Zotonic

Zotonic เป็นโอเฟรมเวิร์กเปิด (open-source framework) สำหรับการพัฒนาเว็บแบบฟูลสแตก (full-stack) ตั้งแต่ฟรอนท์เอนด์ (frontend) ไปจนถึงแบ็กเอนด์ (backend) ประกอบด้วยชุดฟังก์ชันหลักขนาดเล็ก ใช้ระบบจัดการเนื้อหาที่มีน้ำหนักเบาแต่ขยายได้ด้านบน เป้าหมายหลักของ Zotonic คือการทำให้ง่ายต่อการสร้างเว็บไซต์ที่มีประสิทธิภาพ " out of the box " เพื่อให้เว็บไซต์มีขนาดที่ดีตั้งแต่เริ่มต้น แม้ว่าจะมีฟีเจอร์และฟังก์ชันการทำงานมากมายร่วมกับเฟรมเวิร์ก (framework) การพัฒนาเว็บ เช่น Django, Drupal, Ruby on Rails และ Wordpress ข้อได้เปรียบในการแข่งขันหลักคือภาษาที่ Zotonic ขับเคลื่อนโดยใช้ Erlang ภาษานี้ซึ่งเดิมพัฒนาขึ้นสำหรับการสร้างสวิตช์โทรศัพท์ ทำให้ Zotonic สามารถทนต่อข้อผิดพลาดและมีลักษณะการทำงานที่ยอดเยี่ยม เช่นเดียวกับชื่อเรื่อง บทนี้เน้นที่ประสิทธิภาพของ Zotonic เราจะดูสาเหตุที่ Erlang ได้รับเลือกให้เป็นแพลตฟอร์ม (platform) การเขียนโปรแกรม จากนั้นตรวจสอบสแต็กคำขอ HTTP จากนั้นเจาะลึกถึงกลยุทธ์การแคชที่ Zotonic ใช้สุดท้าย เราจะอธิบายการเพิ่มประสิทธิภาพที่เรานำไปใช้กับโมดูลย่อยและฐานข้อมูลของ Zotonic

วัตถุประสงค์

Zotonic เป็นชุดที่สมบูรณ์สำหรับการสร้างเว็บไซต์ขั้นสูงที่ใช้ CMS (Content Management System) ได้อย่างรวดเร็ว และสามารถปรับแต่งเว็บเซิร์ฟเวอร์ (Web-server) ตามความยืดหยุนของผู้ใช้งานได้ทันทีโดยไม่จำเป็นต้องติดตั้ง Apache หรือ Nginx แยก

