึ่งถ้า Scope มีการเปลี่ยนแปลง ตัว Template ซึ่งก็คือหน้าตาของเว็บ ส่วนที่สัมพันธ์กับ Scope นั้นก็จะมีการเปลี่ยนแปลงทันที
<<Controller>>	Class	ฟังก์ชันเพื่อที่จะดำเนินการตาม Business logic
<<CSS>>	Class	ป้ายนิยาม CSS (CSS tag) ของเว็บแอปพลิเคชัน
<<Directive>>	Class	AngularJS directive
<<DirectiveTemplate>>	Class	โครงสร้างของ HTML บางส่วนซึ่งเป็นส่วนประกอบ ของ Directive
<<DirectiveController>>	Class	ฟังก์ชันเพื่อดำเนินการตาม Business logic ใน Directive
<<DirectiveScope>>	Class	Model ที่จะถูกนำไปใช้ใน Directive
<<DirectiveIsolateScope>>	Class	Model ที่ถูกส่งมาจากด้านนอกของ Directive เพื่อเข้ามาทำงานใน Directive

ตารางที่ 3.1 รายละเอียดแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไมโครเอ็นต์ (ปรับจาก [12,15]) (ต่อ)

แม่พิมพ์ต้นแบบ	ประเภทของยูเอ็มแอล	ความหมาย
<<Filter>>	Class	ฟังก์ชันที่ทำหน้าที่ในการแปลงค่าของข้อมูล ก่อนการแสดงผลเพื่อให้อยู่ในรูปแบบที่กำหนด
<<MarkupLanguage>>	Class	ป้ายนิยามของ HTML (HTML tag) เพื่อที่จะไว้แสดงผลที่ฝั่ง Client
<<Passing>>	Association	เป็นความสัมพันธ์ระหว่าง DirectivelsolateScope และ DirectiveController ซึ่ง DirectivelsolateScope จะส่งข้อมูลที่ต้องการนำเข้ามาประมวลผลที่ DirectiveController โดยส่งผ่านมาจาก Directive เพื่อนำมาประมวลผลตาม Business logic
<<Scope>>	Class	Model ที่ถูกใช้ในการกำหนดค่าให้ Template
<<ScriptLanguage>>	Class	JavaScript file ที่ถูกใช้ในเว็บแอปพลิเคชัน
<<Service>>	Class	Shared function ที่ใช้งานตลอดทั้งแองกูลาร์เจเอสเว็บแอปพลิเคชัน
<<Template>>	Class	User interface ลักษณะของ HTML ของแองกูลาร์เจเอส View
<<WebPage>>	Class	Web Page ทั้งหมดที่ถูกส่งมาจาก Web Server

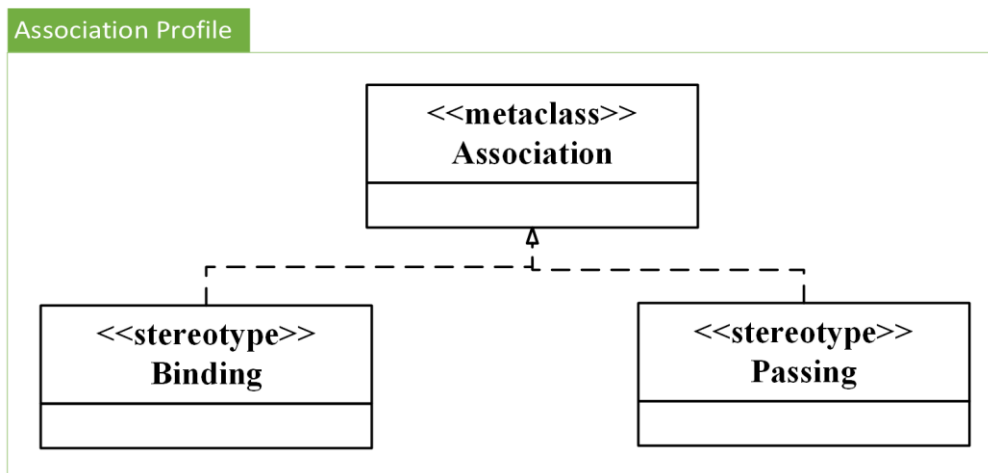


สถาปัตยกรรมความสัมพันธ์ใช้สำหรับอธิบายแบบจำลองของความสัมพันธ์ระหว่างแต่ละคลาสของเว็บแอปพลิเคชัน ซึ่งแม่พิมพ์ต้นแบบของความสัมพันธ์เป็นส่วนขยายมาจากเมตาคลาสชื่อ Association ในเมตาโมเดลของยูเอ็มแอล แสดงดังภาพที่ 3.4 และสามารถอธิบายรายละเอียด

ส่วนขยายยูเอ็มแอลซึ่งประกอบด้วยแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์ได้ดังตารางที่ 3.2

ตารางที่ 3.2 รายละเอียดแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์ (ปรับจาก [12,15])

แม่พิมพ์ต้นแบบ	ประเภทของยูเอ็มแอล	ความหมาย
<<Binding>>	Association	แสดงถึงการกระทำของเฟรมเวิร์กแองกูลาร์เจเอสที่เมื่อมีการเปลี่ยนแปลงค่าของ Scope หรือ DirectiveScope ก็จะมีการเปลี่ยนแปลงค่าของ Template หรือ DirectiveTemplate โดยทันทีตามลำดับ
<<Passing>>	Association	แสดงถึงการส่งค่าของตัวแปร จาก Directive เข้ามายัง DirectiveController ผ่านทาง DirectiveIsolateScope เพื่อที่จะให้ DirectiveController นำไปประมวลผลต่อไป



ภาพที่ 3.4 แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์ (ปรับจาก [12] [15])

### 3.3 การตรวจสอบโปรไฟล์แองกูลาร์เจเอส

ในการตรวจสอบยูเอ็มแอลโปรไฟล์สำหรับเฟรมเวิร์กแองกูลาร์เจเอส จะตรวจสอบโดยพิจารณาจากคุณสมบัติมาตรฐานของยูเอ็มแอลโปรไฟล์ [3] โดยมีรายละเอียดดังต่อไปนี้

- 1) เป็นซับเซตของยูเอ็มแอลเมตาโมเดล (UML Metamodel)
- 2) มีกฎในการควบคุมรูปแบบการใช้งาน (Well-Formedness Rules)
- 3) ใช้ระบุงค์ประกอบที่นอกเหนือจากองค์ประกอบมาตรฐานของยูเอ็มแอล
- 4) มีการกำหนดความหมายของแต่ละองค์ประกอบที่ได้สร้างไว้

5) มีการกำหนดองค์ประกอบแบบจำลองที่ใช้ในองค์ประกอบที่ได้สร้างไว้

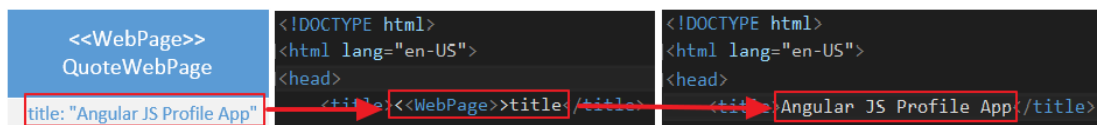
สำหรับการตรวจสอบความครบถ้วนของตัวโปรแกรมแองกูลาร์เจเอส ทำโดยการใช้เมตาโมเดล สำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสที่ได้ออกแบบไว้ในขั้นตอนก่อนหน้า รวมถึงการ นำแม่พิมพ์ต้นแบบในโปรแกรมแองกูลาร์เจเอส ไปทำการออกแบบกฎการแปลงสำหรับแปลง แบบจำลองเป็นโค้ด จากนั้นทำการตรวจสอบดูว่ายังมีส่วนใดของโปรแกรมที่ต้องแก้ไขเพิ่มเติม เพื่อให้ได้โปรแกรมแองกูลาร์เจเอส ที่มีความถูกต้องครบถ้วน

## บทที่ 4

### กฎการแปลงแบบจำลองเป็นโค้ด

กฎการแปลงแม่พิมพ์ต้นแบบซึ่งอ้างอิงตามยูเอ็มแอลโปรไฟล์สำหรับเฟรมเวิร์กแองกูลาร์เจเอสไปเป็นโค้ด จะใช้แนวทางของงานวิจัย [12] ประกอบด้วยกฎสำหรับแปลงแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์ โดยกฎทั้งหมดสำหรับการแปลงแบบจำลองที่ออกแบบโดยใช้แม่พิมพ์ต้นแบบจะนำเสนอเป็น 4 ส่วน คือ ส่วนที่ 1 แสดงยูเอ็มแอลโปรไฟล์สำหรับเฟรมเวิร์กแองกูลาร์เจเอส ส่วนที่ 2 แสดงส่วนของแบบจำลองที่ออกแบบโดยใช้แม่พิมพ์ต้นแบบตามยูเอ็มแอลโปรไฟล์สำหรับเฟรมเวิร์กแองกูลาร์เจเอส ส่วนที่ 3 แสดงแผ่นแบบ (Template) ของแบบจำลองเพื่อเตรียมที่จะนำไปแปลงเป็นโค้ดเทมเพลตต่อไป และส่วนที่ 4 แสดงตัวอย่างของโค้ดเทมเพลต (Code Template) ที่ได้รับหลังจากมีการแปลงแบบจำลอง

ความหมายของสัญลักษณ์ที่ปรากฏในกฎจะใช้สัญลักษณ์ `<<stereotype>>tagname` เพื่อระบุตำแหน่งของการแปลงค่าของป้ายระบุ (Tag) หรือคุณสมบัติ (Attribute) ไปเป็นโค้ด โดยการแปลงจะนำค่าของป้ายระบุหรือคุณสมบัติของแบบจำลองนั้นไปแทนค่าในสัญลักษณ์ดังกล่าว ดังตัวอย่างตามภาพที่ 4.1



ภาพที่ 4.1 การแปลงป้ายระบุหรือคุณสมบัติของแบบจำลองเป็นโค้ด

แม่พิมพ์ต้นแบบจะมีนิยามป้ายระบุ (Tag) ซึ่งช่วยกำหนดคุณลักษณะเฉพาะของแม่พิมพ์ต้นแบบและมีการกำหนดค่าป้ายระบุให้เมื่อนำแม่พิมพ์ต้นแบบไปใช้ในการออกแบบ ดังจะอธิบายในกฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบต่างๆ ทั้ง 17 กฎในหัวข้อถัดไป

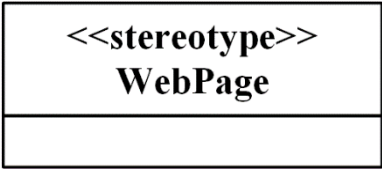
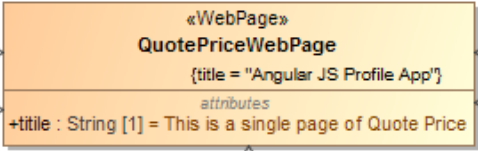
#### 4.1 กฎสำหรับแปลงแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมไคลเอนต์เป็นโค้ด

##### 4.1.1 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ WebPage

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ WebPage แสดงในตารางที่ 4.1 ซึ่งแบบจำลอง WebPage ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw (Version 18.2) [18] ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว



ตารางที่ 4.1 แม่พิมพ์ต้นแบบและแบบจำลอง ของ WebPage

แม่พิมพ์ต้นแบบ	แบบจำลอง
	

ส่วนแม่พิมพ์ต้นแบบของ WebPage จะมีป้ายกำกับ (Tag) คือ

- 1) `<<WebPage>>title` เพื่อกำหนดชื่อเรื่องของเว็บแอปพลิเคชัน และมีกฎการแปลงดังภาพที่ 4.2 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ดังภาพที่ 4.3

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width">
  <title><<WebPage>>title</title>
```

ภาพที่ 4.2 กฎการแปลงป้ายระบุแบบจำลอง WebPage เป็นโค้ด

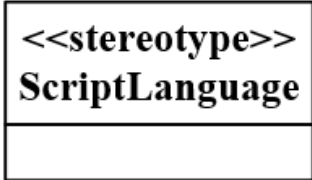
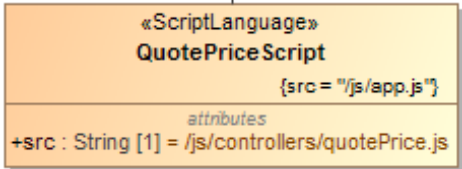
```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width">
  <title>Angular JS Profile App</title>
```

ภาพที่ 4.3 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง WebPage ไปใช้

#### 4.1.2 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ ScriptLanguage

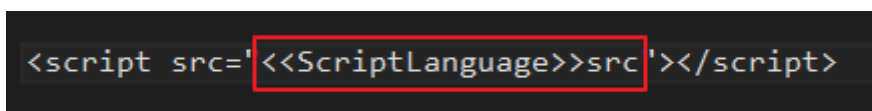
รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ ScriptLanguage แสดงในตารางที่ 4.2 ซึ่งแบบจำลอง ScriptLanguage ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.2 แม่พิมพ์ต้นแบบและแบบจำลอง ของ ScriptLanguage

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> &lt;&lt;stereotype&gt;&gt; ScriptLanguage           </pre>	 <pre> «ScriptLanguage» QuotePrice Script     {src = "/js/app.js"} attributes +src : String [1] = /js/controllers/quotePrice.js           </pre>

ส่วนแม่พิมพ์ต้นแบบของ ScriptLanguage จะมีป้ายกำกับ (Tag) คือ

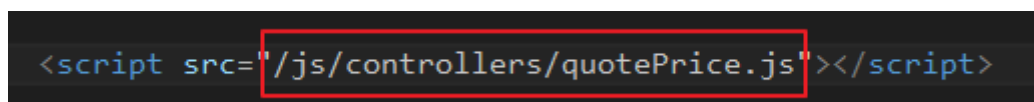
1) `<<ScriptLanguage>>_src` เพื่อกำหนดตำแหน่งของ Javascript file และมีกฎการแปลงดังภาพที่ 4.4 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ดังภาพที่ 4.5



```

<script src='<<ScriptLanguage>>src'></script>
          
```

ภาพที่ 4.4 กฎการแปลงป้ายระบุของแบบจำลอง ScriptLanguage เป็นโค้ด



```

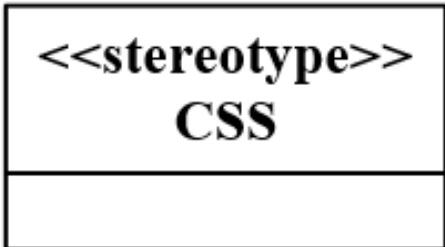
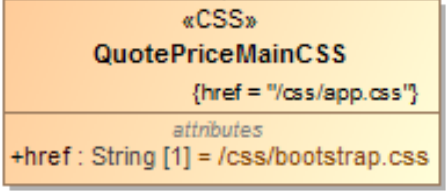
<script src='/js/controllers/quotePrice.js'></script>
          
```

ภาพที่ 4.5 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง ScriptLanguage ไปใช้

#### 4.1.3 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ CSS

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ CSS แสดงในตารางที่ 4.3 ซึ่งแบบจำลอง CSS ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.3 แม่พิมพ์ต้นแบบและแบบจำลอง ของ CSS

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> &lt;&lt;stereotype&gt;&gt; CSS           </pre>	 <pre> «CSS» QuotePriceMainCSS     {href = "/css/app.css"} attributes +href : String [1] = /css/bootstrap.css           </pre>

ส่วนแม่พิมพ์ต้นแบบของ CSS จะมีป้ายกำกับ (Tag) คือ

- 1) `<<CSS>>href` เพื่อกำหนดตำแหน่งของ CSS file และมีกฎการแปลงดังภาพที่ 4.6 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ดังภาพที่ 4.7

```
<link rel="stylesheet" href="<<CSS>>href">
```

ภาพที่ 4.6 กฎการแปลงป้ายระบุของแบบจำลอง ScriptLanguage เป็นโค้ด

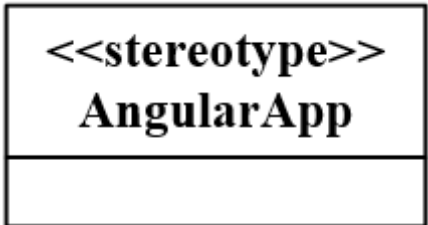
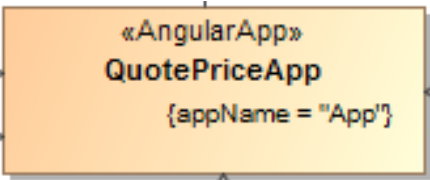
```
<link rel="stylesheet" href="/css/bootstrap.css">
```

ภาพที่ 4.7 โค้ดเต็มเพลทหลังจากผ่านการนำกฎการแปลง CSS ไปใช้

#### 4.1.4 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ AngularApp

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ AngularApp แสดงในตารางที่ 4.4 ซึ่งแบบจำลอง AngularApp ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.4 แม่พิมพ์ต้นแบบและแบบจำลอง ของ AngularApp

แม่พิมพ์ต้นแบบ	แบบจำลอง
	

ส่วนแม่พิมพ์ต้นแบบของ AngularApp จะมีป้ายกำกับ (Tag) คือ

- 1) `<<AngularApp>>appName` เพื่อกำหนดชื่อของ Angular Web Application และมีกฎการแปลงดังภาพที่ 4.8 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ดังภาพที่ 4.9

```
ng-app="<<AngularApp>>appName"
```

ภาพที่ 4.8 กฎการแปลงป้ายระบุของแบบจำลอง AngularApp เป็นโค้ด

```

<body ng-app='App'>

  <!-- Main App UI(view) -->
  <div id="appView" ui-view></div>

</body>

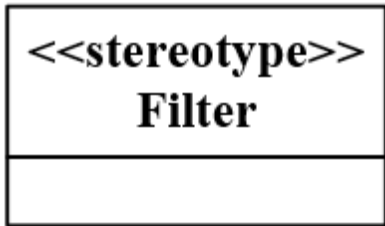

```

ภาพที่ 4.9 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง AngularApp ไปใช้

#### 4.1.5 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Filter

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ Filter แสดงในตารางที่ 4.5 ซึ่งแบบจำลอง Filter ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว

ตารางที่ 4.5 แม่พิมพ์ต้นแบบและแบบจำลอง ของ Filter

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> &lt;&lt;stereotype&gt;&gt; Filter </pre>	 <pre> «Filter» numberFilter {   filterName = "num",   filterParams = "input,defaultIfBlank,decimalPlace" } </pre>

ส่วนแม่พิมพ์ต้นแบบของ Filter จะมีป้ายกำกับ (Tag) คือ

- 1) `<<Filter>>filterName` เพื่อกำหนดชื่อของ AngularJS Filter
- 2) `<<Filter>>filterParams` เพื่อกำหนด parameters ของ AngularJS Filter และมีกฎการแปลงดังภาพที่ 4.10 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ดังภาพที่ 4.11

```

Formatter.filter(<<Filter>>filterName, ['$filter', function($filter) {
  return function(<<Filter>>filterParams) {

    // Add Business logic here !!
    return null;
  };
}]);

```

ภาพที่ 4.10 กฎการแปลงป้ายระบุของแบบจำลอง Filter เป็นโค้ด

```

Formatter.filter('num', ['$filter', function ($filter) {
    return function (input, defaultIfBlank, decimalPlace) {

        // Add business logic here !!
        return null;
    };
}));

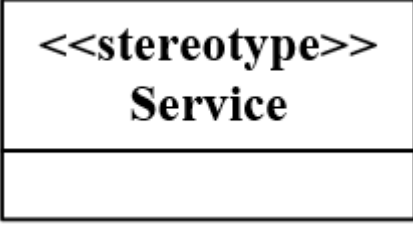
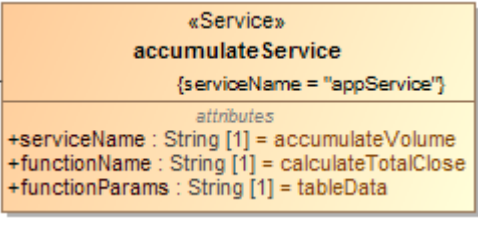
```

ภาพที่ 4.11 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Filter ไปใช้

#### 4.1.6 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Service

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ Service แสดงในตารางที่ 4.6 ซึ่งแบบจำลอง Service ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว

ตารางที่ 4.6 แม่พิมพ์ต้นแบบและแบบจำลอง ของ Service

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> &lt;&lt;stereotype&gt;&gt; Service </pre>	 <pre> «Service» accumulateService {serviceName = "appService"}  attributes +serviceName : String [1] = accumulateVolume +functionName : String [1] = calculateTotalClose +functionParams : String [1] = tableData </pre>

ส่วนแม่พิมพ์ต้นแบบของ Service จะมีป้ายกำกับ (Tag) คือ

- 1) <<Service>>\_serviceName เพื่อกำหนดชื่อของ AngularJS Service
- 2) <<Service>>\_functionName เพื่อกำหนดชื่อของฟังก์ชันของ AngularJS Service
- 3) <<Service>>\_functionParams เพื่อกำหนด Parameters ของ Function ของ AngularJS

Service รวมถึงมีกฎการแปลงดังภาพที่ 4.12 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ดังภาพที่ 4.13

```
AppServices.service('<<Service>>serviceName', ['appModel', function(appModel) {
    return {
        <<Service>>functionName: function(<<Service>>functionParams) {
            return null;
        }
    }
}]);
```

ภาพที่ 4.12 กฎการแปลงป้ายระบุของแบบจำลอง Service เป็นโค้ด

```
AppServices.service('accumulateVolume', ['appModel', function(appModel) {
    return {
        calculateTotalClose: function(tableData) {
            return null;
        }
    }
}]);
```

ภาพที่ 4.13 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Service ไปใช้

#### 4.1.7 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ AppRouter

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ AppRouter แสดงในตารางที่ 4.7 ซึ่งแบบจำลอง AppRouter ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.7 แม่พิมพ์ต้นแบบและแบบจำลอง ของ AppRouter

แม่พิมพ์ต้นแบบ	แบบจำลอง
<pre>&lt;&lt;stereotype&gt;&gt; AppRouter</pre>	<pre>«AppRouter» QuotePriceAppRouter {defaultUrl = "/QuotePrice"}</pre>

ส่วนแม่พิมพ์ต้นแบบของ AppRouter จะมีป้ายกำกับ (Tag) คือ

1) `<<AppRouter>>defaultUrl` เพื่อกำหนด View ที่จะแสดงผลเป็นหน้าหลักของ AngularJS web application และมีกฎการแปลงดังภาพที่ 4.14 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.15

```
$urlRouterProvider.otherwise('<<AppRouter>>defaultUrl');
```

ภาพที่ 4.14 กฎการแปลงป้ายระบุของแบบจำลอง AppRouter เป็นโค้ด

```
App.config(['$stateProvider', '$urlRouterProvider',
function ($stateProvider, $urlRouterProvider) {
    $urlRouterProvider.otherwise('/QuotePrice');
```

ภาพที่ 4.15 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง AppRouter ไปใช้

#### 4.1.8 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ AppView

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ AppView แสดงในตารางที่ 4.8 ซึ่งแบบจำลอง AppView ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแอ่งกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.8 แม่พิมพ์ต้นแบบและแบบจำลอง ของ AppView

แม่พิมพ์ต้นแบบ	แบบจำลอง
<pre>&lt;&lt;stereotype&gt;&gt; AppView</pre>	<pre>«AppView» QuotePriceAppView {controllerName = "quotePriceController", stateName = "quotePrice", stateUrl = "/QuotePrice", templateURL = "templates/views/quotePrice.html"}</pre>

ส่วนแม่พิมพ์ต้นแบบของ AppView จะมีป้ายกำกับ (Tag) คือ

- 1) `<<AppView>>_controllerName` เพื่อกำหนด ชื่อของ Controller ของหน้า View,
- 2) `<<AppView>>_stateName` จะเป็นตัวกำหนดให้ AngularJS ทราบว่าจะต้องใช้ ชื่ออะไร ในการเปิด View
- 3) `<<AppView>>_stateUrl` เป็นตัวกำหนด URL ของ AngularJS web application ที่ users จะนำไปใช้เพื่อเปิดหน้า View
- 4) `<<AppView>>_templateURL` เป็นตำแหน่งของ template file ของหน้า View รวมถึงมีกฎการแปลงดังภาพที่ 4.16 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.17

```
.state(<<AppView>>stateName', {
    url: '<<AppView>>stateUrl',
    templateUrl: '<<AppView>>templateURL',
    controller: '<<AppView>>controllerName'
})
```

ภาพที่ 4.16 กฎการแปลงป้ายระบุของแบบจำลอง AppView เป็นโค้ด

```

$stateProvider
//-----
// Sub-views here!!
.state('quotePrice', {
  url: '/QuotePrice',
  templateUrl: 'templates/views/quotePrice.html',
  controller: 'quotePriceController'
});
//-----

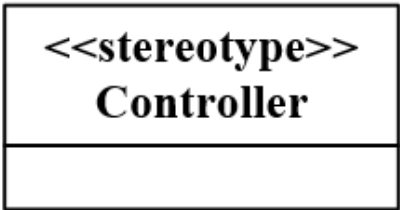
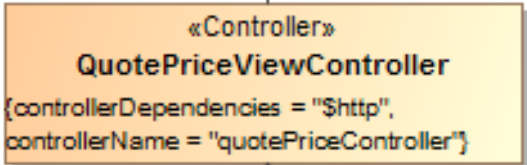
```

ภาพที่ 4.17 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง AppView ไปใช้

#### 4.1.9 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Controller

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ Controller แสดงในตารางที่ 4.9 ซึ่งแบบจำลอง Controller ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.9 แม่พิมพ์ต้นแบบและแบบจำลอง ของ Controller

แม่พิมพ์ต้นแบบ	แบบจำลอง
	

ส่วนแม่พิมพ์ต้นแบบของ Controller จะมีป้ายกำกับ (Tag) คือ

1) <<Controller>><sub>controllerName</sub> จะเป็นชื่อ Controller ของ View

2) <<Controller>><sub>controllerDependencies</sub> เพื่อกำหนด Dependencies ของ AngularJS

Controller ของหน้า View แต่ตัว Dependencies ของ AngularJS นั้นจะมีอยู่ 2 ตำแหน่งทำให้ต้องมีการแบ่งตำแหน่งของ Dependencies ออกเป็น 2 ส่วนคือ ส่วนหน้า (frontDependencies) เพื่อเป็นการบอก AngularJS ถึง Dependencies ที่ทาง AngularJS ต้องมีการ Load เข้ามา ส่วนอีกตำแหน่งคือ Parameters ของ Function (behindDependencies) เพื่อให้ทางผู้พัฒนาได้นำตัวแปรนี้ไปเรียกใช้งานใน Function ได้ รวมถึงมีกฎการแปลงดังภาพที่ 4.18 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.19



```
'use strict';

(function() {

    var App = angular.module('App');

    App.controller('<<Controller>>controllerName', ['$scope' <<Controller>>frontDependencies,
        function ($scope <<Controller>>behindDependencies) {

        }]);

})();
```

ภาพที่ 4.18 กฎการแปลงป้ายระบุของแบบจำลอง Controller เป็นโค้ด

```
'use strict';

(function() {

    var App = angular.module('App');

    App.controller('quotePriceController', ['$scope', '$http', function($scope, $http) {

    }]);

})();
```

ภาพที่ 4.19 โค้ดเต็มเพลทหลังจากผ่านการนำกฎการแปลง Controller ไปใช้

#### 4.1.10 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Scope

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ Scope แสดงในตารางที่ 4.10 ซึ่งแบบจำลอง Scope ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.10 แม่พิมพ์ต้นแบบและแบบจำลอง ของ Scope

แม่พิมพ์ต้นแบบ	แบบจำลอง
<pre>&lt;&lt;stereotype&gt;&gt; Scope (Model)</pre>	<pre>«Scope» QuotePriceViewScope {     name = "title",     value = "This is Quote Price Table component" }</pre>

ส่วนแม่พิมพ์ต้นแบบของ Scope จะมีป้ายกำกับ (Tag) คือ

- 1) `<<Scope>>_value` เป็นการกำหนดค่าของ Scope

2) `<<Scope>>name` จะเป็นชื่อของ Scope ของ Controller เพื่อกำหนดให้ View หรือ Template นำค่า Value ไปแสดงผล รวมถึงมีกฎการแปลงดังภาพที่ 4.20 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.21

```
$scope.<<Scope>>name = <<Scope>>value;
```

ภาพที่ 4.20 กฎการแปลงป้ายระบุของแบบจำลอง Scope เป็นโค้ด

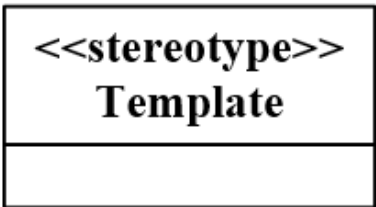
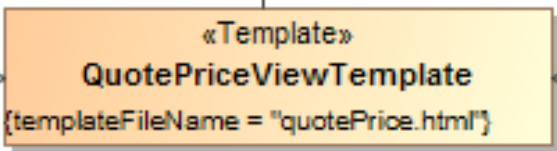
```
$scope.title = "This is Quote Price Table component";
```

ภาพที่ 4.21 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Scope ไปใช้

#### 4.1.11 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Template

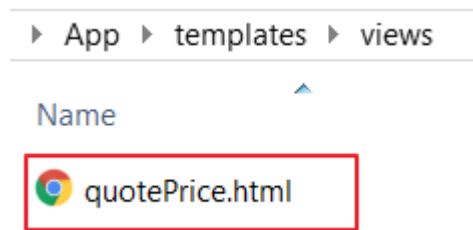
รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ Template แสดงในตารางที่ 4.10 ซึ่งแบบจำลอง Template ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว

ตารางที่ 4.10 แม่พิมพ์ต้นแบบและแบบจำลองของ Template

แม่พิมพ์ต้นแบบ	แบบจำลอง
	

ส่วนแม่พิมพ์ต้นแบบของ Template จะมีป้ายกำกับ (Tag) คือ

1) `<<Template>>templateFileName` เป็นการกำหนดชื่อของ AngularJS Template file แต่ไม่มีกฎการแปลงเนื่องจากต้องมีการสร้าง File Template ขึ้นมาเพื่อเตรียมเก็บแม่พิมพ์ MarkupLanguage ดังภาพที่ 4.20



ภาพที่ 4.22 ไฟล์เทมเพลตหลังจากผ่านการนำแบบจำลอง Template ไปใช้

#### 4.1.12 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ MarkupLanguage

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ MarkupLanguage แสดงในตารางที่ 4.12 ซึ่งแบบจำลอง MarkupLanguage ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว

ตารางที่ 4.12 แม่พิมพ์ต้นแบบและแบบจำลองของ MarkupLanguage

แม่พิมพ์ต้นแบบ	แบบจำลอง

ส่วนแม่พิมพ์ต้นแบบของ MarkupLanguage จะมีป้ายกำกับ (Tag) คือ

- 1) `<<MarkupLanguage>>domId` เป็นการกำหนด Id ของ HTML tag
- 2) `<<MarkupLanguage>>domType` จะเป็น HTML Tag ของ HTML element
- 3) `<<MarkupLanguage>>order` เป็นลำดับของชั้นของ HTML tag
- 4) `<<MarkupLanguage>>css` จะเป็น CSS ของ HTML tag
- 5) `<<MarkupLanguage>>modelName` เป็น Scope ของ AngularJS ซึ่งเมื่อ AngularJS มีการทำงานจะนำค่าของ Value ของตัวแปรที่มีชื่อตรงกับ modelName มาแปลงเป็นค่าที่ต้องการแสดงผล
- 6) `<<MarkupLanguage>>filterName` เป็นชื่อ AngularJS Filter ที่จะทำการแปลงค่าของข้อมูลอยู่ในรูปแบบที่ต้องการ
- 7) `<<MarkupLanguage>>filterParams` เป็น Parameters ของ AngularJS Filter
- 8) `<<MarkupLanguage>>directiveName` เป็นชื่อของ AngularJS Directive

9) `<<MarkupLanguage>>text` เป็นข้อความที่จะให้แสดงผลของ HTML Tag รวมถึงมีกฎการแปลงดังภาพที่ 4.23 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.24

```
<<MarkupLanguage>>domType id="<<MarkupLanguage>>domId" <<MarkupLanguage>>css
<<MarkupLanguage>>directiveName
<<MarkupLanguage>>filterName <<MarkupLanguage>>filterParams >
<<MarkupLanguage>>text <<MarkupLanguage>>modelName
</ <<MarkupLanguage>>domType >
```

ภาพที่ 4.23 กฎการแปลงป้ายระบุของแบบจำลอง MarkupLanguage เป็นโค้ด

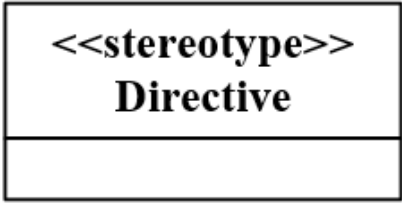
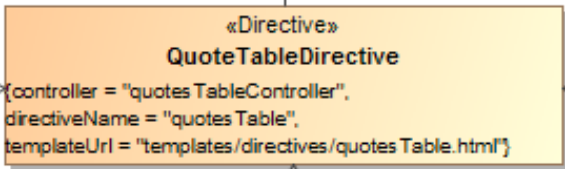
```
<div id="domTemplate">
  <h3 id="domTemplate"></h3>
  <div id="domTemplate" quotes-table></div>
</div>
```

ภาพที่ 4.24 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง MarkupLanguage ไปใช้

#### 4.1.13 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ Directive

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ Directive แสดงในตารางที่ 4.13 ซึ่งแบบจำลอง Directive ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.13 แม่พิมพ์ต้นแบบและแบบจำลองของ Directive

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre>&lt;&lt;stereotype&gt;&gt; Directive</pre>	 <pre>«Directive» QuoteTableDirective {   controller = "quotes TableController",   directiveName = "quotes Table",   templateUrl = "templates/directives/quotes Table.html" }</pre>

ส่วนแม่พิมพ์ต้นแบบของ Directive จะมีป้ายกำกับ (Tag) คือ

- 1) `<<Directive>>controller` เป็นการกำหนดค่า Controller ของ Directive
- 2) `<<Directive>>directiveName` เป็นตัวกำหนดชื่อของ Directive ที่จะถูกนำไปใช้ใน

AngularJS web application

3) `<<Directive>>templateUrl` จะเป็นการระบุตำแหน่งของ Template file ของ Directive รวมถึงมีกฎการแปลงดังภาพที่ 4.25 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.26

```
App.directive(<<Directive>>directiveName', [function() {
  return {
    restrict: 'A',
    replace: true,
    <<DirectiveIsolateScope>>DirectiveIsolateScope
    templateUrl: 'templates/directives/<<Directive>>directiveName.html',
    controller: '<<Directive>>controller'
  };
}]);
```

ภาพที่ 4.25 กฎการแปลงป้ายระบุของแบบจำลอง Directive เป็นโค้ด


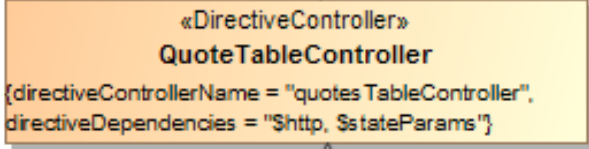
```
App.directive('quotesTable', [function () {
  return {
    restrict: 'A',
    replace: true,
    scope: {
      mainData: '@'
    },
    templateUrl: 'templates/directives/quotesTable.html',
    controller: 'quotesTableController'
  };
}]);
```

ภาพที่ 4.26 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง Directive ไปใช้

#### 4.1.14 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ DirectiveController

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveController แสดงในตารางที่ 4.14 ซึ่งแบบจำลอง DirectiveController ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.14 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveController

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> classDiagram     class DirectiveController {         &lt;&lt;stereotype&gt;&gt;     }         </pre>	 <pre> classDiagram     class QuoteTableController {         &lt;&lt;DirectiveController&gt;&gt;         {directiveControllerName = "quotesTableController",         directiveDependencies = "\$http, \$stateParams"}     }         </pre>

ส่วนแม่พิมพ์ต้นแบบของ DirectiveController ซึ่งเป็นตัวควบคุม Business logic ต่าง ๆ ของ Directive จะมีป้ายกำกับ (Tag) คือ

- 1) `<<DirectiveController>>directiveControllerName` เป็นการกำหนดชื่อของ DirectiveController
- 2) `<<DirectiveController>>directiveDependencies` เพื่อกำหนด Dependencies ของ AngularJS Directive แต่ตัว Dependencies ของ AngularJS นั้นจะมีอยู่ 2 ตำแหน่งทำให้ต้องมีการแบ่งตำแหน่งของ Dependencies ออกเป็น 2 ส่วนคือ ส่วนหน้า (frontDependencies) เพื่อเป็นการบอก AngularJS ถึง Dependencies ที่ทาง AngularJS ต้องมีการ Load เข้ามา ส่วนอีกตำแหน่งคือ Parameters ของ Function (behindDependencies) เพื่อให้ทางผู้พัฒนาได้นำตัวแปรนี้ไปเรียกใช้งานใน Function ของ Directive Controller ได้ รวมถึงมีกฎการแปลงดังภาพที่ 4.27 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.28

```

App.controller('<<DirectiveController>>directiveControllerName', ['$scope', '<<DirectiveController>>frontDependencies',
function ($scope, <<DirectiveController>>behindDependencies) {

    // Add directive scope here !!

}]);
        
```

ภาพที่ 4.27 กฎการแปลงป้ายระบุของแบบจำลอง DirectiveController เป็นโค้ด

```

App.controller('quotesTableController', ['$scope', '$http', '$stateParams',
function ($scope, $http, $stateParams) {

    // Add directive scope here !!