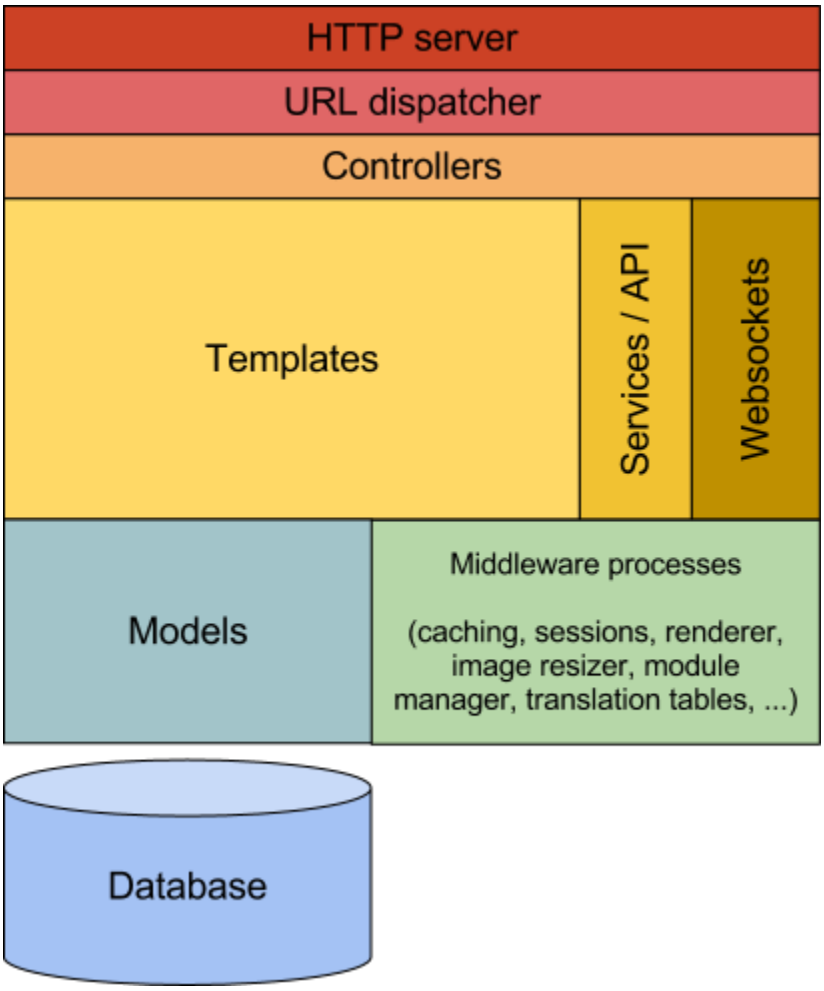
Architectural Patterns / Styles

Zotonic ใช้สถาปัตยกรรมแบบ MVC (model-view-controller) แต่แตกต่างตรงที่ view zotonic ใช้Templates และ มาพร้อมกับเว็บเซิร์ฟเวอร์ในตัว Mochiweb ไม่ต้องใช้เว็บเซิร์ฟเวอร์ภายนอก สิ่งนี้ทำให้การพึ่งพาการปรับใช้ให้น้อยที่สุด URLใช้เพื่อจับคู่คำขอกับตัวควบคุม ตัวควบคุมจะจัดการคำขอแต่ละรายการในลักษณะที่สงบ ต้องขอบคุณไลบรารี Webmachine ตัวควบคุมเป็น “dumb” โดยมีวัตถุประสงค์โดยไม่มีตรรกะเฉพาะแอปพลิเคชันมากนัก Zotonic มีตัวควบคุมมาตรฐานจำนวนหนึ่งซึ่งมักจะดีเพียงพอสำหรับการพัฒนาเว็บแอปพลิเคชันพื้นฐาน ตัวอย่างเช่น มี **controller_template** ซึ่งมีวัตถุประสงค์เพียงเพื่อตอบกลับ คำขอ HTTP GETโดยการแสดงเทมเพลตที่กำหนด

ภาษาเทมเพลตคือการนำ Erlang ไปใช้ในภาษาเทมเพลต Django ที่รู้จักกันดี เรียกว่า ErlyDTL หลักการทั่วไปใน Zotonic คือเทมเพลตจะขับเคลื่อนคำขอข้อมูล เทมเพลตจะกำหนดข้อมูลที่ต้องการ และดึงข้อมูลจากแบบจำลอง

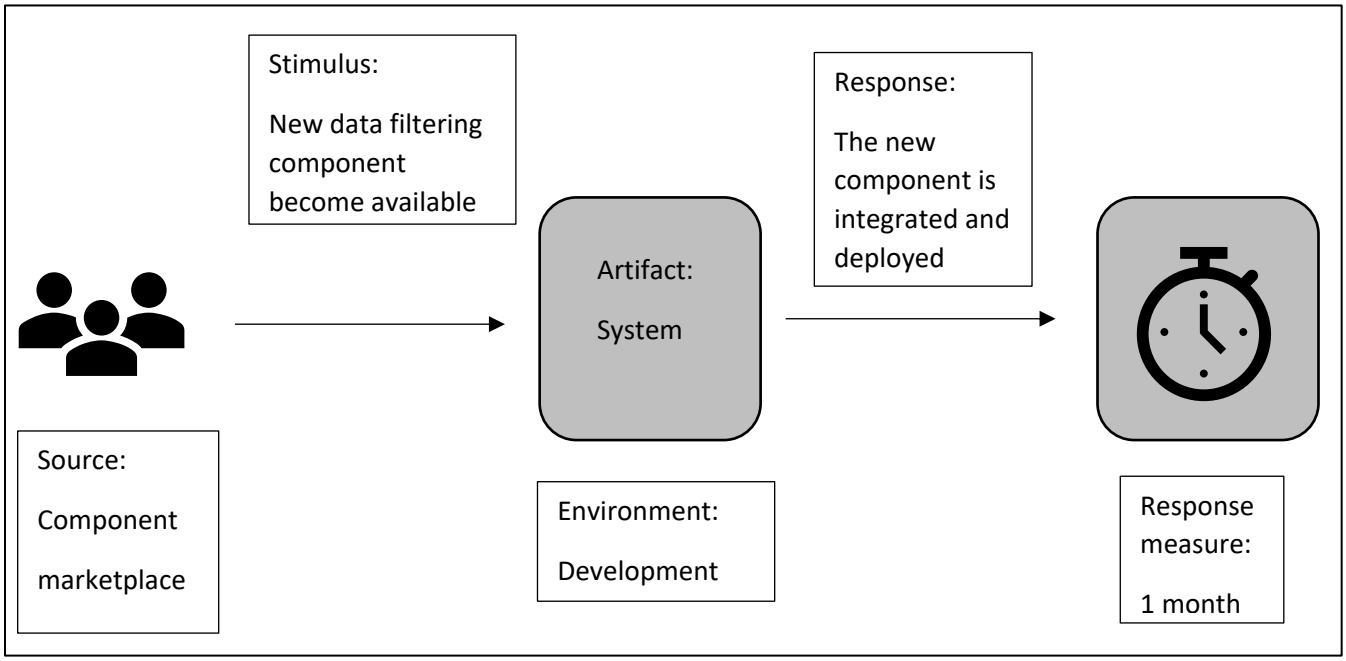
โมเดลแสดงฟังก์ชันเพื่อดึงข้อมูลจากแหล่งข้อมูลต่างๆ เช่น ฐานข้อมูล โมเดลต่างๆ เปิดเผยAPIให้กับเทมเพลต โดยจะกำหนดวิธีใช้งาน โมเดลยังรับผิดชอบในการแคชผลลัพธ์ในหน่วยความจำ พวกเขาตัดสินใจว่าจะแคชเมื่อใดและอะไรและนานแค่ไหน เมื่อเทมเพลตต้องการข้อมูล พวกเขาเรียกโมเดลราวกับว่ามันเป็นตัวแปรที่พร้อมใช้งานทั่วโลก