}]);
        
```

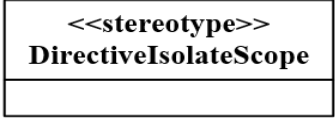
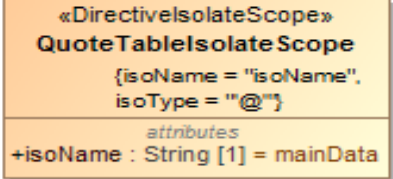
ภาพที่ 4.28 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง DirectiveController ไปใช้

#### 4.1.15 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ DirectivelsolateScope

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ DirectivelsolateScope แสดงใน ตารางที่ 4.15 ซึ่งแบบจำลอง DirectivelsolateScope ในตารางนั้นได้ผ่านการสร้างจาก UML Tool

ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ภายใน UML Modeling tool แล้ว

ตารางที่ 4.15 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectivelsolateScope

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> &lt;&lt;stereotype&gt;&gt; DirectiveIsolateScope           </pre>	 <pre> «DirectiveIsolateScope» QuoteTableIsolate Scope {isoName = "isoName",  isoType = "@"} attributes +isoName : String [1] = mainData           </pre>

ส่วนแม่พิมพ์ต้นแบบของ DirectivelsolateScope ซึ่งเป็นช่องทางที่ให้ Directive สามารถส่งค่าตัวแปร เข้ามาประมวลผลใน DirectiveController ซึ่งทำให้ DirectiveController นำไปประมวลผลตาม Business logic ได้และ DirectivelsolateScope จะมีป้ายกำกับ (Tag) คือ

1) `<<DirectiveIsolateScope>>isoName` เป็นการกำหนดชื่อตัวแปรที่จะถูกส่งเข้ามาประมวลผลตาม Business logic ใน DirectiveController รวมถึงมีกฎการแปลงดังภาพที่ 4.29 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.30

```

scope : {
  <<DirectiveIsolateScope>>IsolateScope
}
          
```

ภาพที่ 4.29 กฎการแปลงป้ายระบุของแบบจำลอง DirectivelsolateScope เป็นโค้ด

```

App.directive('quotesTable', [function () {
  return {
    restrict: 'A',
    replace: true,
    scope: {
      mainData: '@'
    },
    templateUrl: 'templates/directives/quotesTable.html',
    controller: 'quotesTableController'
  };
}]);
          
```

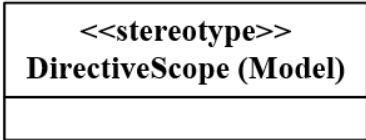
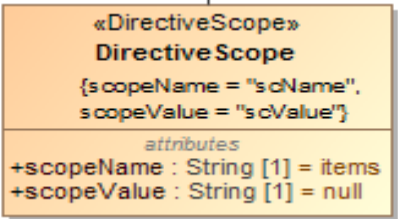
ภาพที่ 4.30 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง DirectivelsolateScope ไปใช้

#### 4.1.16 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ DirectiveScope

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveScope แสดงในตารางที่ 4.16 ซึ่งแบบจำลอง DirectiveScope ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า

MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว

ตารางที่ 4.16 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveScope

แม่พิมพ์ต้นแบบ	แบบจำลอง
 <pre> &lt;&lt;stereotype&gt;&gt; DirectiveScope (Model)           </pre>	 <pre> «DirectiveScope» Directive Scope {scopeName = "s cName", scopeValue = "s cValue"}  attributes +scopeName : String [1] = items +scopeValue : String [1] = null           </pre>

ส่วนแม่พิมพ์ต้นแบบของ DirectiveScope จะมีป้ายกำกับ (Tag) คือ

- 1) `<<DirectiveScope>>scopeValue` เป็นการกำหนดค่าของ Scope ใน Directive
- 2) `<<DirectiveScope>>scopeName` จะเป็นชื่อของ Scope ของ Directive เพื่อกำหนดให้

Directive Template นำค่า Value ไปแสดงผล รวมถึงมีกฎการแปลงดังภาพที่ 4.31 หลังจากทำการแปลงแบบจำลองเป็นโค้ดแล้วจะได้ผลลัพธ์ ดังภาพที่ 4.32

```
$scope.<<DirectiveScope>>scopeName = <<DirectiveScope>>scopeValue;
```

ภาพที่ 4.31 กฎการแปลงป้ายระบุของแบบจำลอง DirectiveScope เป็นโค้ด

```
App.controller('quotesTableController', ['$scope', '$http', '$stateParams',
function ($scope, $http, $stateParams) {

    $scope.items = null;

}]);
```

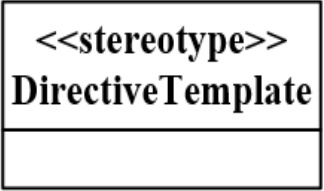
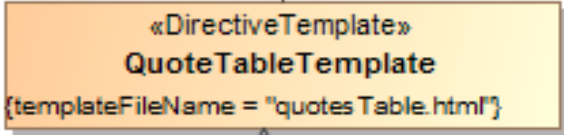
ภาพที่ 4.32 โค้ดเทมเพลตหลังจากผ่านการนำกฎการแปลง DirectiveScope ไปใช้

#### 4.1.17 กฎการแปลงสำหรับแบบจำลองที่ใช้แม่พิมพ์ต้นแบบของ DirectiveTemplate

รายละเอียดของแม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveTemplate แสดงในตารางที่ 4.17 ซึ่งแบบจำลอง DirectiveTemplate ในตารางนั้นได้ผ่านการสร้างจาก UML Tool ที่ชื่อว่า MagicDraw ซึ่งผู้วิจัยได้สร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ใน UML Modeling tool แล้ว

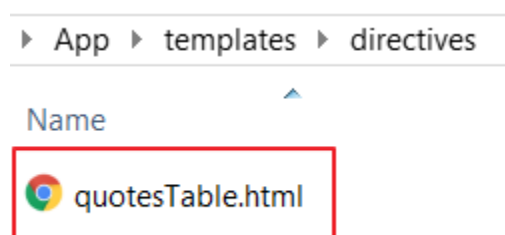


ตารางที่ 4.17 แม่พิมพ์ต้นแบบและแบบจำลองของ DirectiveTemplate

แม่พิมพ์ต้นแบบ	แบบจำลอง
	

ส่วนแม่พิมพ์ต้นแบบของ DirectiveTemplate คือโครงของ HTML tag ที่เอาไว้แสดงผลในส่วนของ Directive จะมีป้ายกำกับ (Tag) คือ

1) `<<DirectiveTemplate>>_templateFileName` เป็นการกำหนดชื่อของ AngularJS Directive Template file แต่ไม่มีกฎการแปลงเนื่องจากต้องมีการสร้าง File Template ขึ้นมาเพื่อเตรียมเก็บแม่พิมพ์ MarkupLanguage ดังภาพที่ 4.33



ภาพที่ 4.33 ไฟล์เทมเพลตหลังจากผ่านการนำแบบจำลอง DirectiveTemplate ไปใช้

## 4.2 การตรวจสอบกฎการแปลง

ในขั้นตอนนี้จะเป็นการตรวจสอบความครบถ้วนของกฎการแปลงสำหรับแปลงแบบจำลองซึ่งสร้างจากแม่พิมพ์ต้นแบบในโปรไฟล์แองกูลาร์เจเอสไปเป็นโค้ด โดยการใช้โปรไฟล์แองกูลาร์เจเอสซึ่งประกอบด้วย แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมโคเลเอ็นต์ และแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์เป็นข้อมูลสำหรับตรวจสอบความครบถ้วนของกฎการแปลง สำหรับวิธีการทวนสอบจะใช้แบบสำรวจรายการ (Checklist) ดังตารางที่ 4.18

ตารางที่ 4.18 แบบสำรวจรายการสำหรับตรวจสอบความครบถ้วนของกฎการแปลง

แม่พิมพ์ต้นแบบ	กฎการแปลง		หมายเหตุ
	หัวข้อที่	ไม่มี	
แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมโคลเอนต์			
<<AngularApp>>	4.1.4		
<<AppRouter>>	4.1.7		

ตารางที่ 4.18 แบบสำรวจรายการสำหรับตรวจสอบความครบถ้วนของกฎการแปลง (ต่อ)

แม่พิมพ์ต้นแบบ	กฎการแปลง		หมายเหตุ
	หัวข้อที่	ไม่มี	
<<AppView>>	4.1.8		
<<Controller>>	4.1.9		
<<CSS>>	4.1.3		
<<Directive>>	4.1.13		
<<DirectiveTemplate>>		X	ไม่มีกฎการแปลง เนื่องจากไม่มีส่วนในการแปลงเป็นโค้ด แต่จะเป็นการสร้าง Template File ของ Directive ขึ้นมาซึ่งสามารถดูได้จากหัวข้อ 4.1.17
<<DirectiveController>>	4.1.14		
<<DirectiveScope>>	4.1.16		
<<DirectiveIsolateScope>>	4.1.15		
<<Filter>>	4.1.5		
<<MarkupLanguage>>	4.1.12		
<<Scope>>	4.1.10		
<<ScriptLanguage>>	4.1.2		
<<Service>>	4.1.6		
<<Template>>		X	ไม่มีกฎการแปลง เนื่องจากไม่มีส่วนในการแปลงเป็นโค้ด แต่จะเป็นการสร้าง Template File ของ Directive ขึ้นมาซึ่งสามารถดูได้จากหัวข้อ 4.1.11
<<WebPage>>	4.1.1		

ตารางที่ 4.18 แบบสำรวจรายการสำหรับตรวจสอบความครบถ้วนของกฎการแปลง (ต่อ)

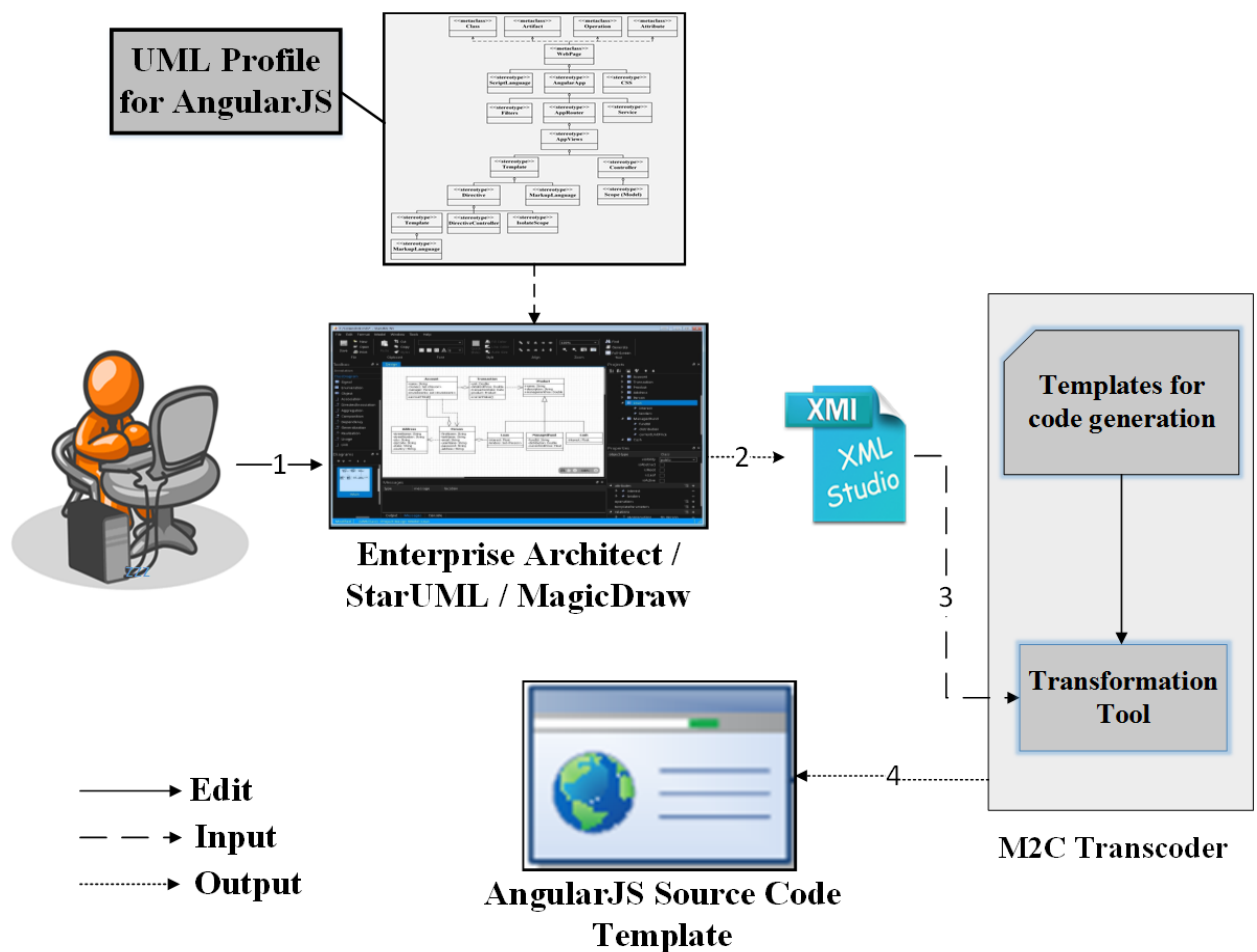
แม่พิมพ์ต้นแบบ	กฎการแปลง		หมายเหตุ
	หัวข้อที่	ไม่มี	
แม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมความสัมพันธ์			
<<Binding>>		X	ไม่มีกฎการแปลง เนื่องจากการทำงานภายในของเฟรมเวิร์กแองกูลาร์เจเอส และไม่มีส่วนในการแปลงเป็นโค้ด แต่จะเป็นความสัมพันธ์ระหว่าง 2 ชุดของแม่พิมพ์ต้นแบบ โดยชุดที่ 1 คือ ระหว่าง<<Template>> และ <<Scope>>ชุดที่ 2 คือ ระหว่าง<<DirectiveTemplate>> และ<<DirectiveScope>>
<<Passing>>		X	ไม่มีกฎการแปลง เนื่องจากการทำงานภายในของเฟรมเวิร์กแองกูลาร์เจเอส และไม่มีส่วนในการแปลงเป็นโค้ด แต่จะเป็นความสัมพันธ์ระหว่างแม่พิมพ์ต้นแบบ<<DirectiveIsolateScope>> และ<<DirectiveController>>

## บทที่ 5

### การออกแบบและพัฒนาเครื่องมือ

ในบทนี้จะกล่าวถึงการออกแบบและการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงสำหรับแปลงแบบจำลองซึ่งสร้างจากแม่พิมพ์ต้นแบบในโปรไฟล์แองกูลาร์เจเอสไปเป็นโค้ด

#### 5.1 ภาพรวมการพัฒนา



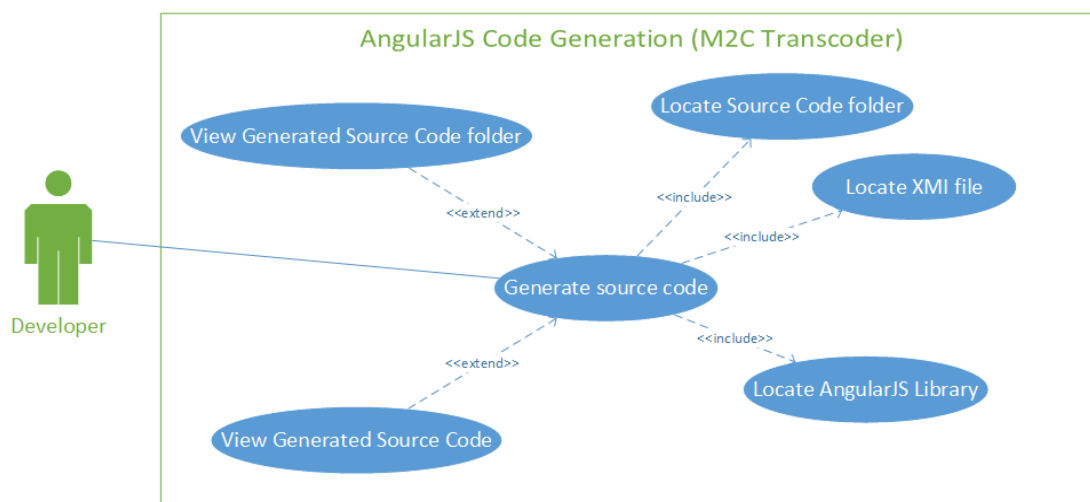
ภาพที่ 5.1 ภาพรวมแนวคิดการทำงานของเครื่องมือ

ภาพรวมของการพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสเป็นดังภาพที่ 5.1 เริ่มต้นจากนักพัฒนาทำการออกแบบแผนภาพคลาสของระบบเว็บแอปพลิเคชันตามยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส โดยใช้เครื่องมือสร้างแบบจำลอง (Modeling Tool) เช่น MagicDraw ซึ่งมีการสร้างยูเอ็มแอลโปรไฟล์สำหรับแองกูลาร์เจเอสไว้ในเครื่องนั้นแล้ว จากนั้นนำแบบจำลองในระดับพีเอสเอ็มที่ได้ซึ่งอยู่ในรูปของไฟล์ XMI นำเข้าไปยัง M2C Transcoder ซึ่งเป็นเครื่องมือการแปลง (Transformation Tool) ร่วมกับแผ่นแบบของการสร้างโค้ดเทมเพลต (Template for code generation) ซึ่งประกอบด้วยชุดของกฎการแปลง เพื่อใช้

สำหรับดำเนินการแปลงแบบจำลองพีเอสเอ็มที่ได้มาจากเครื่องมือสร้างแบบจำลอง ผลลัพธ์ที่ได้จากเครื่องมือการแปลงคือ โค้ดเทมเพลตของเว็บแอปพลิเคชันเฟรมเวิร์กแองกูลาร์เจเอส

## 5.2 การออกแบบหน้าที่การทำงานของเครื่องมือ

การออกแบบหน้าที่การทำงานของเครื่องมือมีวัตถุประสงค์เพื่อให้เห็นถึงความสามารถในการทำงานของเครื่องมือที่จะพัฒนา โดยใช้แบบจำลองเพื่อเป็นตัวแทนในการสื่อความหมายระหว่างผู้ออกแบบระบบและผู้ใช้งานระบบ ซึ่งในโครงงานมหาบัณฑิตนี้ เลือกใช้แผนภาพยูสเคส (Use Case Diagram) ในการวิเคราะห์และออกแบบเครื่องมือสำหรับแปลงแบบจำลองเป็นโค้ด ฟังก์ชันงานสำหรับแปลงโค้ดประกอบด้วยการนำเข้าไฟล์ XMI ซึ่งได้จากแบบจำลองของระบบงานที่ได้พัฒนาขึ้นในระดับเฉพาะเจาะจงกับแพลตฟอร์ม (PSM) โดยใช้ยูเอ็มแอลโปรไฟล์แองกูลาร์เจเอส กำหนดไลบรารีสำหรับแองกูลาร์เจเอส และเลือกไคเรกทอรีสำหรับจัดเก็บโค้ดหลังจากแปลงเสร็จ เมื่อการแปลงเสร็จเรียบร้อยแล้วผู้ใช้งานสามารถดูผลลัพธ์ของการแปลงได้ 2 แบบคือ การดูโค้ดหลังจากแปลงเสร็จ และการดูไฟล์เตอร์ที่เก็บโค้ด โดยสามารถแสดงเป็นแผนภาพยูสเคสของเครื่องมือสำหรับการแปลงแบบจำลองเป็นโค้ด ดังภาพที่ 5.2 ส่วนคำอธิบายยูสเคสแสดงในภาคผนวก ก



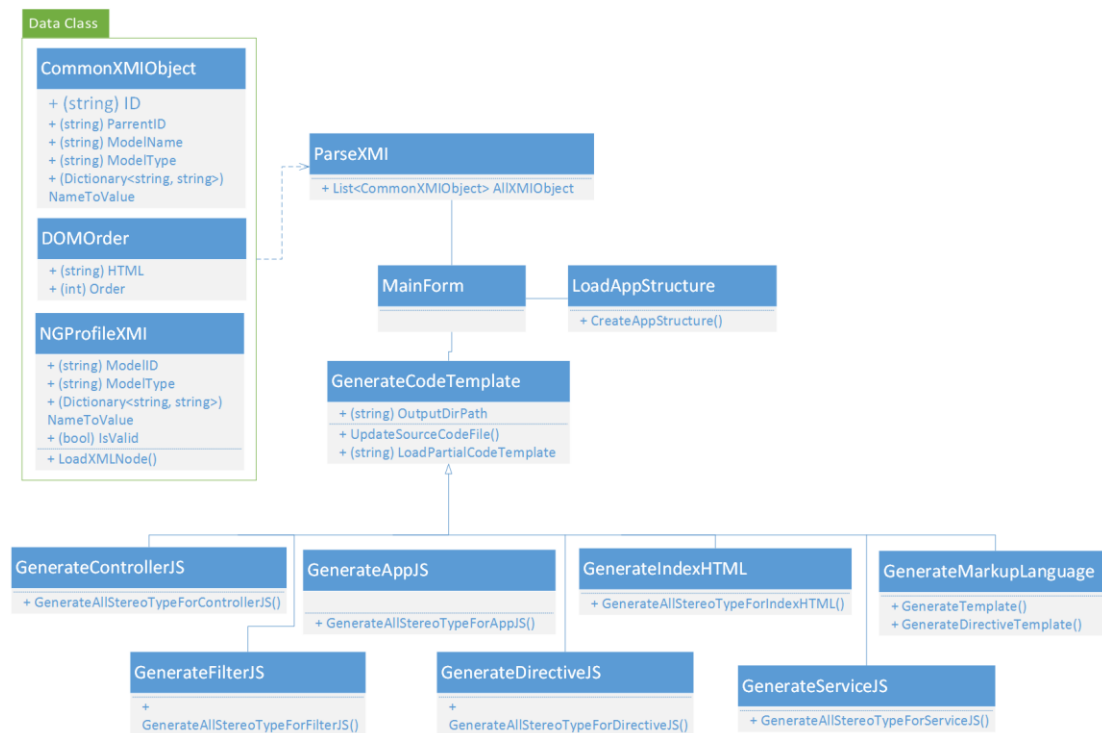
ภาพที่ 5.2 แผนภาพยูสเคสของเครื่องมือสำหรับการแปลงแบบจำลองเป็นโค้ด

## 5.3 การออกแบบเชิงแนวคิด

ในส่วนนี้จะเป็นการอธิบายโครงสร้างการจัดเก็บข้อมูลของเครื่องมือที่จะพัฒนาขึ้น เพื่อเป็นการนำเสนอสิ่งต่าง ๆ ในเครื่องมือและแสดงความสัมพันธ์ระหว่างสิ่งเหล่านั้นจึงนำเสนอโดยใช้แผนภาพคลาส ซึ่งแผนภาพคลาสจะแสดงให้เห็นถึงคลาสต่าง ๆ ที่มีอยู่ในเครื่องมือและความสัมพันธ์ระหว่างกัน ซึ่งประกอบด้วยคลาสทั้งหมด 13 คลาสแสดงดังภาพที่ 5.3 และมีคำอธิบายของแต่ละคลาสดังต่อไปนี้

- 1) คลาส LoadAppStructure ทำหน้าที่ในการสร้างโครงสร้างพื้นฐานของเว็บแอปพลิเคชัน ทั้งการสร้างแฟ้มข้อมูล (Directory) และ ไฟล์ (File) ที่จำเป็น
- 2) คลาส ParseXML ทำหน้าที่ในการอ่าน File XML ที่กำหนดและดำเนินการแปลง Tag ต่าง ๆ ของ File XML เป็น ชุดของออบเจ็กต์ CommonXMIOBJECT
- 3) คลาส GenerateIndexHTML ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของไฟล์ index.html ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ WebPage, ScriptLanguage, CSS และ AngularApp ไปปรับใช้
- 4) คลาส GenerateAppJS ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของไฟล์ app.js ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ AppRouter และ AppView ไปปรับใช้
- 5) คลาส GenerateFilterJS ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของไฟล์ filter.js ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ Filter ไปปรับใช้
- 6) คลาส GenerateServiceJS ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของไฟล์ service.js ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ Service ไปปรับใช้
- 7) คลาส GenerateControllerJS ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของ AngularJS View ทั้ง Controller และ Scope ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ Controller และ Scope ไปปรับใช้
- 8) คลาส GenerateMarkupLanguage ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของ Template และ DirectiveTemplate โดยทำหน้าที่สร้าง HTML Element ต่างๆ ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ Template, DirectiveTemplate และ MarkupLanguage ไปปรับใช้
- 9) คลาส GenerateDirectiveJS ทำหน้าที่ในการแปลงชุดของออบเจ็กต์ CommonXMIOBJECT ให้เป็นโค้ดเทมเพลตของ AngularJS Directive ทั้ง Directive, DirectiveController, DirectiveScope และ DirectivesolateScope ซึ่งมีการนำกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบ Directive, DirectiveController, DirectiveScope และ DirectivesolateScope ไปปรับใช้
- 10) คลาส CommonXMIOBJECT เป็นคลาสที่เป็นตัวแทนของข้อมูลหลังจากมีการแปลง Tag ต่าง ๆ จาก File XML
- 11) คลาส DOMOrder เป็นคลาสที่ทำหน้าที่จัดลำดับการแสดงผลของ HTML Element ว่า จะให้ HTML Element ใดแสดงผลก่อนหลัง

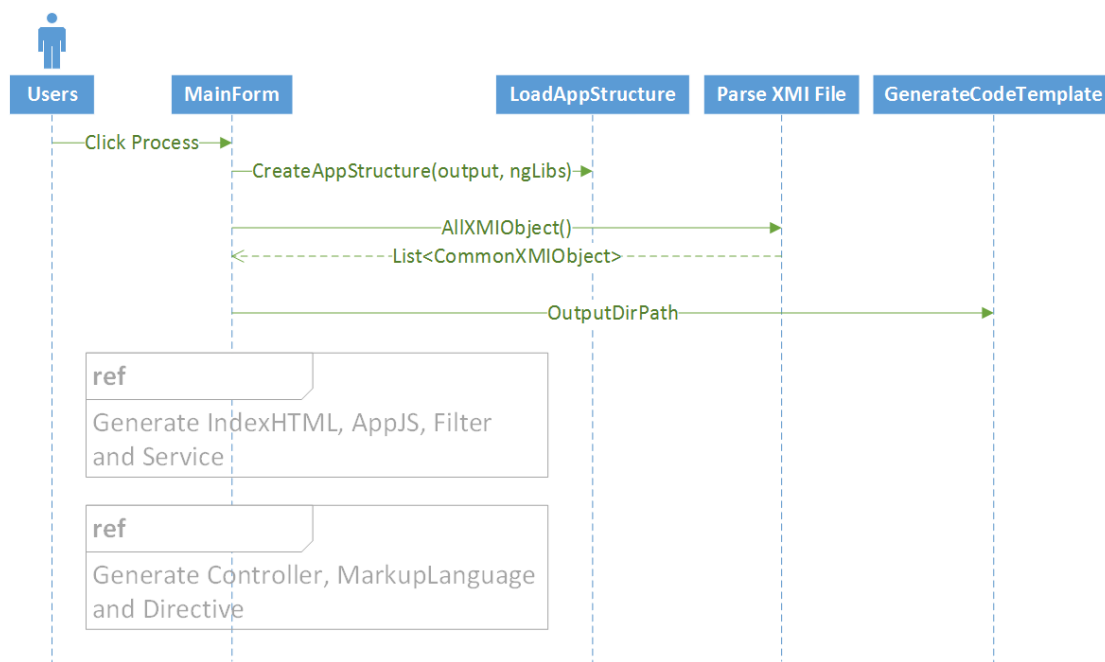
- 12) คลาส NGProfileXML เป็นคลาสที่ใช้เก็บค่าก่อนที่จะมีการแปลงไปเป็นโค้ดเทมเพลต
- 13) คลาส GenerateCodeTemplate ทำหน้าที่อ่าน File Template รวมถึงการบันทึกโค้ดเทมเพลตที่ผ่านการแปลงเรียบร้อยแล้วไปยัง Output directory ที่ได้กำหนดไว้



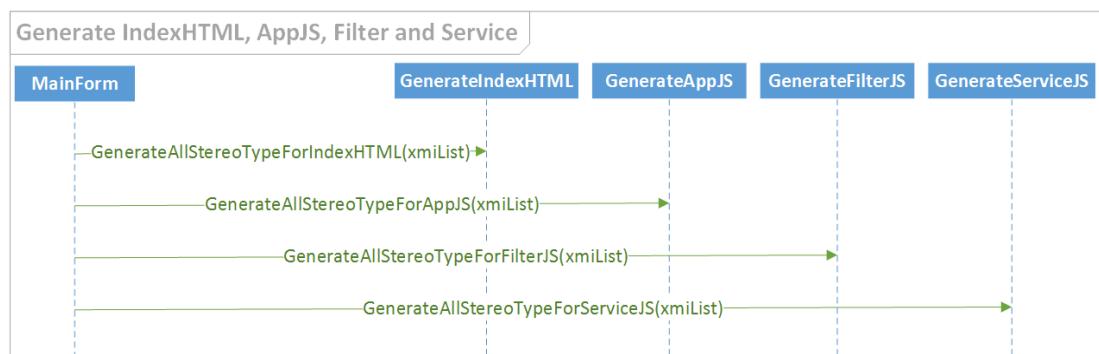
ภาพที่ 5.3 แผนภาพคลาสของเครื่องมือสำหรับการแปลงแบบจำลองเป็นโค้ด

#### 5.4 ลำดับการทำงานของคลาสต่าง ๆ ของเครื่องมือ

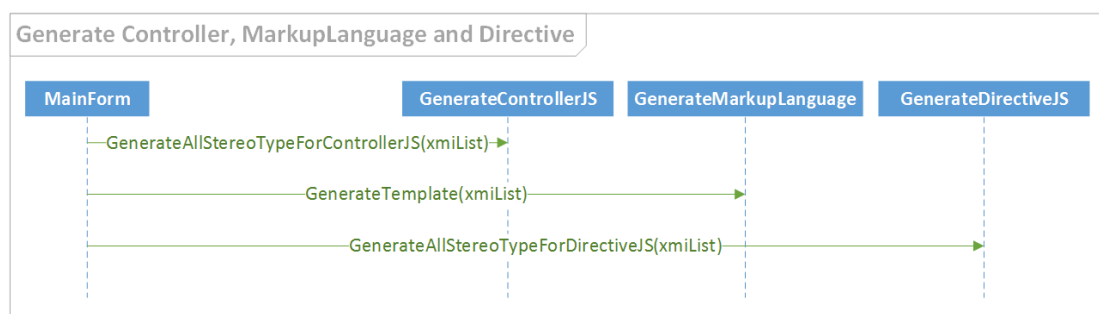
ในส่วนนี้จะเป็นการอธิบายลำดับการทำงานของคลาสต่างๆ ของเครื่องมือ โดยมีลำดับการทำงานดังภาพที่ 5.4, 5.5 และ 5.6 ตามลำดับ โดยเริ่มจากการอ่านไฟล์ XMI ขึ้นมาแล้วทำการ Parse Tag ต่างๆ ในไฟล์ XMI ให้เป็นรายการ (List) ของออบเจ็กต์ CommonXMIOObject จากนั้นจึงมีการกำหนดค่าเส้นทางของโฟลเดอร์ปลายทางสำหรับเก็บโค้ดเทมเพลต จากนั้นจึงเริ่มทำการอ่านค่าของรายการของออบเจ็กต์ CommonXMIOObject ที่มีคุณสมบัติ (Property) ชื่อ ModelType ตรงกับค่าของกฎการแปลงแบบจำลองของแม่พิมพ์ต้นแบบในแต่ละกฎตามลำดับ โดยเริ่มจากแม่พิมพ์ต้นแบบ WebPage, CSS, ScriptLanguage, AngularApp, AppRouter, AppView, Filter, Service, Controller, Scope, Template, DirectiveController, DirectiveScope, DirectiveTemplate และ DirectivesolateScope



ภาพที่ 5.4 ลำดับการทำงานของคลาสต่างๆ ของเครื่องมือ



ภาพที่ 5.5 ลำดับการทำงานของคลาสต่างๆ ของเครื่องมือ (ต่อ)



ภาพที่ 5.6 ลำดับการทำงานของคลาสต่างๆ ของเครื่องมือ (ต่อ)



## 5.5 ขั้นตอนการแปลงค่าจากไฟล์ XMI เป็นชุดของออบเจกต์ CommonXMIOObject

ขั้นตอนนี้เป็นการอ่านค่าในไฟล์ XMI ขึ้นมาและทำการแปลงค่าหรือ Tag ต่าง ๆ ในไฟล์ XMI ให้อยู่ในรูปของชุดของออบเจกต์ CommonXMIOObject เพื่อที่จะถูกนำไปใช้ในการแปลงชุดของออบเจกต์ CommonXMIOObject ให้เป็นโค้ดเทมเพลตต่อไปซึ่ง Tag ต่าง ๆ ในไฟล์ XMI ที่สามารถนำไปใช้ได้นั้นจะมีทั้งหมด 8 จุดดังภาพที่ 5.7 และ 5.8 ซึ่งคลาส ParseXMI จะทำหน้าที่ในการแปลงค่าในแต่ละตำแหน่งของไฟล์ XMI ให้อยู่ในรูปชุดของออบเจกต์ CommonXMIOObject โดยมีหลักการทำงานตามลำดับดังนี้

- 1) อ่านไฟล์ XMI ที่ Element ที่ชื่อว่า packagedElement
- 2) อ่านค่า Tag ที่ชื่อว่า xmi:id ตามตำแหน่งที่ 1. ในภาพที่ 5.7 มากำหนดเป็น ID ของออบเจกต์ CommonXMIOObject
- 3) อ่านค่า Tag ที่ชื่อว่า name ตามตำแหน่งที่ 2. ในภาพที่ 5.7 มากำหนดเป็น ModelName ของออบเจกต์ CommonXMIOObject
- 4) อ่านค่า Tag ที่ชื่อว่า general ตามตำแหน่งที่ 3. ในภาพที่ 5.7 มากำหนดเป็น ParentID ของออบเจกต์ CommonXMIOObject
- 5) อ่านค่า Tag ที่ชื่อว่า name ตามตำแหน่งที่ 4. ในภาพที่ 5.7 มากำหนดเป็นค่า Key ของ Dictionary ของออบเจกต์ CommonXMIOObject
- 6) อ่านค่า Tag ที่ชื่อว่า value ตามตำแหน่งที่ 5. ในภาพที่ 5.7 มากำหนดเป็นค่า Value ของ Dictionary ของออบเจกต์ CommonXMIOObject
- 7) นำค่า ID จาก 2) มาหา XMI Element ที่ขึ้นต้นด้วย AngularJS และมีค่าของ base\_Class (ตามตำแหน่งที่ 7 ในภาพที่ 5.8) ตรงกับ ID จาก 2) ดังแสดงตามตำแหน่งที่ 6 ในภาพที่ 5.8 เมื่อได้มาแล้วจึงทำการแบ่งชื่อของ XMI Element AngularJS โดยนำค่าที่อยู่หลัง : มากำหนดเป็น ModelType ของออบเจกต์ CommonXMIOObject ซึ่งจากรูปจะได้ค่า ModelType คือ CSS
- 8) ถ้าหากไม่มีค่าในข้อ 5) และ 6) ก็จะนำค่า Default value จากตำแหน่งที่ 8 ในภาพที่ 5.8 โดยค่าที่อยู่ด้านหน้าเครื่องหมาย = จะถูกกำหนดเป็นค่า Key และค่าที่อยู่หลังเครื่องหมาย = จะถูกกำหนดมาเป็นค่า Value ของ Dictionary ของออบเจกต์ CommonXMIOObject