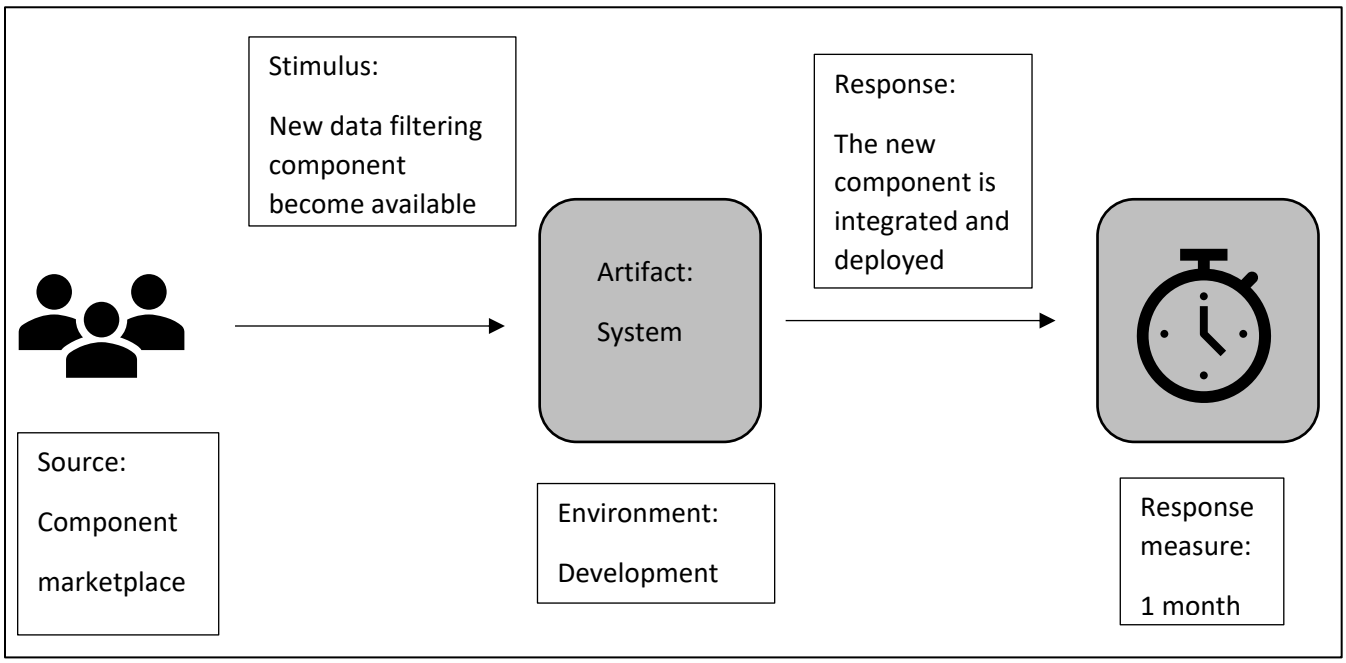
รูปแบบสถาปัตยกรรมของZotonic

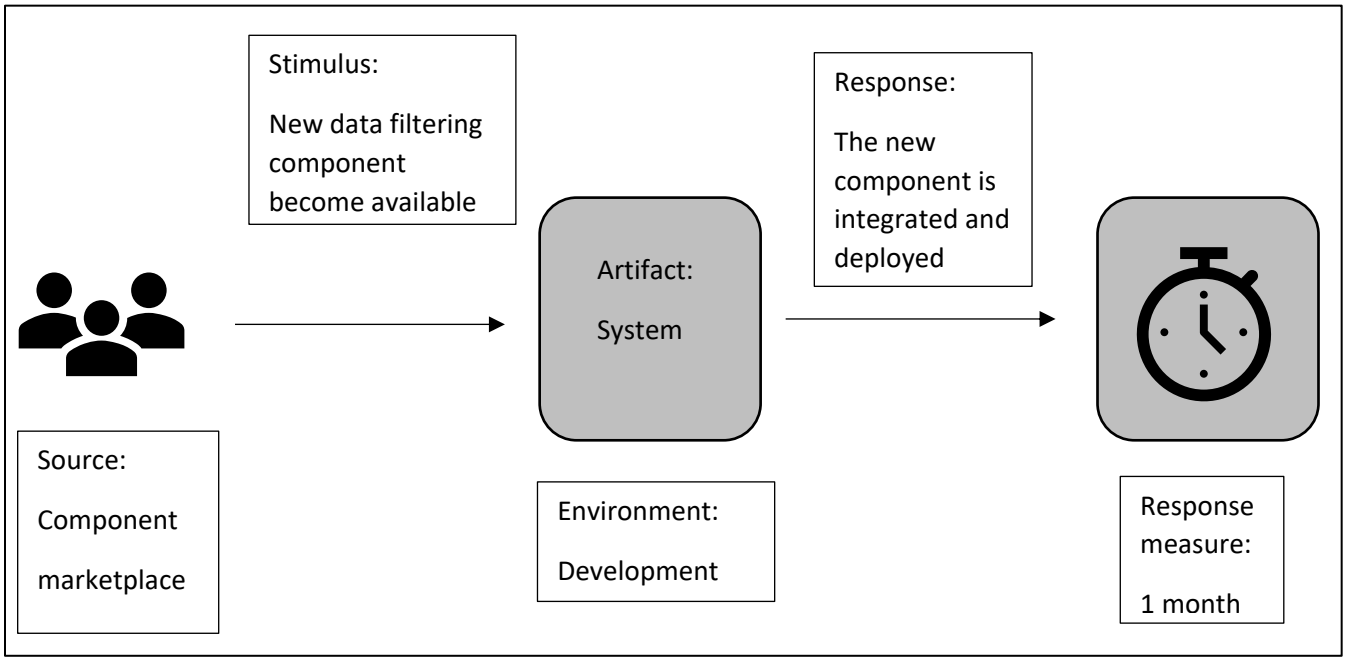


แหล่งที่มาของรูปภาพ: https://www.aosabook.org/en/posa/zotonic.html

Quality Attribute Scenarios







แหลงอ้างอิง

- https://www.aosabook.org/en/posa/zotonic.html

- https://zotonic.com/docs/1277/the-zotonic-data-model

Matplotlib

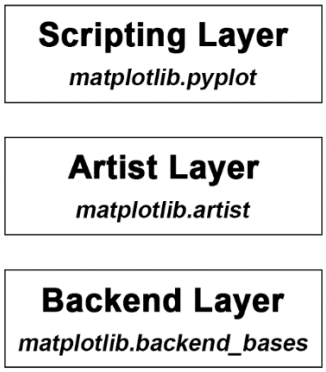
Matplotlib เป็น library ที่ใช้สำหรับการพล็อต โดยมีภาษา Python เป็นฐาน โดยสามารถรองรับ 2D ได้อย่างเต็มรูปแบบ และก็ยังสามารถรองรับกราฟิก 3D ได้แต่ยังมีข้อจำกัด ซึ่งเป็นที่นิยมใช้กันอย่างแพร่หลายในการคำนวณทางวิทยาศาสตร์โดยใช้ Python เป้าหมายของ Matplotlib คือการทำให้สามารถใช้งานได้หลากหลาย สามารถฝังกราฟิกในชุดเครื่องมือการติดต่อกับผู้ใช้งาน และปัจจุบันมารถรองรับกราฟิกแบบโต้ตอบบนระบบปฏิบัติการเดสก์ท็อปหลักทั้งหมดโดยใช้ GTK+, Qt, Tk, FLTK, wxWidgets และ Cocoa ซึ่งสามารถเรียกแบบโต้ตอบจากinteractive Python shell เพื่อสร้างกราฟิกด้วยคำสั่งขั้นตอนง่าย ๆ เช่น Mathematica, IDL หรือ MATLAB matplotlib และยังสามารถฝังตัวใน headless webserver เพื่อจัดเตรียมเอกสารทั้งในรูปแบบ raster-based เช่น Portable Network Graphics (PNG) และรูปแบบเวกเตอร์ เช่น PostScript, Portable Document Format (PDF) และ Scalable Vector Graphics (SVG)

วัตถุประสงค์

เพื่อสามารถแสดงผลข้อมูลได้อย่างรวดเร็วและสามารถบันทึกผลที่ได้ออกมาเป็นรูปภาพได้ง่ายและมีประสิทธิภาพมากขึ้น

Architectural Patterns / Styles

รูปภาพประกอบสถาปัตยกรรมของ Matplotlib



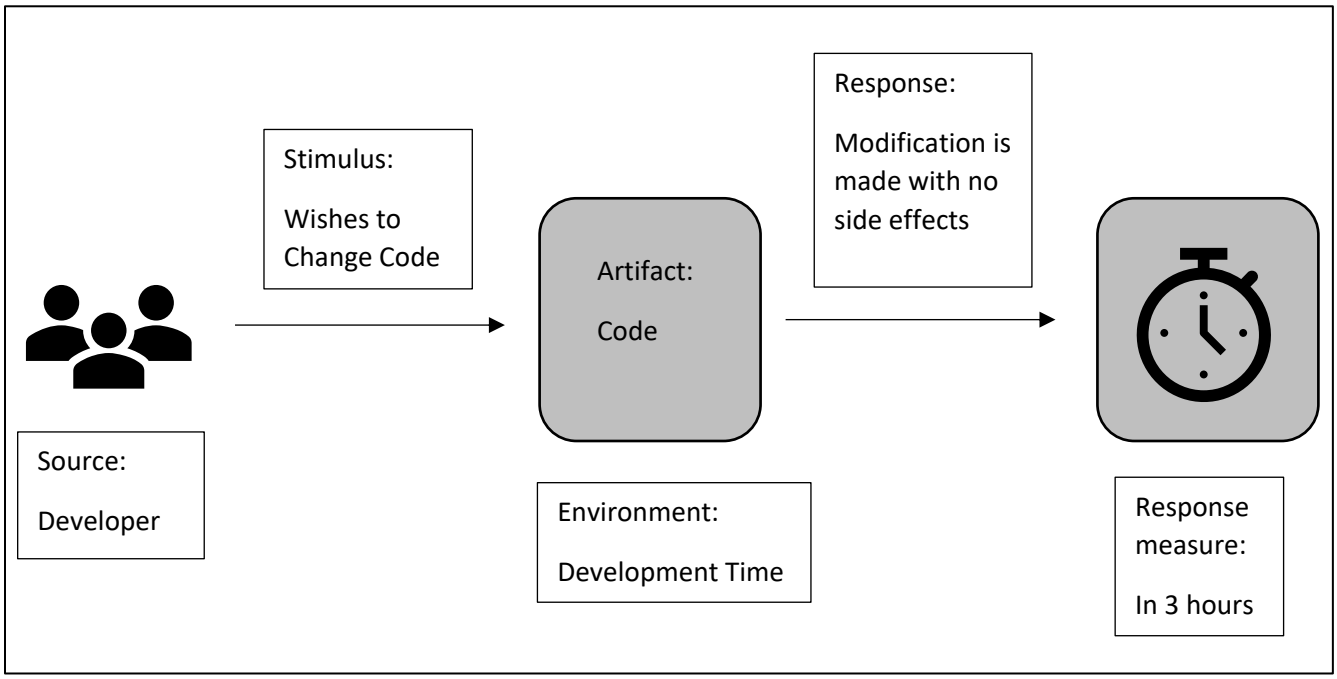
แหล่งที่มาของรูปภาพ https://miro.medium.com/max/875/1*hi9AFzlV-nQyTmbaV3O1kg.png

รูปแบบสถาปัตยกรรมที่ Matplotlib ใช้เป็นรูปแบบ Layer architectural โดยมีทั้งหมด 3 layer