```

<packagedElement xmi:type='uml:Class' xmi:id='_18_2_649022e_1464281651611_486732_4572' name='QuotePriceMainCSS'>
  <generalization xmi:type='uml:Generalization' xmi:id='_18_2_649022e_1464283719931_38333_5203' general='_18_2_649022e_1464281445017_262706_447'
  <ownedAttribute xmi:type='uml:Property' xmi:id='_18_2_649022e_1464281687265_889407_4601' name='href' visibility='public'>
    <type href='http://www.omg.org/spec/UML/20131001/PrimitiveTypes.xmi#String'>
      <xmi:Extension extender='MagicDraw UML 18.2'>
        <referenceExtension referentPath='UML Standard Profile::UML2 Metamodel::PrimitiveTypes::String' referentType='PrimitiveType'>
        </xmi:Extension>
      </type>
      <xmi:Extension extender='MagicDraw UML 18.2'>
        <modelExtension>
          <lowerValue xmi:type='uml:LiteralInteger' xmi:id='_18_2_649022e_1464281726391_742768_4604' value='1'>
          </modelExtension>
        </xmi:Extension>
        <xmi:Extension extender='MagicDraw UML 18.2'>
          <modelExtension>
            <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='_18_2_649022e_1464281726392_882676_4605' value='1'>
            </modelExtension>
          </xmi:Extension>
          <defaultValue xmi:type='uml:LiteralString' xmi:id='_18_2_649022e_1464281725099_861254_4603' value='/css/bootstrap.css'>
          </ownedAttribute>
        </packagedElement>

```

1. 2. 3. 4. 5.

ภาพที่ 5.7 Tag ต่าง ๆ ในไฟล์ XMI

```

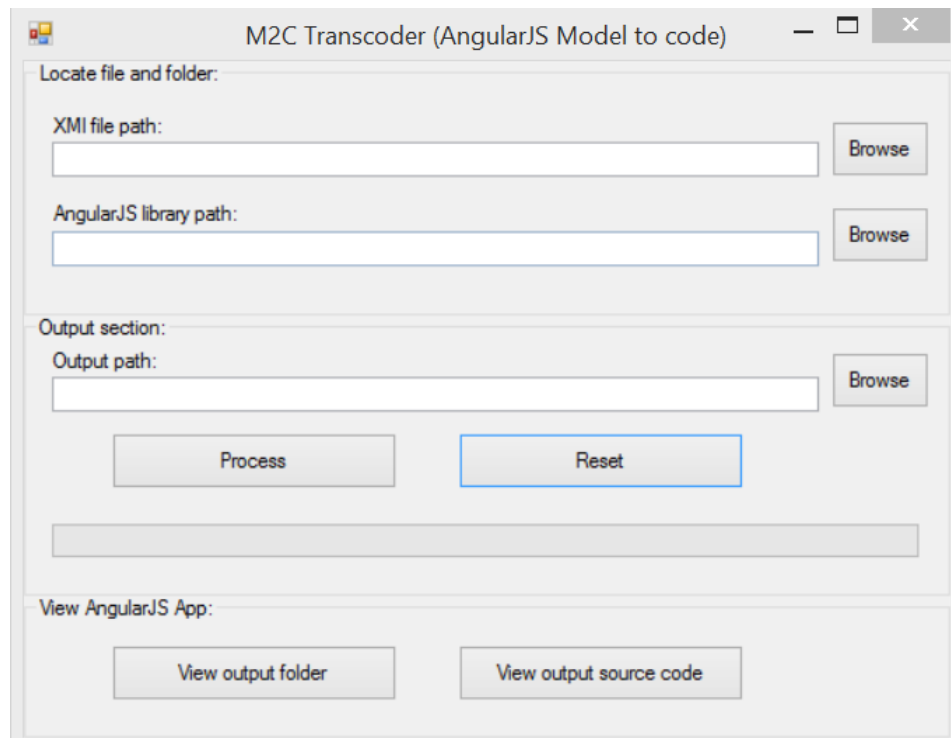
<xmi:Extension extender='MagicDraw UML 18.2'>
  <stereotypesHREFS>...</stereotypesHREFS>
</xmi:Extension>
<AngularJS:WebPage xmi:id='_18_2_649022e_1464281499769_260592_4503' base_Class='_18_2_649022e_1464281445017_262706_447' title='Angular JS P
<AngularJS:CSS xmi:id='_18_2_649022e_1464281632784_374452_4568' base_Class='_18_2_649022e_1464281611871_538741_4542' href='/css/app.css'>
<AngularJS:AngularApp xmi:id='_18_2_649022e_1464281590259_400350_4538' base_Class='_18_2_649022e_1464281562211_343959_4512' appName='App'>

```

6. 7. 8.

ภาพที่ 5.8 Tag ต่าง ๆ ในไฟล์ XMI (ต่อ)

## 5.6 การออกแบบส่วนต่อประสานของเครื่องมือ



ภาพที่ 5.9 การออกแบบส่วนต่อประสานของเครื่องมือ

เครื่องมือการแปลงแบบจำลองเป็นโค้ดนั้นพัฒนาขึ้นในรูปแบบของไมโครซอฟต์แวร์ดอทเน็ตแอปพลิเคชันโดยสามารถรองรับการแปลงแบบจำลองที่กำหนดขึ้น จากภาพที่ 5.9 การออกแบบส่วน

ต่อประสานของเครื่องมือที่ประยุกต์ใช้กฎการแปลงสำหรับแปลงแบบจำลอง ซึ่งสร้างจากแม่พิมพ์ต้นแบบในโปรไฟล์แองกูลาร์เจเอสไปเป็นโค้ด ประกอบด้วย 2 ส่วน ดังนี้

#### 5.6.1 การออกแบบส่วนต่อประสานสำหรับข้อมูลนำเข้า

การออกแบบการนำเข้าข้อมูลเป็นส่วนที่จะรับข้อมูลนำเข้าจากผู้ใช้งาน สำหรับเป็นข้อมูลที่ต้องใช้ในการทำการแปลงจากเครื่องมือ คือ แบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์มของโปรไฟล์แองกูลาร์เจเอสในรูปแบบไฟล์ XMI ไบรารีของแองกูลาร์เจเอส และโพลเดอร์สำหรับจัดเก็บโค้ดหลังจากแปลงเสร็จ

#### 5.6.2 การออกแบบส่วนต่อประสานสำหรับการแสดงผลลัพธ์

การออกแบบการแสดงผลลัพธ์เป็นส่วนที่ทำหน้าที่แสดงผลลัพธ์ที่ได้จากการแปลง ซึ่งประกอบด้วย 2 ส่วน คือ 1) การแสดงผลลัพธ์ของการดูโค้ดหลังจากแปลงเสร็จ ได้ออกแบบเป็นลำดับขั้นเพื่อเลือกดูไฟล์ที่ต้องการ เมื่อเลือกแล้วจะแสดงเนื้อหาของไฟล์ที่เลือก 2) การแสดงผลลัพธ์โดยการเปิดโพลเดอร์ที่เก็บไฟล์ผลลัพธ์

### 5.7 เครื่องมือที่ใช้ในการพัฒนาระบบ

#### 5.7.1 ฮาร์ดแวร์ที่ใช้ในการพัฒนาระบบ

- 1) เครื่องคอมพิวเตอร์ที่ใช้พัฒนาระบบ
  - หน่วยประมวลผล อินเทล คอร์ไอ 7-4500 ยู ความเร็ว 1.8 กิกะเฮิรตซ์
  - หน่วยความจำ ดีดีอาร์ 3 ขนาด 8.00 กิกะไบต์
  - ฮาร์ดดิสก์ ความจุ 1 เทราไบต์
  - จอภาพ 15 นิ้ว

#### 5.7.2 ซอฟต์แวร์ที่ใช้ในการพัฒนาระบบ

- 1) ระบบปฏิบัติการ
  - ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ 8 (Microsoft Window 8)
- 2) เครื่องมือที่ใช้ในการออกแบบและจัดทำเอกสารของกระบวนการ
  - ไมโครซอฟท์ออฟฟิศ รุ่น 365 (Microsoft Office 365)
  - ไมโครซอฟท์วิสิโอ รุ่น 2013 (Microsoft Visio 2013)
  - เมจิกดรอ รุ่น 18.2 (MagicDraw 18.2)
- 3) เครื่องมือที่ใช้ในการพัฒนาเครื่องมือ
  - ไมโครซอฟท์วิซวลสตูดิโอ รุ่น 2015 (Microsoft Visual Studio Community

2015)

## 5.8 การพัฒนาระบบ

การพัฒนาระบบเป็นการพัฒนาชุดคำสั่งตามที่ได้วิเคราะห์และออกแบบไว้ รวมถึงได้ติดตั้งเครื่องมือการพัฒนาระบบเรียบร้อยแล้ว สามารถอธิบายส่วนการพัฒนาได้ 2 ส่วน ดังนี้

### 5.8.1 ขั้นตอนการพัฒนาระบบ

หลังจากที่ได้ทำการวิเคราะห์ และ ออกแบบระบบ รวมถึงการติดตั้งเครื่องมือที่ใช้ในการพัฒนาระบบเรียบร้อยแล้ว ในขั้นตอนนี้กล่าวถึงขั้นตอนการพัฒนาส่วนของซอฟต์แวร์สำหรับการแปลงแบบจำลองเป็นโค้ด ซึ่งสามารถอธิบายเป็นขั้นตอน ดังต่อไปนี้

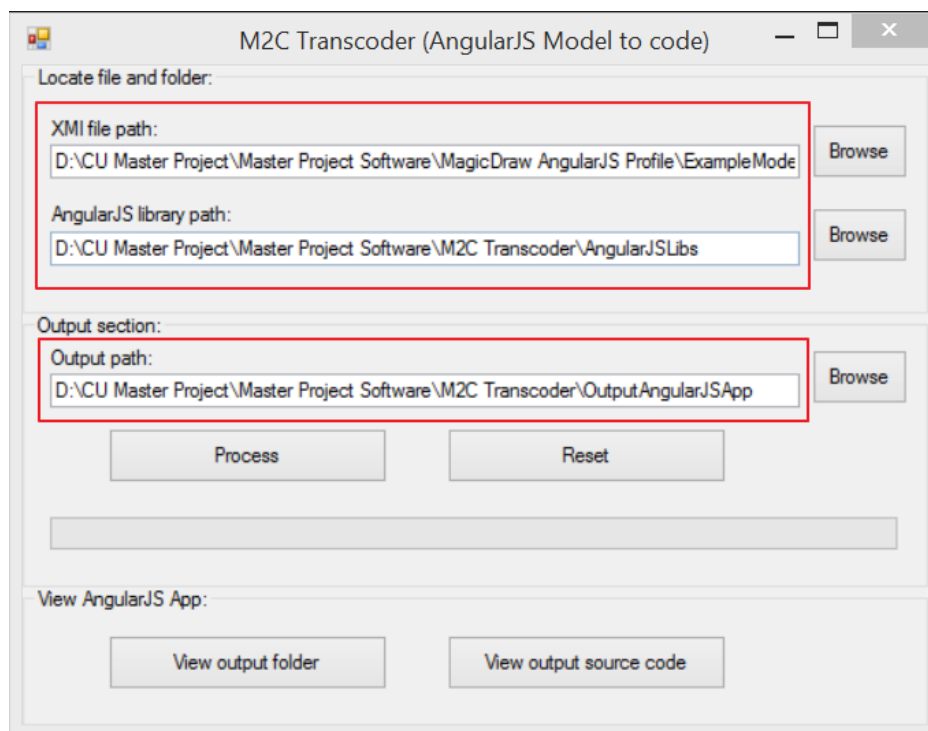
1) ส่วนของการนำเข้าข้อมูล เป็นการนำเข้าแบบจำลองของโปรไฟล์แองกูลาร์เจเอส ในรูปแบบไฟล์ XMI กำหนดไลบรารีสำหรับแองกูลาร์เจเอส และข้อมูลเกี่ยวกับโพลเดอร์สำหรับจัดเก็บโค้ดหลังจากแปลงเสร็จ ซึ่งเป็นข้อมูลที่ต้องใช้ในการทำงานของเครื่องมือ

2) ส่วนของการแปลงไปเป็นโค้ด จะทำการอ่านข้อมูลแบบจำลองในรูปแบบไฟล์ XMI ผลลัพธ์ที่ได้จะทราบว่าแบบจำลองที่ออกแบบมีการระบุด้วยแม่พิมพ์ต้นแบบใดบ้าง จากนั้นทำการแปลงแม่พิมพ์ต้นแบบดังกล่าวด้วยกฎการแปลงที่กล่าวในบทที่ 4 ผลลัพธ์ที่ได้จากการแปลงคือ ไฟล์เอกสารนามสกุล \*.js และ \*.html

3) ส่วนของการแสดงผล ในการแสดงผลข้อมูลที่ได้หลังจากการแปลงเสร็จแล้ว เพื่อให้สะดวกแก่การใช้งานของผู้ใช้ จึงแสดงผลได้ใน 2 แบบคือ เปิดดูโค้ดที่สมบูรณ์ และเปิดโพลเดอร์ปลายทาง

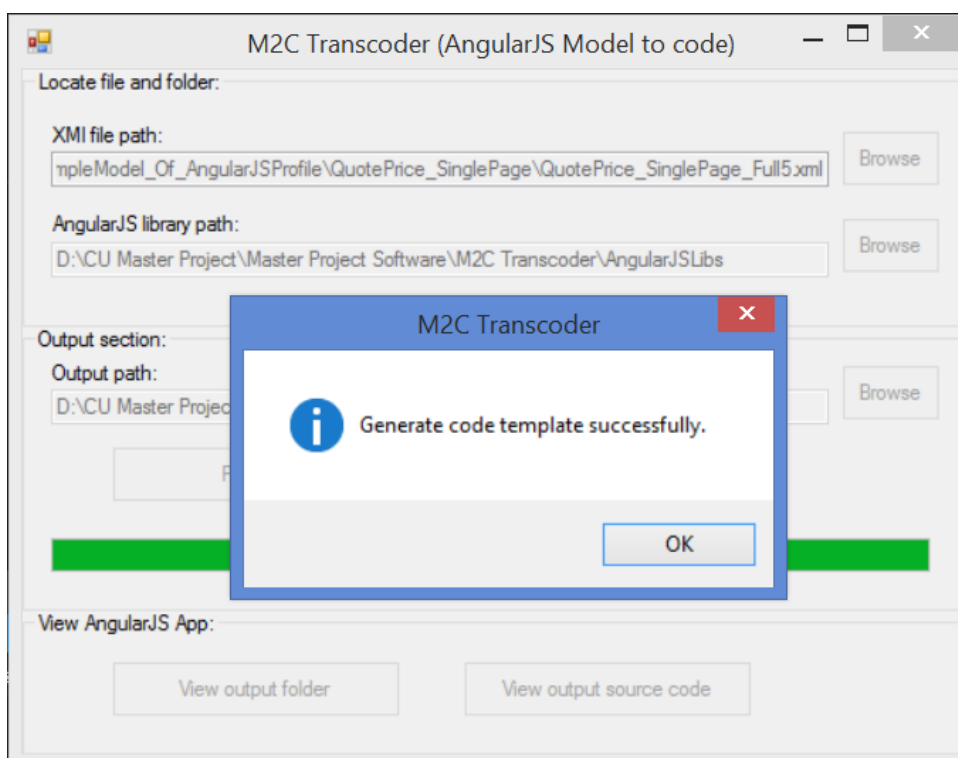
### 5.8.2 ตัวอย่างการใช้งานระบบ

1) เริ่มจากการแบบจำลองที่เฉพาะเจาะจงกับแพลตฟอร์มของโปรไฟล์แองกูลาร์เจเอส ในรูปแบบไฟล์ XMI กำหนดไลบรารีสำหรับแองกูลาร์เจเอส และไดเรกทอรีสำหรับจัดเก็บโค้ดหลังจากแปลงเสร็จ ดังภาพที่ 5.10



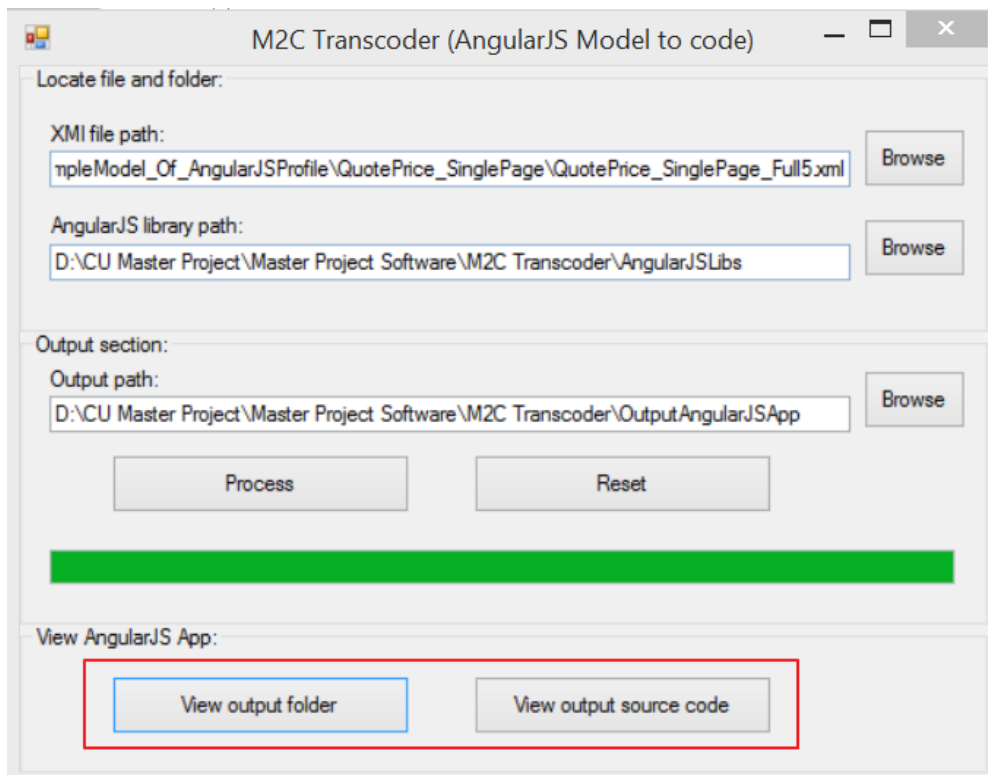
ภาพที่ 5.10 การระบุข้อมูลนำเข้า

2) จากนั้นผู้ใช้งานทำการคลิกปุ่ม “Process” เพื่อทำการแปลงแบบจำลองไปเป็นโค้ด เมื่อโปรแกรมกำลังทำการแปลงแบบจำลองจะมี Progress bar แสดงความคืบหน้าของการทำงานของเครื่องมือเป็นระยะๆ และเมื่อทำการแปลงเสร็จเรียบร้อยแล้วจะมีป๊อปอัพแสดงว่าเครื่องมือทำงานเสร็จเรียบร้อยแล้ว ดังภาพที่ 5.11

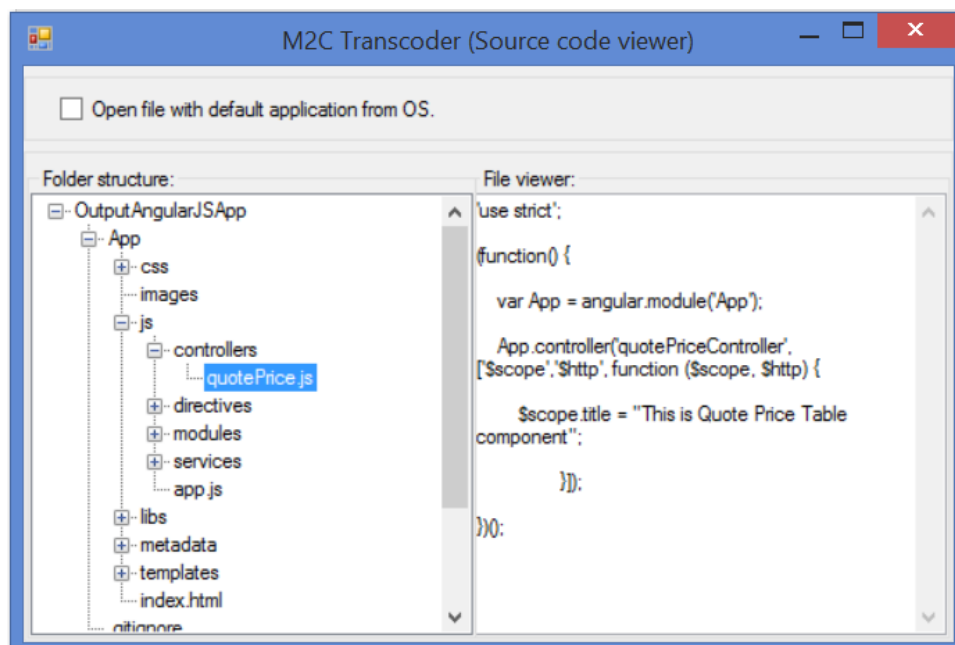


ภาพที่ 5.11 การแสดงผลเมื่อการทำงานของเครื่องมือเสร็จสิ้น

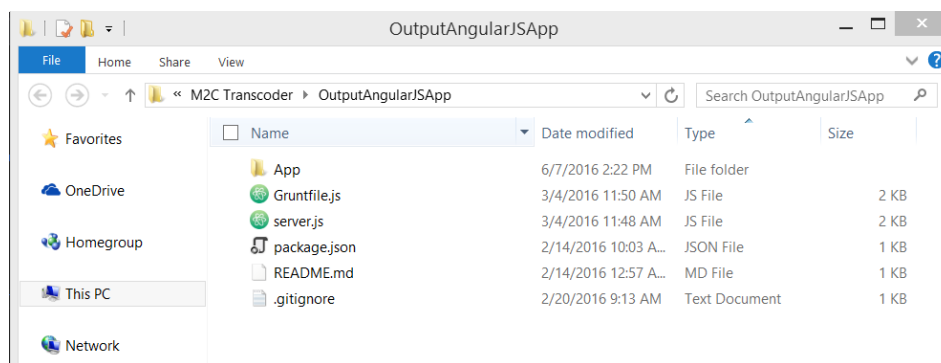
3) หลังจากเครื่องมือทำการประมวลผลเสร็จสิ้นผู้ใช้งานสามารถ เลือกดูผลลัพธ์ได้ 2 ลักษณะ เป็นดังภาพที่ 5.12 คือ กดที่ปุ่ม “View output folder” จะเป็นการเปิด Folder ของผลลัพธ์ที่ได้ และเมื่อกดปุ่ม “View output source code” จะเป็นการเปิดหน้าต่างของเครื่องมือขึ้นมาใหม่เพื่ออำนวยความสะดวกให้ผู้ใช้งานสามารถเปิดดูโค้ดได้อย่างทันที ผลลัพธ์การแสดงผลข้อมูลที่ได้หลังจากการแปลงเสร็จแล้ว เป็นดังภาพที่ 5.13 และ 5.14 ตามลำดับ



ภาพที่ 5.12 การแสดงผลลัพธ์ของส่วนเปิดดูผลลัพธ์ของเครื่องมือ



ภาพที่ 5.13 การแสดงผลลัพธ์ของการดูโค้ดหลังจากแปลงเสร็จ



ภาพที่ 5.14 การแสดงผลลัพธ์ของการเปิดไฟล์เดอร์ที่เก็บโค้ด

4) หลังจากแปลงโค้ดเรียบร้อยแล้ว ในขั้นตอนนี้ขอยกตัวอย่างการเพิ่มเติมโค้ดให้เป็นโค้ดที่สามารถทำงานได้ โดยจะยกตัวอย่างไฟล์ `quotesTable.js` ซึ่งเป็น AngularJS Directive ดังแสดงในตารางที่ 5.1 สังเกตว่าโค้ดเดิมเพลตจะยังไม่สามารถทำงานได้ จากนั้นจึงทำการเพิ่มโค้ดในบางส่วนเพื่อให้ดึงข้อมูลมาแสดงผลบนเว็บแอปพลิเคชันได้เพื่อให้เป็นโค้ดที่สามารถทำงานได้ ดังแสดงในกรอบสี่เหลี่ยม

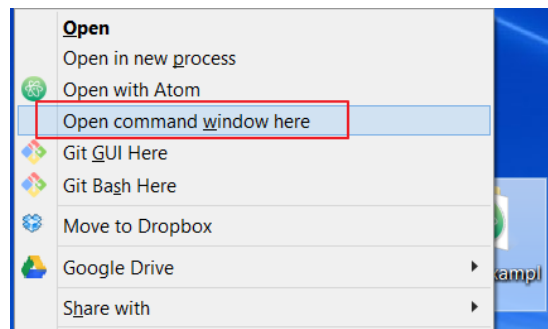
ตารางที่ 5.1 การเพิ่มเติมโค้ดเดิมเพลตให้เป็นโค้ดที่สามารถทำงานได้

โค้ดเดิมเพลต	โค้ดที่สามารถทำงานได้
<pre>App.controller('quotesTableController', ['\$scope', '\$http', '\$stateParams', function (\$scope, \$http, \$stateParams) {      \$scope.items = null;  }]);</pre>	<pre>App.controller('quotesTableController', ['\$scope', '\$http', '\$stateParams', function (\$scope, \$http, \$stateParams) {      \$scope.items = null;      \$http.get('metadata/stockPrice.json').then(function(res){         \$scope.items = res.data.stock;     });  }]);</pre>

5) ขั้นตอนต่อไปเป็นการทำให้เว็บแอปพลิเคชันสามารถแสดงผลได้ซึ่งในโครงการขั้นนี้ ได้สร้างเว็บเซิร์ฟเวอร์จำลองโดยใช้ `express` ซึ่งเป็น Library บน Node Package Manager (NPM) [19] ในการสร้างเว็บเซิร์ฟเวอร์จำลอง สำหรับการทำให้แอปพลิเคชันสามารถแสดงผลได้ มีขั้นตอนดังนี้

- Install Node.js
- หลังจากลง Node.js เรียบร้อยแล้วให้ทำการคลิกขวาเลือกเปิด Command window ดังภาพที่ 5.15





ภาพที่ 5.15 การเลือกเปิด Command ของ Window

- ใน Command Window ได้อะลือกพิมพ์คำสั่ง npm install ดังภาพที่ 5.16

```
C:\Users\Wutty\Desktop\Example>npm install
npm WARN deprecated lodash@2.4.2: lodash@<3.0.0 is no longer maintained. Upgrade
to lodash@^4.0.0.
npm WARN deprecated lodash@0.9.2: Grunt needs your help! See https://github.com/
gruntjs/grunt/issues/1403.
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail o
n node releases >= v7.0. Please update to graceful-fs@^4.0.0 as soon as possible
. Use 'npm ls graceful-fs' to find it in the tree.
npm WARN deprecated lodash@1.0.2: lodash@<3.0.0 is no longer maintained. Upgrade
to lodash@^4.0.0.
```

ภาพที่ 5.16 การลง Package ของ NPM

- เมื่อทำการลง Package ของ NPM เรียบร้อยแล้วให้ทำการ Run คำสั่ง grunt serve เพื่อให้เว็บเซิร์ฟเวอร์มีการทำงาน จะได้ผลลัพธ์ดังภาพที่ 5.17

```
grunt
C:\Users\Wutty\Desktop\Fully_Example>grunt serve
Running "express:dev" (express) task
Starting background Express server
My App listening on port9999
http://localhost:9999