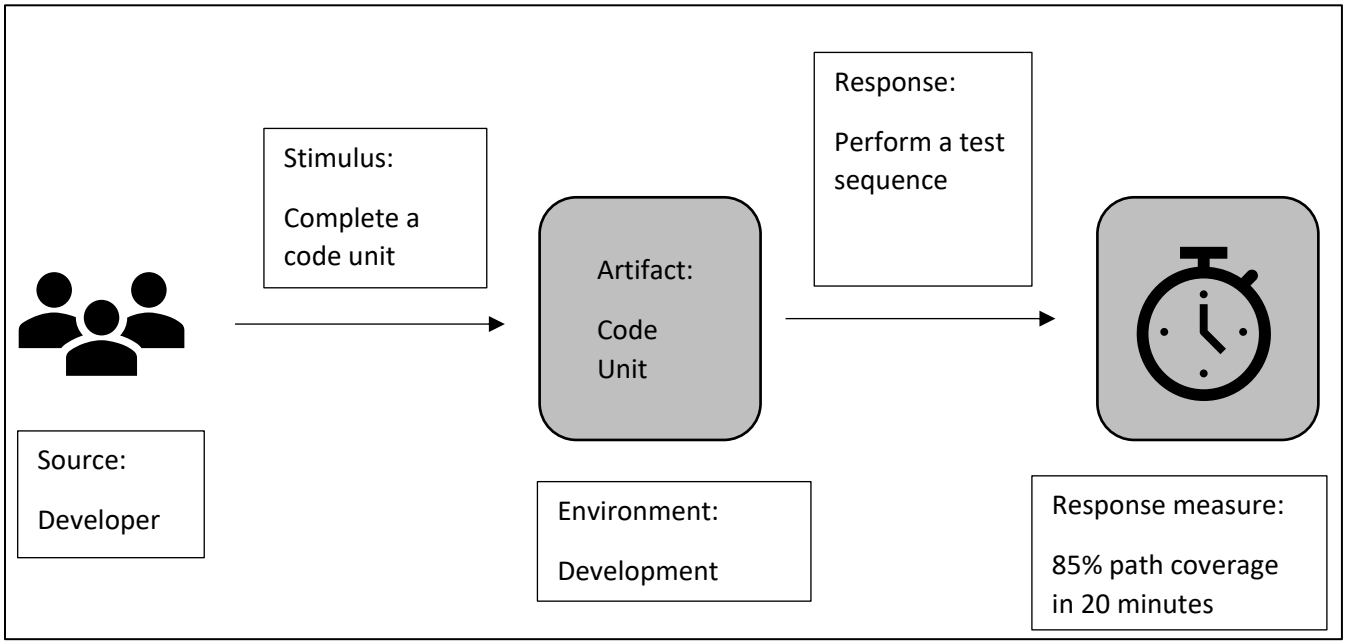
- **Scripting Laye** ถูกออกแบบมาเพื่อทำให้ไลบรารี่ Matplotlib ทำงานคล้ายคลึงกับ MATLAB script และยังเป็นเลเยอร์ที่อยู่ชั้นบนสุดและเป็นเลเยอร์ที่รวบรวมชุดคำสั่งเพื่อให้ง่ายต่อการใช้งาน
- **Artist Layer** เป็น Layer ที่ช่วยให้สามารถควบคุมและปรับแต่งองค์ประกอบต่างๆ ได้มากที่สุด เลเยอร์นี้ประกอบด้วยวัตถุหลักหนึ่งชิ้นคือ Artist ที่ช่วยให้คุณปรับแต่งได้มากขึ้นเมื่อเทียบกับ Scripting Layer และ สะดวกกว่าสำหรับพล็อตขั้นสูง
- Backend Layer เป็นเลเยอร์ที่จัดการงานที่หนักผ่านการสื่อสารไปยังชุดเครื่องมือวาดภาพในเครื่องเช่น wxPython หรือภาษาวาดภาพอย่าง PostScript เป็นเลเยอร์ที่ซับซ้อนที่สุด ประกอบไปด้วย 3 บิลท์อินคลาสหลัก ๆ
    1. FigureCanvas เป็นแคสวาสที่จะแสดงรูปภาพ
    2. Renderer เป็น abstract class ที่จัดการในเรื่องการวาดและการแสดงผลมีหน้าที่ในการวาดใน FigureCanvas
    3. Event จัดการเกี่ยวการป้อนของผู้ใช้งานเช่น การกดคีย์บอร์ดและการกดเมาส์เป็นต้น