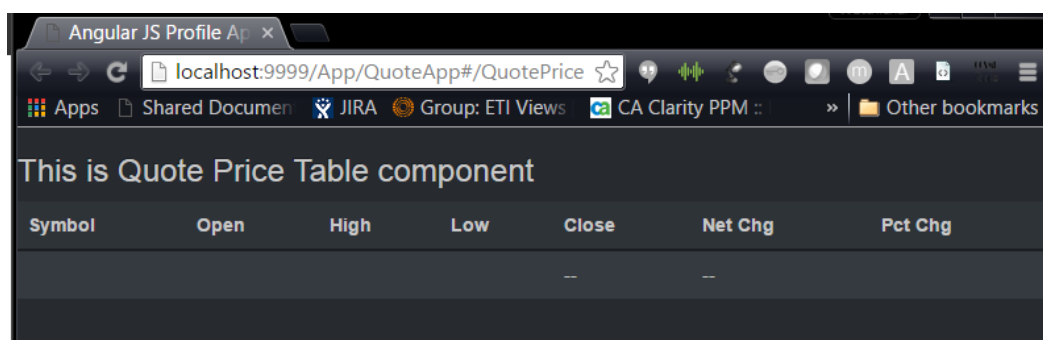
Running "open:app" (open) task

Running "watch" task
Waiting...
>> Redirect QuoteApp /App/QuoteApp
express deprecated res.sendFile: Use res.sendFile instead server.js:19:7
>> Load all css files /css/app.css
```

ภาพที่ 5.17 การเริ่มทำงานของเว็บเซิร์ฟเวอร์จำลอง

- เมื่อเว็บเซิร์ฟเวอร์จำลองทำงาน จะทำการเปิดเว็บแอปพลิเคชันให้โดยอัตโนมัติ ดัง

ภาพที่ 5.18



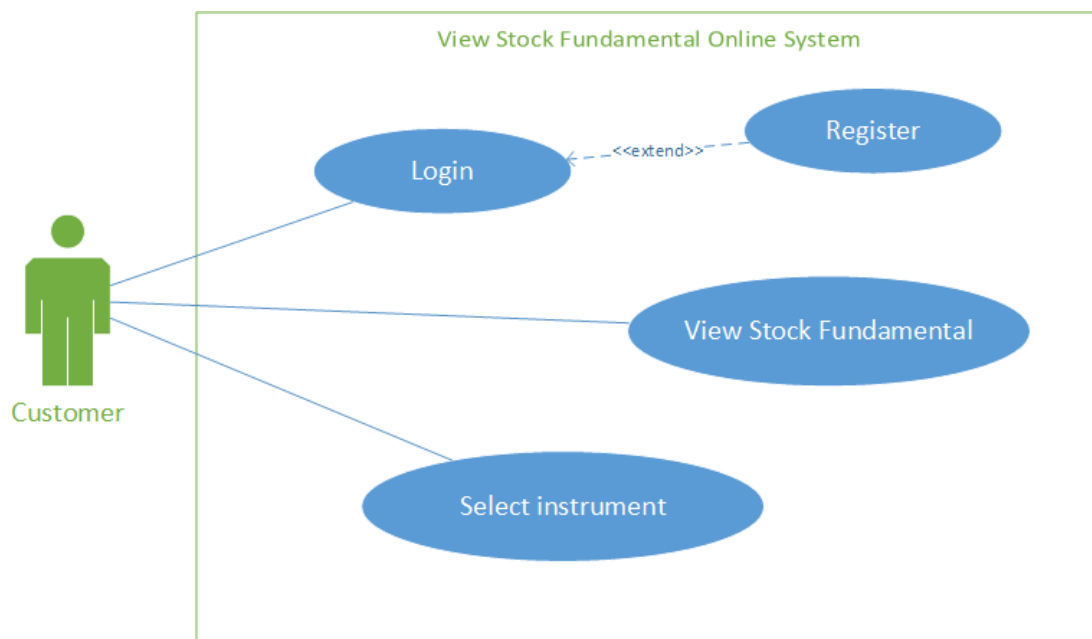
ภาพที่ 5.18 เว็บแอปพลิเคชันถูกเปิดโดยอัตโนมัติ

## บทที่ 6

### การทดสอบและประเมินผล

ในบทนี้จะอธิบายรายละเอียดของการทดสอบและการประเมินผลของการออกแบบกฎการแปลงสำหรับแปลงแบบจำลองที่สร้างจากแม่พิมพ์ต้นแบบในโปรไฟล์แองกูลาร์เจเอสไปเป็นโค้ด โดยจะประยุกต์ใช้กฎที่ได้นิยามไว้นามาทดสอบกับกรณีศึกษาระบบข้อมูลพื้นฐานของหุ้นออนไลน์ (View Stock Fundamental Online System)

#### 6.1 การพัฒนาระบบข้อมูลพื้นฐานของหุ้นออนไลน์



ภาพที่ 6.1 แผนภาพยูสเคสของระบบข้อมูลพื้นฐานของหุ้นออนไลน์

จากภาพที่ 6.1 ระบบข้อมูลพื้นฐานของหุ้นออนไลน์ เป็นเว็บแอปพลิเคชันที่ให้ผู้ใช้งานสามารถดูข้อมูลพื้นฐานของหุ้นผ่านทางเว็บไซต์ โดยจะยกตัวอย่างหน้าจกระบบข้อมูลพื้นฐานของหุ้นออนไลน์ ดังภาพที่ 6.2

Menu

Home

Company Information

Quote Price

This is Quote Price Table component

Symbol	Open	High	Low	Close	Net Chg	Pct Chg
PTT	234.000	238.000	233.000	235.000	0.000	0.000%
TOP	59.000	60.250	58.000	59.000	--	--
ADVANC	167.500	168.500	167.000	167.000	0.500	0.300%
DTAC	32.000	32.500	31.750	32.000	0.250	0.790%
TRUE	6.100	6.200	6.050	6.100	0.050	0.830%

ภาพที่ 6.2 ตัวอย่างภาพหน้าจอของระบบข้อมูลพื้นฐานของหุ้นออนไลน์

จากกรณีศึกษาซึ่งคือระบบข้อมูลพื้นฐานของหุ้นออนไลน์ ทางผู้วิจัยได้มีการสร้างยูเอเอ็มแอลโปรไฟล์ของแองกูลาร์เจเอสขึ้นมาผ่านทางโปรแกรม MagicDraw (Version 18.2) ได้ผลลัพธ์ดังภาพที่ 6.3 ซึ่งหลังจากได้ยูเอเอ็มแอลโปรไฟล์ของแองกูลาร์เจเอสขึ้นมาแล้วจึงได้ทำการสร้างแบบจำลองเป็นแผนภาพคลาสของระบบกรณีศึกษาจากยูเอเอ็มแอลโปรไฟล์ของแองกูลาร์เจเอสได้ดังภาพที่ 6.4 ซึ่งแผนภาพคลาสที่ได้มานั้นสามารถแบ่งออกเป็น 4 ส่วนประกอบด้วยกัน

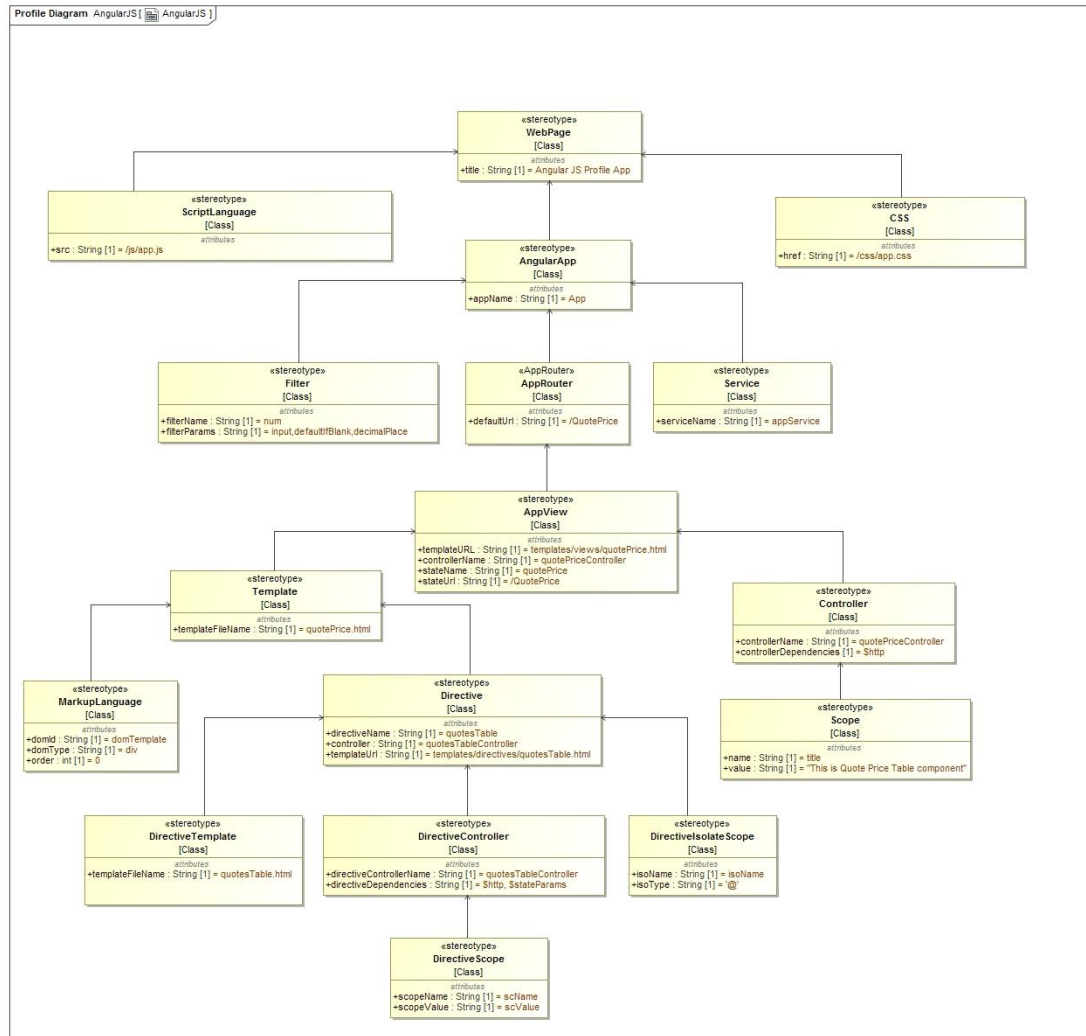
2) ส่วนโครงสร้างหลักของแองกูลาร์เจเอสเว็บแอปพลิเคชันซึ่งคือตำแหน่ง 1. ในภาพที่ 6.4 จะเป็นการอิมพอร์ตไลบรารีต่าง ๆ ที่ต้องใช้ในแองกูลาร์เจเอสเว็บแอปพลิเคชันรวมถึงโครงสร้างของไฟล์ index.html ดังแสดงในภาพที่ 6.5

3) ส่วนโครงสร้างของแองกูลาร์เจเอส View ซึ่งคือตำแหน่ง 2. ในภาพที่ 6.4 จะเป็นการประกาศหน้าเว็บเพจหลักและ URL ที่จะแสดงผลของหน้าเว็บเพจรวมถึงแองกูลาร์เจเอส Template หรือโครงสร้างของ HTML ที่จะประกอบเป็นโครงสร้างของเว็บเพจหลัก ดังแสดงในภาพที่ 6.6

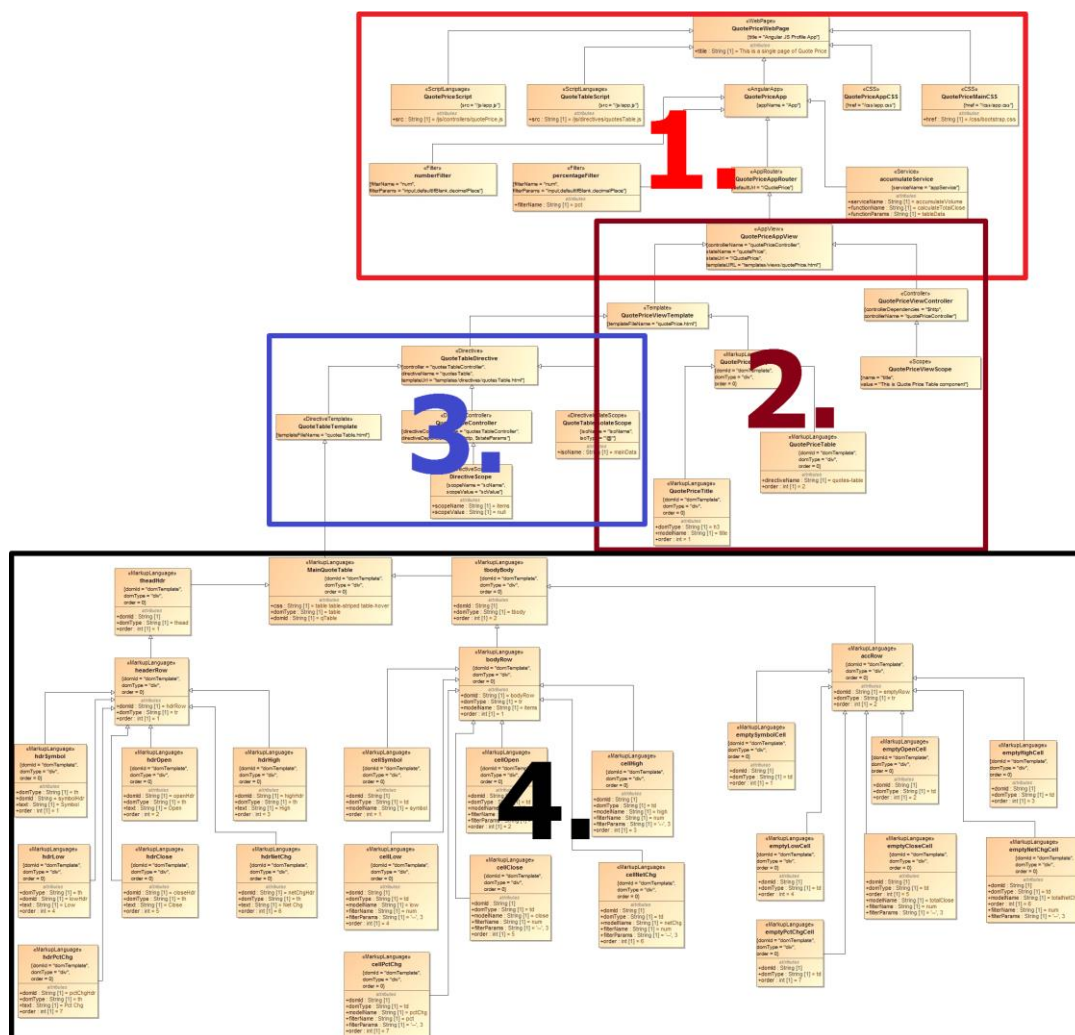
4) ส่วนโครงสร้างของแองกูลาร์เจเอส Directive ซึ่งคือตำแหน่ง 3. ดังภาพที่ 6.4 จะเป็นการประกาศส่วนของแองกูลาร์เจเอส Directive, DirectiveController และ DirectiveTemplate ซึ่งจะเป็นโครงสร้างหลักของแองกูลาร์เจเอส Directive ในรูปแบบของ HTML เพื่อไว้เป็นส่วนแสดงผลของแองกูลาร์เจเอส Directive ดังแสดงในภาพที่ 6.7

5) ส่วนโครงสร้างของ MarkupLanguage ซึ่งคือโครงสร้างของ HTML ที่ประกอบเข้าด้วยกันเพื่อเป็นส่วนแสดงผลของ DirectiveTemplate ซึ่งคือตำแหน่ง 4. ดังภาพที่ 6.4 แต่ในโครงสร้างของ MarkupLanguage นี้สามารถแบ่งออกได้เป็น 3 ส่วน ดังแสดงในภาพที่ 6.8 คือ Header ของ HTML Table ซึ่งคือตำแหน่ง 1. ในภาพที่ 6.8 จะเป็นการแสดงแถวของ Header ของ Table ดังแสดงในภาพที่ 6.9, Body ของ HTML Table ซึ่งคือตำแหน่ง 2. ในภาพที่ 6.8 ซึ่งคือโครงสร้างของ Table สำหรับแสดงข้อมูล ดังแสดงในภาพที่ 6.10 และส่วนของแถวสุดท้ายของ Table

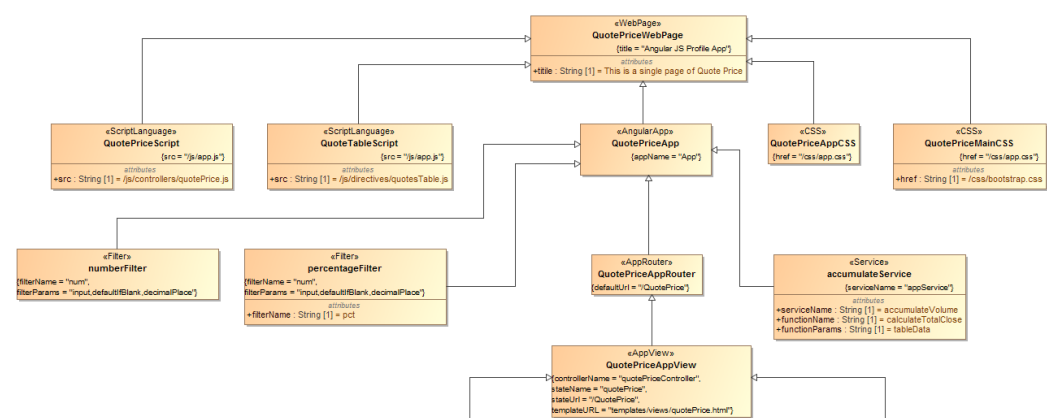
ซึ่งคือตำแหน่ง 3. ในภาพที่ 6.8 ซึ่งมีไว้แสดงผลราคาที่เปลี่ยนแปลงของมูลค่าหุ้นทั้งหมด ดังแสดงในภาพที่ 6.11



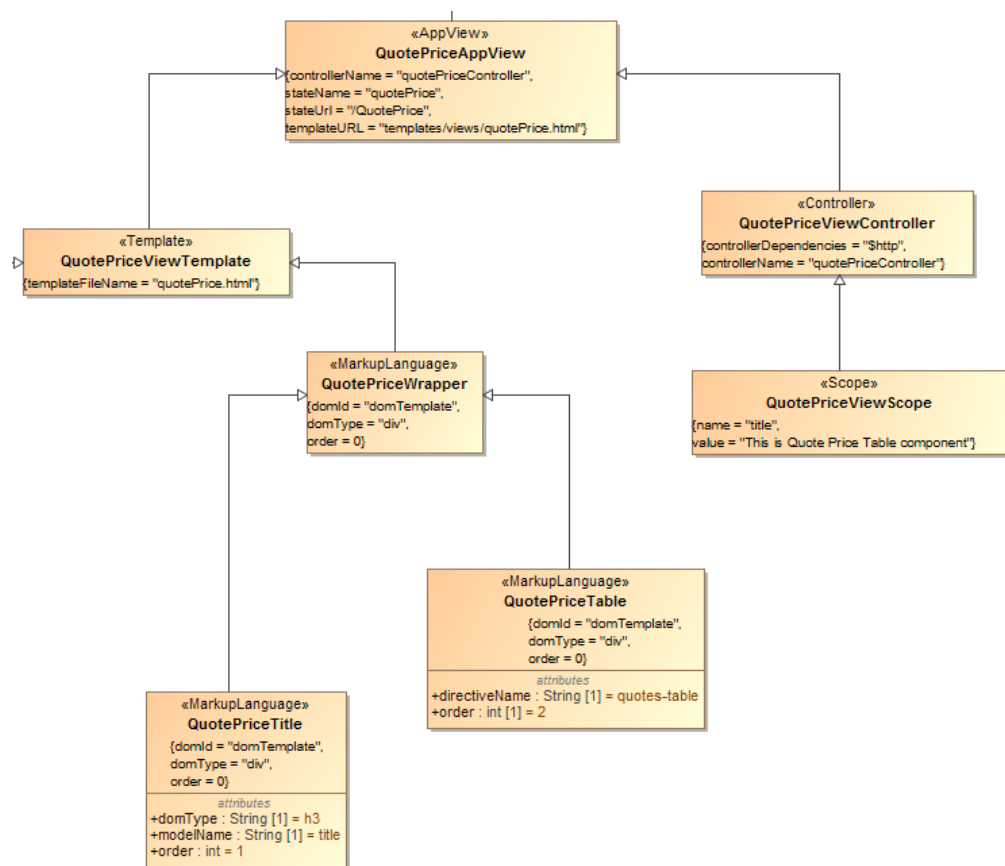
ภาพที่ 6.3 ยูเอ็มแอลโปรไฟล์ของแองกูลาร์เจสจากโปรแกรม MagicDraw



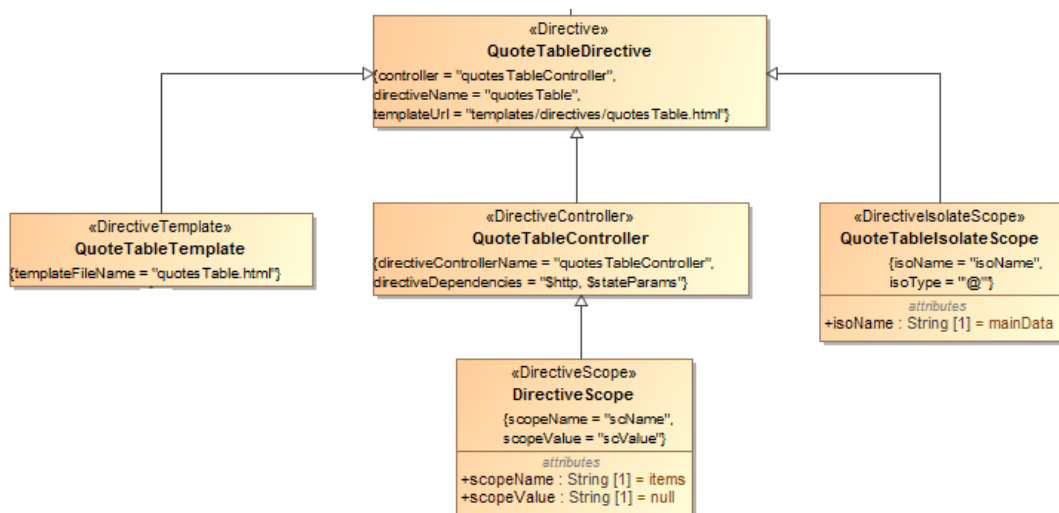
ภาพที่ 6.4 แบบจำลองแผนภาพคลาสของกรณีศึกษาจากโปรแกรม MagicDraw



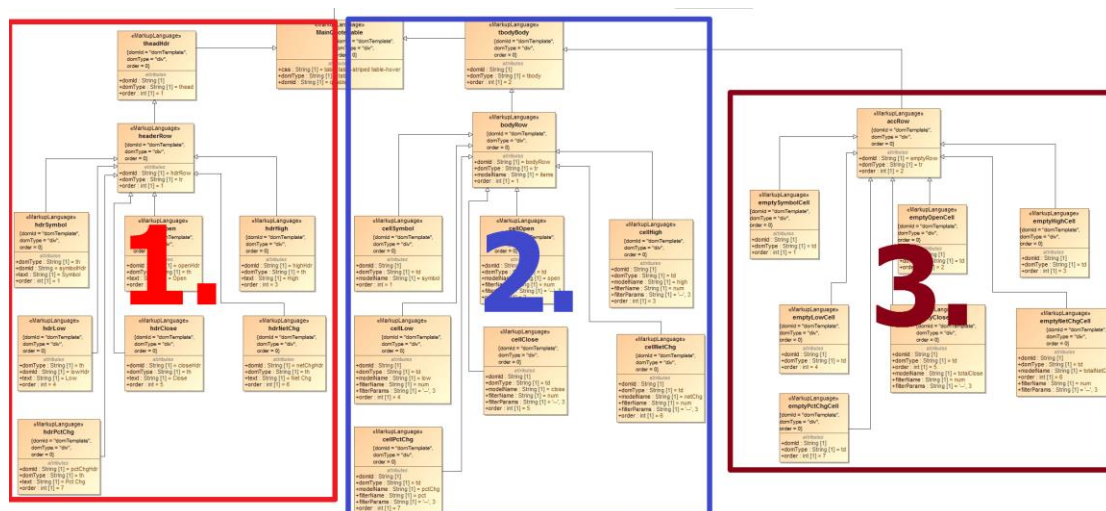
ภาพที่ 6.5 แบบจำลองแผนภาพคลาสส่วนโครงสร้างหลักของแองกการ์เจเอสเว็บแอปพลิเคชัน



ภาพที่ 6.6 แบบจำลองแผนภาพคลาสส่วนโครงสร้างของแองกูลาร์เจเอส View

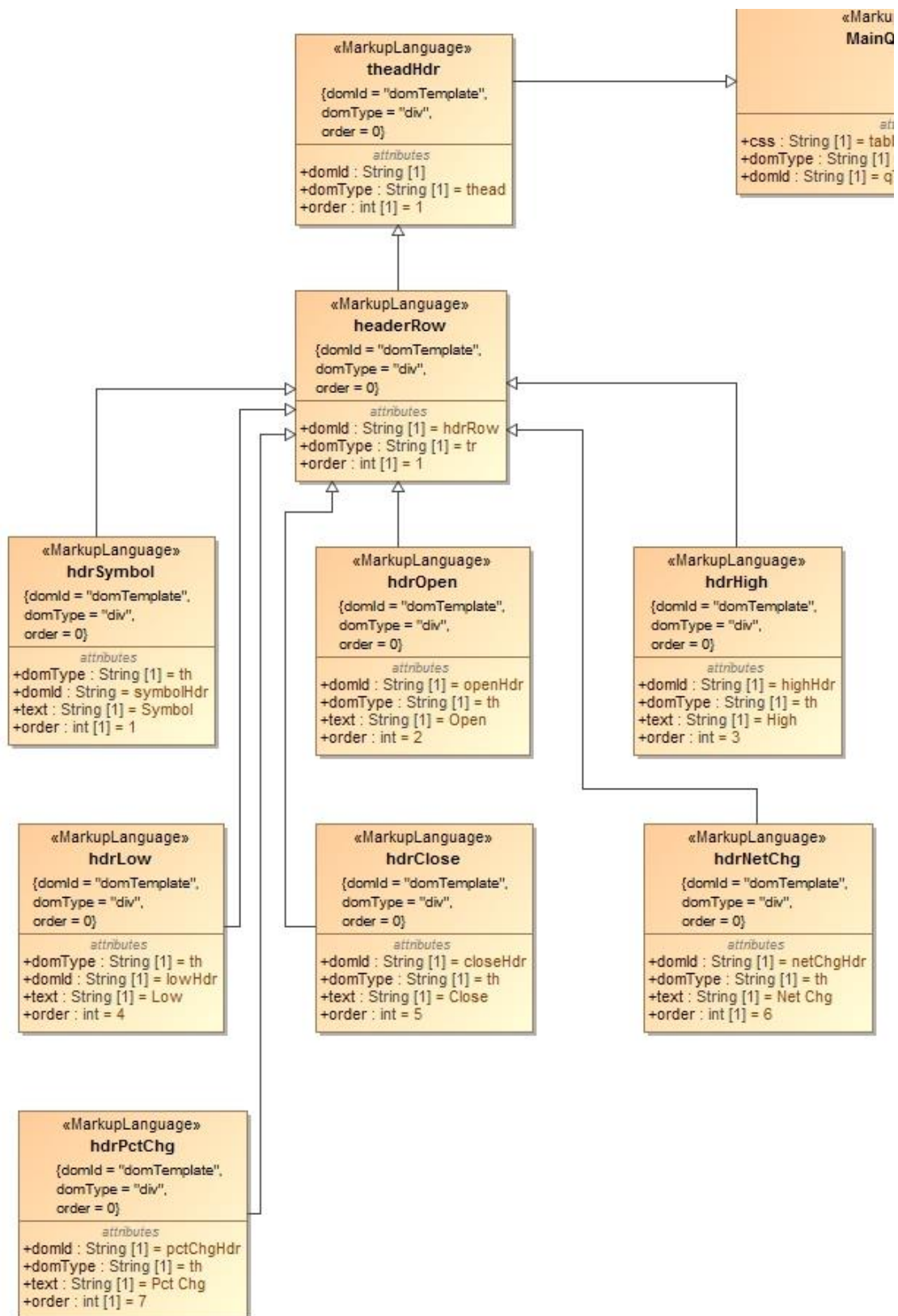


ภาพที่ 6.7 แบบจำลองแผนภาพคลาสส่วนโครงสร้างของแองกูลาร์เจเอส Directive

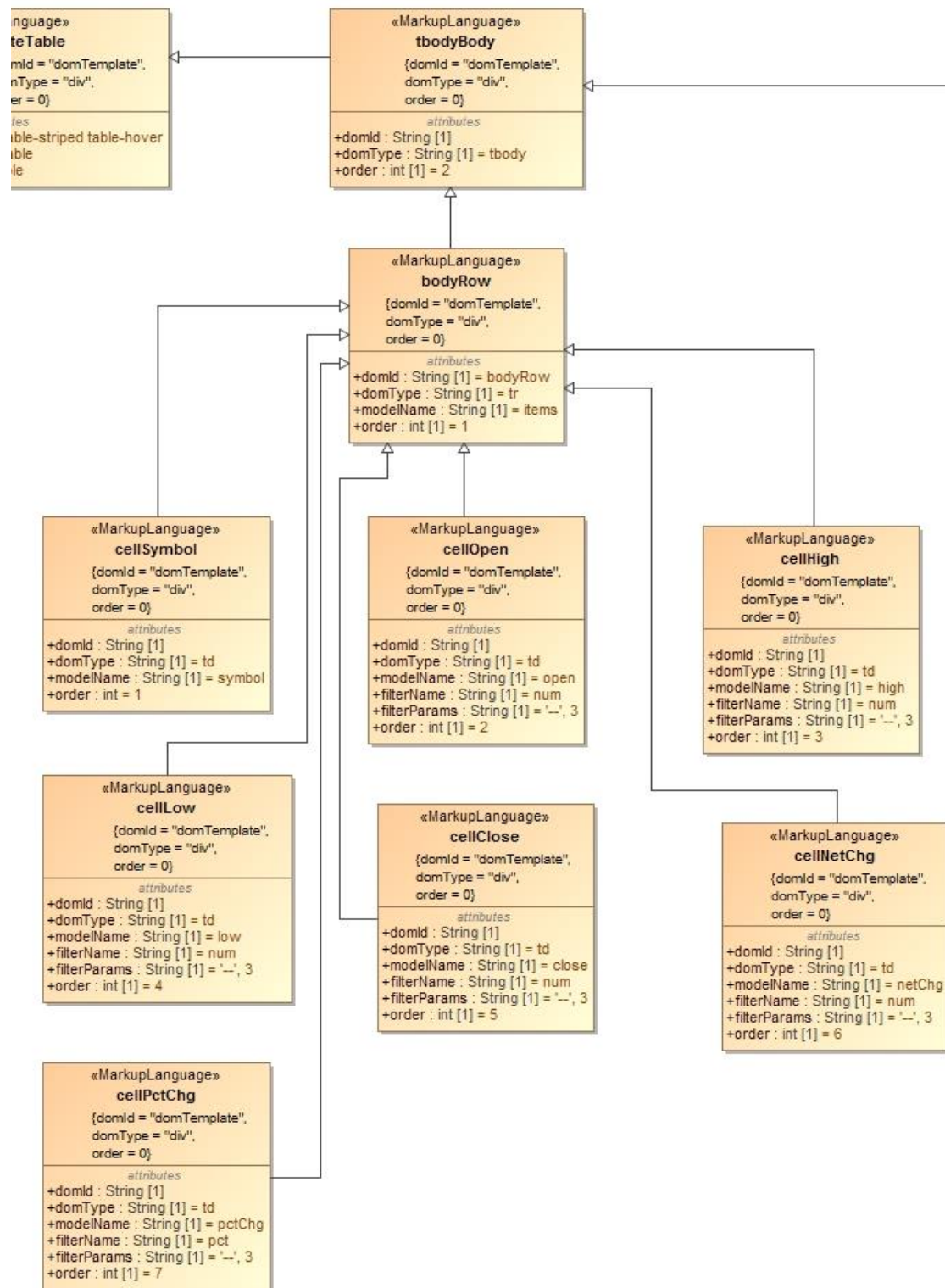


ภาพที่ 6.8 แบบจำลองแผนภาพคลาสส่วนโครงสร้างของ MarkupLanguage

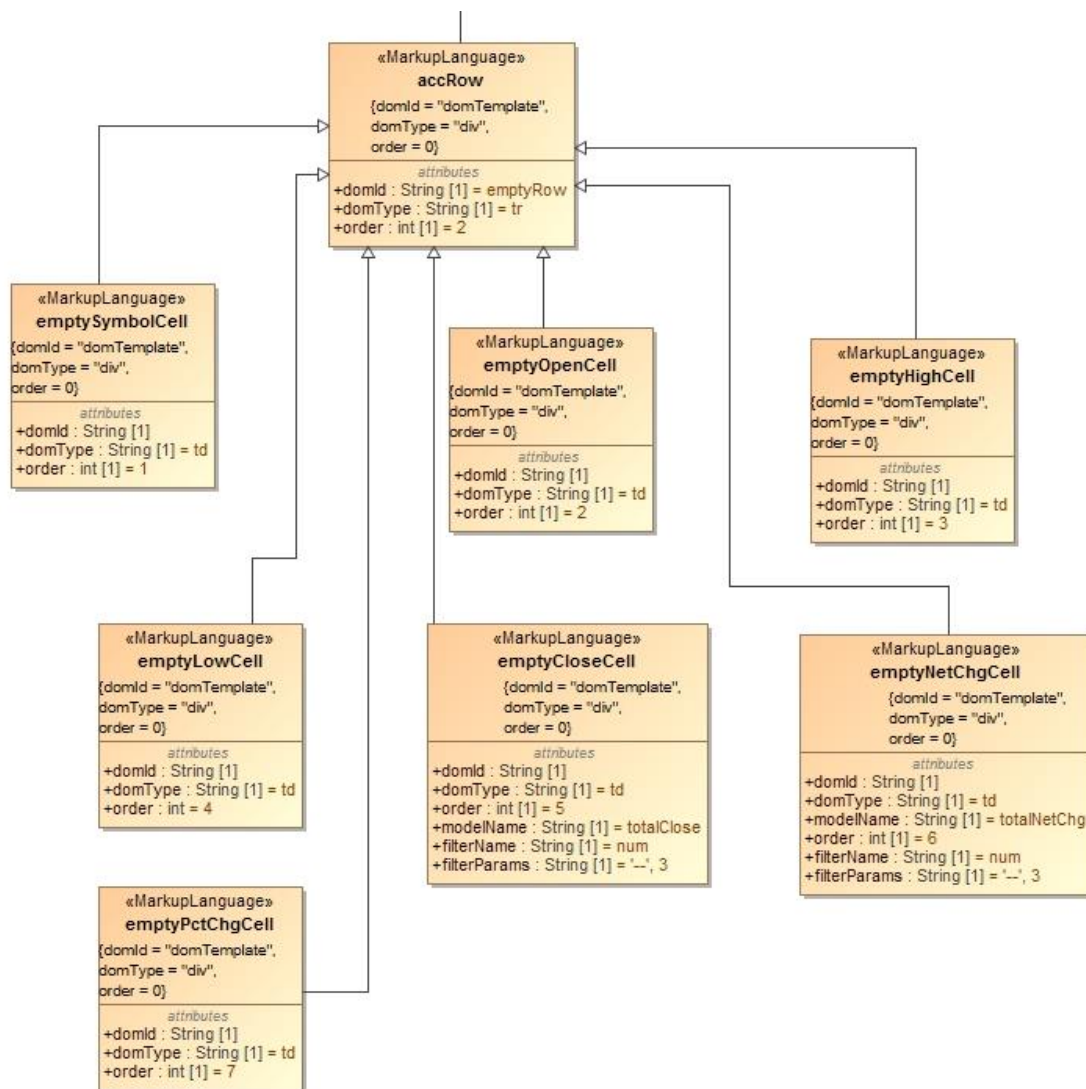




ภาพที่ 6.9 แบบจำลองแผนภาพคลาสส่วน Header ของ Table



ภาพที่ 6.10 แบบจำลองแผนภาพคลาสส่วน Body ของ Table



ภาพที่ 6.11 แบบจำลองแผนภาพคลาสส่วนแถวสุดท้ายของ Table

เมื่อได้ทำการสร้างแบบจำลองแผนภาพคลาสของระบบข้อมูลพื้นฐานของหุ้นออนไลน์ (View Stock Fundamental Online System) โดยใช้ยูเอ็มแอลโปรไฟล์แองกูลาร์เจสสำเร็จออกมาแล้ว จึงให้เครื่องมือทำการแปลงออกมาเป็นโค้ดเทมเพลต หลังจากนั้นจึงนำโค้ดเทมเพลตไปให้ทางผู้พัฒนา (Developer) ทำการเติมโค้ดเพื่อให้ได้ระบบข้อมูลพื้นฐานของหุ้นออนไลน์ที่สมบูรณ์แล้วจึงทำการประเมินผล

## 6.2 การประเมินผล

การประเมินผลการทำงานของเครื่องมือเป็นการประเมินประสิทธิภาพของการแปลงแบบจำลองไปเป็นโค้ด แต่เนื่องจากโค้ดที่เครื่องมือแปลงได้เป็นโค้ดเทมเพลตเท่านั้น เว็บแอปพลิเคชันจึงยังไม่สามารถทำงานได้อย่างสมบูรณ์ ดังนั้นนักพัฒนาจึงยังต้องเติมโค้ดเพื่อให้เว็บแอปพลิเคชันทำงานได้

ตามฟังก์ชันที่ต้องการ ผู้จัดทำโครงการได้เลือกที่จะนำการวัดสัดส่วนการแปลงโค้ดได้เมื่อเทียบกับโค้ดที่ทำงานได้ ซึ่งถูกนำเสนอในงานวิจัย [13] มาใช้ โดยพิจารณาจากโค้ดที่แปลงได้โดยนับจำนวนบรรทัด (Line of Code: LOC) ของโค้ดที่สร้างได้เทียบกับปริมาณโค้ดในแอปพลิเคชันที่นักพัฒนาทำการเพิ่มเข้าไปจนเว็บแอปพลิเคชันสามารถทำงานได้ และได้ทำการประเมินโดยใช้นักพัฒนากลุ่มตัวอย่างเป็นคนทำการทดสอบ โดยมีการแบ่งกลุ่มของนักพัฒนาเป็น 2 กลุ่มคือ กลุ่มที่มีประสบการณ์ในการเขียนเว็บแอปพลิเคชันด้วยเฟรมเวิร์กแองกูลาร์เจสมากกว่า 3 ปีขึ้นไป กับอีกกลุ่มที่มีประสบการณ์ในการเขียนเว็บแอปพลิเคชันด้วยเฟรมเวิร์กแองกูลาร์เจสจำนวนน้อยกว่า 3 ปี แล้วหาค่าเฉลี่ยของจำนวนบรรทัดของโค้ดที่แต่ละกลุ่มต้องเติมเข้าไปยังโค้ดเทมเพลต เพื่อให้ระบบข้อมูลพื้นฐานของหุ่นออนไลน์ใช้งานได้ ซึ่งผลการประเมินเป็นไปตามตารางที่ 6.1

$$\text{สัดส่วนการแปลง} = \text{LOC ของโค้ดเทมเพลต} / \text{LOC ของโค้ดที่สามารถทำงานได้} \times 100 \quad (1)$$

ตารางที่ 6.1 ผลการประเมินการแปลงแบบจำลองเป็นโค้ด

	LOC ของโค้ดที่เติมเข้ามา	AVG ของ LOC ทั้งหมด	สัดส่วนในการเติมโค้ดโดยนักพัฒนาเอง
นักพัฒนามีประสบการณ์มากกว่า 3 ปี (3 คน)	11	105	10%
นักพัฒนามีประสบการณ์น้อยกว่า 3 ปี (3 คน)	18	113	16%
			เฉลี่ย 13%

จากผลการประเมินในตารางที่ 6.1 พบว่าสัดส่วนการแปลง (Transformation Ratio) ซึ่งใช้สูตรจาก (1) ของนักพัฒนาที่มีประสบการณ์น้อยกว่า 3 ปีและมากกว่า 3 ปีมีอัตราการเติมโค้ดเฉลี่ยของนักพัฒนากลุ่มนี้อยู่ที่ 13% ดังนั้นจากกรณีศึกษานี้ปริมาณโค้ดที่ทางเครื่องมือของผู้วิจัยได้พัฒนาออกมานั้นจึงมีส่วนในการสร้างโค้ดที่ครอบคลุมเฉลี่ยสูงถึง 87%

## บทที่ 7

### บทสรุปโครงการและข้อเสนอแนะ

#### 7.1 สรุปผลโครงการมหาบัณฑิต

โครงการมหาบัณฑิตนี้ได้นำเสนอยูเอ็มแอลโปรไฟล์สำหรับเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอส กฎการแปลงสำหรับแปลงแบบจำลองซึ่งสร้างจากแม่พิมพ์ต้นแบบในโปรไฟล์แองกูลาร์เจเอสไปเป็นโค้ด และเครื่องมือในการแปลงแบบจำลองไปเป็นโค้ดเทมเพลต เพื่อให้สามารถนำกฎการแปลงไปประยุกต์ใช้สำหรับแปลงแผนภาพคลาสไปเป็นโค้ด ซึ่งโครงการฉบับนี้ได้ทำการออกแบบกฎสำหรับแปลงแม่พิมพ์ต้นแบบสำหรับสถาปัตยกรรมโคเลเอนด์จำนวน 15 ข้อ จากนั้นได้สร้างแบบจำลองแผนภาพคลาสของระบบกรณีศึกษา ระบบข้อมูลพื้นฐานของหุ้นออนไลน์ (View Stock Fundamental Online System) ซึ่งแบบจำลองจะถูกสร้างจากแม่พิมพ์ต้นแบบในโปรไฟล์แองกูลาร์เจเอส เพื่อทำการทดสอบและพิสูจน์ว่ากฎที่ได้ออกแบบไว้สามารถนำมาประยุกต์ใช้ได้จริง แล้วจึงทำการแปลงแบบจำลองที่ออกแบบไว้ผ่านทางเครื่องมือที่โครงการมหาบัณฑิตได้สร้างขึ้นมาเพื่อแปลงเป็นโค้ดเทมเพลต หลังจากได้โค้ดเทมเพลตออกมานักพัฒนาต้องมีการเติมโค้ดเพื่อให้เว็บแอปพลิเคชันระบบข้อมูลพื้นฐานของหุ้นออนไลน์สามารถใช้งานได้จริง หลังจากได้โค้ดที่สมบูรณ์แล้วจึงทำการประเมินผลเพื่อหาสัดส่วนในการแปลง ซึ่งผลการประเมินพบว่าสัดส่วนในการแปลงแบบจำลองเป็นโค้ดของเครื่องมือและกฎการแปลงที่ได้พัฒนาออกมานั้นมีสัดส่วนการแปลงเฉลี่ยอยู่ที่ 87% ของโค้ดที่สมบูรณ์ ดังนั้นประโยชน์ที่จะได้รับคือนักพัฒนาเว็บแอปพลิเคชันบนเฟรมเวิร์กแองกูลาร์เจเอสจะสามารถลดระยะเวลาในการพัฒนาได้ เนื่องจากสัดส่วนในการแปลงเฉลี่ยที่ได้ค่อนข้างสูง รวมถึงวิธีที่เสนอยังทำให้ได้แบบจำลองการออกแบบเว็บแอปพลิเคชันซึ่งเป็นเอกสารที่จะเป็นประโยชน์ในการบำรุงรักษาเว็บแอปพลิเคชันต่อไป

#### 7.2 ข้อจำกัดในการทำโครงการ

โครงการมหาบัณฑิตนี้ยังมีข้อจำกัดดังต่อไปนี้

- 1) โค้ดที่ได้จากการแปลงออกมาจากเครื่องมือที่พัฒนาขึ้นนั้น ยังเป็นเพียงโค้ดเทมเพลต ซึ่งยังไม่สามารถนำไปใช้ได้จริง ต้องมีการเพิ่มเติมโค้ดที่จำเป็นทางด้าน Business logic เพิ่มเติมเข้าไปเพื่อให้โค้ดเหล่านั้นทำงานได้จริง
- 2) การทดสอบและประเมินประสิทธิภาพของเครื่องมือยังทำโดยใช้ระบบกรณีศึกษาเพียงระบบเดียว จึงอาจยังไม่สะท้อนประสิทธิภาพที่แท้จริงของเครื่องมือ
- 3) ค่าสัดส่วนการแปลงโค้ดที่ใช้ในการประเมินประสิทธิภาพของเครื่องมือ ยังขึ้นอยู่กับนักพัฒนาที่ต้องเพิ่มโค้ดให้ทำงานได้ รวมถึงความซับซ้อนของ Business logic เพราะนักพัฒนาแต่ละรายอาจเขียนโค้ดที่ทำงานได้โดยมีจำนวนบรรทัดหรือคุณภาพที่แตกต่างกัน รวมถึงในแง่ของความแตกต่างทางด้าน Business logic ที่แตกต่างกันในแต่ละความต้องการ จึงอาจกระทบกับค่าสัดส่วนที่ได้

4) การออกแบบยูเอ็มแอลโปรไฟล์แองกูลาร์เจเอส ในส่วนของการสร้างแบบจำลอง Template และ DirectiveTemplate นั้นยังไม่ครอบคลุมทุก HTML Element แต่สามารถต่อยอดเพิ่มเติมได้ในภายหลังเพื่อให้มีความครอบคลุมมากที่สุด

### 7.3 ข้อเสนอแนะ

ข้อเสนอแนะในการพัฒนาเพิ่มเติมมีดังนี้

1) เนื่องจากกรณีศึกษาของงานวิจัยชิ้นนี้ คือ เว็บแอปพลิเคชันระบบข้อมูลพื้นฐานของหุ้นออนไลน์ (View Stock Fundamental Online System) ดังนั้นถ้ามีการนำยูเอ็มแอลโปรไฟล์ของทางด้านการเงินหรือการลงทุนเข้ามาควบรวมในยูเอ็มแอลโปรไฟล์แองกูลาร์เจเอส ก็จะทำให้ยูเอ็มแอลโปรไฟล์แองกูลาร์เจเอส นั้นมีความน่าสนใจในการนำไปใช้งานมากยิ่งขึ้น และมีความเฉพาะเจาะจงกับโดเมนเฉพาะด้านมากยิ่งขึ้น

2) ปรับปรุงการประเมินผลประสิทธิภาพของเครื่องมือให้แม่นยำยิ่งขึ้นโดยเพิ่มกรณีศึกษาหรือใช้โค้ดของเว็บแอปพลิเคชันที่มีอยู่แล้วเป็นตัวเปรียบเทียบกับโค้ดที่แปลงได้จากเครื่องมือในโครงการว่าสามารถแปลงได้เป็นสัดส่วนเท่าใด

## รายการอ้างอิง

- [1] OMG. [Online]. Available: <http://www.omg.org/>. Last Accessed: 15 Nov 2015.
- [2] OMG. MDA - The Architecture Of Choice For A Changing World [Online]. Available: <http://www.omg.org/mda/>. Last Accessed: 15 Nov 2015.
- [3] UML. Documents Associated With Unified Modeling Language (UML) Version 2.5 [Online]. Available: <http://www.omg.org/spec/UML/2.5/>. Last Accessed: 15 Nov 2015.
- [4] Michael, Mikowski, and Josh Powell. "Single page web applications: JavaScript end-to-end". Manning Publications Co., 2013.
- [5] AngularJS. [Online]. Available: <https://builtwith.angularjs.org/>. Last Accessed: 15 Nov 2015.
- [6] MDA Processes. [Online]. Available: <http://www.jtripathi.com/Downhome/Architecture/modeldrivenarchitecturemda/>. Last Accessed: 15 Nov 2015.
- [7] UML Profile. [Online]. Available: <http://www.omg.org/mda/specs.htm>. Last Accessed: 15 Nov 2015.
- [8] Siti Azreena, Mubin, and Azrul Hazri Jantan. "A UML 2.0 profile web design framework for modeling complex web application", 2014 International Conference on Information Technology and Multimedia (ICIMU), IEEE, 2014, pp. 324-329.
- [9] OCL. Object Constraint Language (OCL) [Online]. Available: <http://www.omg.org/spec/OCL/>. Last Accessed: 15 Nov 2015.
- [10] Nilesh, Jain, Priyanka Mangal, and Deepak Mehta. "AngularJS: A Modern MVC Framework in JavaScript", Journal of Global Research in Computer Science 5.12 (2015): 17-23.
- [11] AngularJS. [Online]. Available: <https://angularjs.org/>. Last Accessed: 15 Nov 2015.
- [12] Somrudee Kaewkao and Twittie Senivongse, "A Model-Driven Development of Web-Based Applications on Google App Engine Platform", Proceedings of 10th National Conference on Computing and Information Technology (NCCIT 2014), Bangkok, Thailand, 8-9 May 2014, pp. 140-145 (in Thai).
- [13] Yen-Chieh Huang, Chih-Ping Chu, Zhu-An Lin and Michael Matuschek, "Transformation from Web PSM to Code", in International Conference on Distributed Multimedia Systems (DMS), 2009.

- [14] Mukesh, Kataria, Raj Yadav, and Ajay Khunteta, "A component-centric UML based approach for modeling the architecture of web applications.", *Int. J. Recent Research and Review* 5 (2013): 22-27.
- [15] I-Ching, Hsu, "Visual modeling for Web 2.0 applications using model driven architecture approach", *Simulation Modelling Practice and Theory* 31 (2013): 63-76.
- [16] Wen-Hao David, Huang, Denice Ward Hood, and Sun Joo Yoo. "Gender divide and acceptance of collaborative Web 2.0 applications for learning in higher education", *The Internet and Higher Education* 16 (2013): 57-65.
- [17] XML. Documents Associated With XML Metadata Interchange (XMI), Version 2.4.2 [Online]. Available: <http://www.omg.org/spec/XMI/2.4.2/>. Last Accessed: 15 Nov
- [18] 2015.  
MagicDraw. The multi award-winning UML business process, architecture, software and system modeling tool with teamwork support, Version 18.2 [Online]. Available:
- [19] <http://www.nomagic.com/products/magicdraw.html>. Last Accessed: 7 Jun 2016.  
NPM. Node Package Manager [Online]. Available: <https://www.npmjs.com/>. Last Accessed: 7 Jun 2016.



ภาคผนวก

## ภาคผนวก ก

### แผนภาพและตารางคำอธิบายยูสเคส

แผนภาพยูสเคสนั้นแสดงให้เห็นถึงโครงสร้างของระบบ ระบบการทำงานย่อย และฟังก์ชันการทำงานส่วนต่าง ๆ ของระบบ รวมถึงแสดงให้เห็นถึงความสัมพันธ์ระหว่างผู้ใช้กับระบบอีกด้วย ซึ่งสามารถแสดงดังต่อไปนี้

#### ก.1 คำอธิบายยูสเคส (Use Case description)

ตารางที่ ก. 1 คำอธิบายยูสเคสการนำเข้าแบบจำลองยูเอ็มแอลโปรไฟล์ในรูปแบบไฟล์ XMI (Locate XMI)

ชื่อยูสเคส :	Locate XMI	รหัส : UC01	ระดับความสำคัญ : สูง
ผู้กระทำหลัก :	ผู้ใช้งานระบบ (Users)	ชนิดยูสเคส :	ภาพรวม
<b>ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :</b> ผู้ใช้งานระบบ ต้องการนำเข้าแบบจำลองในรูปแบบของไฟล์ XMI เครื่องมือ M2C Transcoder รับค่าข้อมูลจากผู้ใช้งานระบบ เพื่อใช้เป็นข้อมูลนำเข้าสำหรับการแปลงแบบจำลอง			
<b>รายละเอียดยูสเคส :</b> แผนภาพยูสเคสนี้อธิบายขั้นตอนการนำเข้าแบบจำลองยูเอ็มแอลโปรไฟล์ในรูปแบบไฟล์ XMI			
<b>สิ่งกระตุ้น :</b> ผู้ใช้งานระบบต้องการให้ระบบทำงาน			
<b>ความสัมพันธ์ :</b> ความเกี่ยวเนื่อง : การรวม : การขยาย : การสืบทอด :			
<b>สายงานปกติ :</b> <ol style="list-style-type: none"> <li>1. ผู้ใช้งานระบบคลิกปุ่ม Browse</li> <li>2. ผู้ใช้งานระบบเลือกไฟล์แบบจำลองในรูปแบบไฟล์ XMI ที่ต้องการ</li> <li>3. ระบบทำการแสดงค่าเส้นทางของไฟล์ที่เลือก</li> </ol>			
<b>สายงานย่อย :</b>			
<b>สายงานทางเลือก / สายงานพิเศษ :</b>			

ตารางที่ ก. 2 คำอธิบายยูสเคสการเลือกโฟลเดอร์สำหรับจัดเก็บโค้ด (Locate Source Code Folder)

ชื่อยูสเคส :	Locate Source Code Folder	รหัส : UC02	ระดับความสำคัญ : สูง
ผู้กระทำหลัก :	ผู้ใช้งานระบบ (Users)	ชนิดยูสเคส :	ภาพรวม
<b>ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :</b> ผู้ใช้งานระบบ ต้องการเลือกโฟลเดอร์สำหรับจัดเก็บโค้ดที่ได้หลังจากมีการแปลงแบบจำลองเครื่องมือ M2C Transcoder รับค่าข้อมูลจากผู้ใช้งานระบบ เพื่อใช้เป็นข้อมูลนำเข้าสำหรับระบบตำแหน่งของผลลัพธ์จากการแปลงแบบจำลอง			
<b>รายละเอียดยูสเคส :</b> แผนภาพยูสเคสนี้อธิบายขั้นตอนการเลือกโฟลเดอร์สำหรับจัดเก็บโค้ดหลังจากแปลงแบบจำลองเสร็จ			
<b>สิ่งกระตุ้น :</b> ผู้ใช้งานระบบความต้องการให้ระบบทำงาน			
<b>ความสัมพันธ์ :</b> ความเกี่ยวเนื่อง : การรวม : การขยาย : การสืบทอด :			
<b>สายงานปกติ :</b> <ol style="list-style-type: none"> <li>1. ผู้ใช้งานระบบคลิกปุ่ม Browse</li> <li>2. ผู้ใช้งานระบบเลือกโฟลเดอร์สำหรับจัดเก็บโค้ดที่ต้องการ</li> <li>3. ระบบทำการแสดงค่าเส้นทางของโฟลเดอร์ที่เลือก</li> </ol>			
<b>สายงานย่อย :</b>			
<b>สายงานทางเลือก / สายงานพิเศษ :</b>			

ตารางที่ ก. 3 คำอธิบายยูสเคสการกำหนดไลบรารีของแองกูลาร์เจเอส (Locate AngularJS Library)

ชื่อยูสเคส :	Locate AngularJS Library	รหัส : UC03	ระดับความสำคัญ : สูง
ผู้กระทำหลัก :	ผู้ใช้งานระบบ (Users)	ชนิดยูสเคส :	ภาพรวม
<b>ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :</b> ผู้ใช้งานระบบ ต้องการเลือกโพลเดอร์สำหรับกำหนดไลบรารีของ AngularJS เครื่องมือ M2C Transcoder รับค่าตำแหน่งของ AngularJS Library เพื่อนำไปกำหนดให้ผลลัพธ์ที่ได้จากการแปลงแบบจำลองเรียกใช้ไลบรารีชุดนี้ในการทำงาน			
<b>รายละเอียดยูสเคส :</b> แผนภาพยูสเคสนี้อธิบายขั้นตอนการเลือกโพลเดอร์สำหรับระบุตำแหน่งของ AngularJS Library			
<b>สิ่งกระตุ้น :</b> ผู้ใช้งานระบุความต้องการให้ระบบทำงาน			
<b>ความสัมพันธ์ :</b> ความเกี่ยวเนื่อง : การรวม : การขยาย : การสืบทอด :			
<b>สายงานปกติ :</b> <ol style="list-style-type: none"> <li>1. ผู้ใช้งานระบบคลิกปุ่ม Browse</li> <li>2. ผู้ใช้งานระบบเลือกโพลเดอร์สำหรับระบุตำแหน่งของ AngularJS Library</li> <li>3. ระบบทำการแสดงค่าเส้นทางของโพลเดอร์ที่เลือก</li> </ol>			
<b>สายงานย่อย :</b>			
<b>สายงานทางเลือก / สายงานพิเศษ :</b>			

ตารางที่ ก. 4 คำอธิบายยูสเคสการแปลงแบบจำลองเป็นโค้ด (Generate Code)

ชื่อยูสเคส :	Generate Code	รหัส : UC04	ระดับความสำคัญ : สูง
ผู้กระทำหลัก :	ผู้ใช้งานระบบ (Users)	ชนิดยูสเคส :	ภาพรวม
<b>ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :</b> ผู้ใช้งานระบบ ต้องการแปลงแบบจำลองเป็นโค้ด			
<b>รายละเอียดยูสเคส :</b> แผนภาพยูสเคสนี้อธิบายขั้นตอนการแปลงแบบจำลองเป็นโค้ด			
<b>สิ่งกระตุ้น :</b> ผู้ใช้งานระบุความต้องการให้ระบบทำงาน			
<b>ความสัมพันธ์ :</b> <b>ความเกี่ยวเนื่อง :</b> <b>การรวม :</b> Locate XML, Locate Source Code Folder, Locate AngularJS Library <b>การขยาย :</b> View Generated Source Code, Open Generated Source Code Folder <b>การสืบทอด :</b>			
<b>สายงานปกติ :</b> 1. ผู้ใช้งานระบบคลิกปุ่ม Process เพื่อแปลงแบบจำลองเป็นโค้ด 2. ระบบทำการแปลงแบบจำลองเป็นโค้ด			
<b>สายงานย่อย :</b>			
<b>สายงานทางเลือก / สายงานพิเศษ :</b> 1a. ระบบไม่ทำการประมวลผลการแปลง หากค่าเส้นทางของไฟล์และโฟลเดอร์มีค่าว่าง			

ตารางที่ ก. 5 คำอธิบายยูสเคสการดูโค้ดหลังจากแปลงแบบจำลองเสร็จ (View Generated Source Code)

ชื่อยูสเคส :	View Generated Source Code	รหัส : UC05	ระดับความสำคัญ : ปานกลาง
ผู้กระทำหลัก :	ผู้ใช้งานระบบ (Users)	ชนิดยูสเคส :	ภาพรวม
<b>ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :</b> ผู้ใช้งานระบบ ต้องการดูโค้ดหลังจากแปลงแบบจำลองเสร็จ			
<b>รายละเอียดยูสเคส :</b> แผนภาพยูสเคสนี้อธิบายขั้นตอนการแสดงผลโค้ดหลังจากแปลงแบบจำลองเสร็จ			
<b>สิ่งกระตุ้น :</b> ผู้ใช้งานระบบต้องการให้ระบบทำงาน			
<b>ความสัมพันธ์ :</b> ความเกี่ยวเนื่อง : การรวม : การขยาย : การสืบทอด :			
<b>สายงานปกติ :</b> <ol style="list-style-type: none"> <li>1. ผู้ใช้งานระบบคลิกปุ่ม View output source code</li> <li>2. ระบบทำการแสดงโค้ดทั้งหมดที่ได้จากการแปลง รูปแบบแสดงผลเป็นลำดับขั้น เพื่อเลือกดูไฟล์ที่ต้องการได้</li> <li>3. เมื่อผู้ใช้งานคลิกที่ไฟล์ที่ต้องการเปิดจากช่องการแสดงผลทางด้านซ้าย เครื่องมือจะทำการแสดงรายละเอียดของไฟล์ที่ด้านขวา</li> </ol>			
<b>สายงานย่อย :</b>			
<b>สายงานทางเลือก / สายงานพิเศษ :</b>			

ตารางที่ ก. 6 คำอธิบายยูสเคสการดูไฟล์เตอร์ของโค้ดที่ได้จากการแปลงแบบจำลอง (View Generated Source Code Folder)

ชื่อยูสเคส :	View Generated Source Code Folder	รหัส : UC06	ระดับความสำคัญ : ปานกลาง
ผู้กระทำหลัก :	ผู้ใช้งานระบบ (Users)	ชนิดยูสเคส :	ภาพรวม
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : ผู้ใช้งานระบบ ต้องการดูโฟลเตอร์ของผลลัพธ์ที่ได้จากการแปลงแบบจำลอง			
รายละเอียดยูสเคส : แผนภาพยูสเคสนี้ อธิบายขั้นตอนการเปิดดูโฟลเตอร์ของผลลัพธ์ที่ได้จากการแปลงแบบจำลอง			
สิ่งกระตุ้น :	ผู้ใช้งานระบบ ต้องการให้ระบบทำงาน		
ความสัมพันธ์ : ความเกี่ยวเนื่อง : การรวม : การขยาย : การสืบทอด :			
สายงานปกติ : 1. ผู้ใช้งานระบบคลิกปุ่ม View output folder 2. ระบบทำการแสดงข้อมูลโฟลเตอร์ของผลลัพธ์ที่ได้จากการแปลงแบบจำลอง			
สายงานย่อย :			
สายงานทางเลือก / สายงานพิเศษ :			

### ประวัติผู้เขียนโครงงานมหำบัณฑิต

นายวุฒิชัย จันทรสุวัฒน์ เกิดวันที่ 1 พฤษภาคม 2526 สำเร็จการศึกษาระดับปริญญาตรี  
หลักสูตรวิทยาศาสตร์บัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบัน  
เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2548

ประสบการณ์ พนักงานองค์กรเอกชน บริษัท ทอมสัน รอยเตอร์ ซอฟต์แวร์ ประเทศไทย  
จำกัด ตำแหน่งนักพัฒนาซอฟต์แวร์

เข้าศึกษาต่อระดับปริญญาโทบัณฑิต ปีการศึกษา 2558 หลักสูตรวิทยาศาสตรมหาบัณฑิต  
(วท.ม.) สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์  
มหาวิทยาลัย