Quality Attribute Scenarios

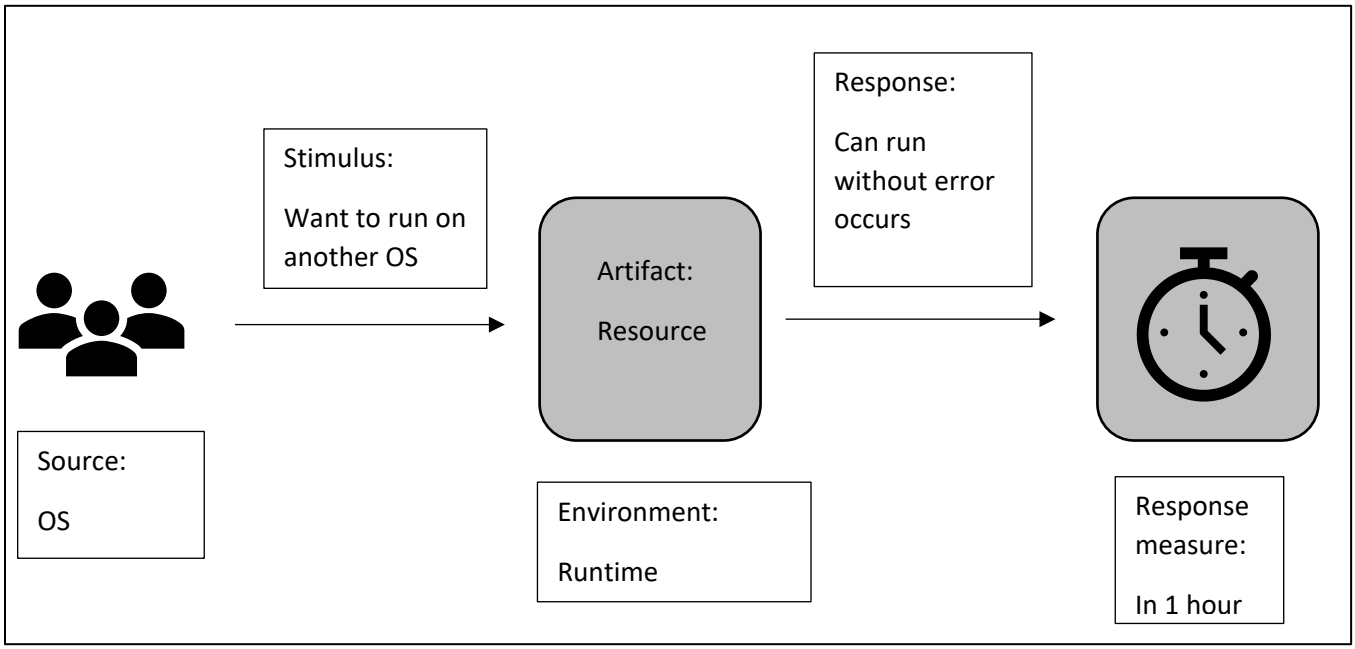
Modifiability



Testability



Portability



แหลงอ้างอิง:- https://www.aosabook.org/en/matplotlib.html

- https://medium.datadriveninvestor.com/data-visualization-with-python-matplotlib-architecture-6b05af533569

YESOD

Yesod เป็นเว็บเฟรมเวิร์กที่เขียนด้วยภาษาการเขียนโปรแกรม Haskell ในขณะที่เฟรมเวิร์กเว็บยอดนิยมจำนวนมากใช้ประโยชน์จากลักษณะไดนามิกของภาษาโฮสต์ Yesod ใช้ประโยชน์จากลักษณะคงที่ของ Haskell เพื่อสร้างโค้ดที่ปลอดภัยและเร็วขึ้น การพัฒนาเริ่มขึ้นเมื่อประมาณสองปีที่แล้วและมีความแข็งแกร่งตั้งแต่นั้นเป็นต้นมา Yesod ในโครงการชีวิตจริงด้วยคุณสมบัติเริ่มต้นทั้งหมดที่เกิดจากความต้องการในชีวิตจริง ในตอนแรกการพัฒนาเป็นการแสดงเพียงคนเดียวเกือบทั้งหมด หลังจากผ่านไปประมาณหนึ่งปีของการพัฒนา ความพยายามของชุมชนก็เริ่มขึ้น และตั้งแต่นั้นเป็นต้นมา Yesod ก็กลายเป็นโครงการโอเพ่นซอร์สที่เฟื่องฟู ในระยะแรกเมื่อ Yesod เป็นเพียงชั่วคราวและไม่ชัดเจน มันคงเป็นผลที่ตรงกันข้ามหากพยายามให้ทีมทำงาน เมื่อมันเสถียรพอที่จะเป็นประโยชน์ต่อผู้อื่น มันก็เป็นเวลาที่เหมาะสมที่จะค้นหาข้อเสียของการตัดสินใจบางอย่างที่เกิดขึ้น ตั้งแต่นั้นมาเราได้ทำการเปลี่ยนแปลงครั้งสำคัญกับ API สำหรับผู้ใช้เพื่อให้มีประโยชน์มากขึ้น ภาษา

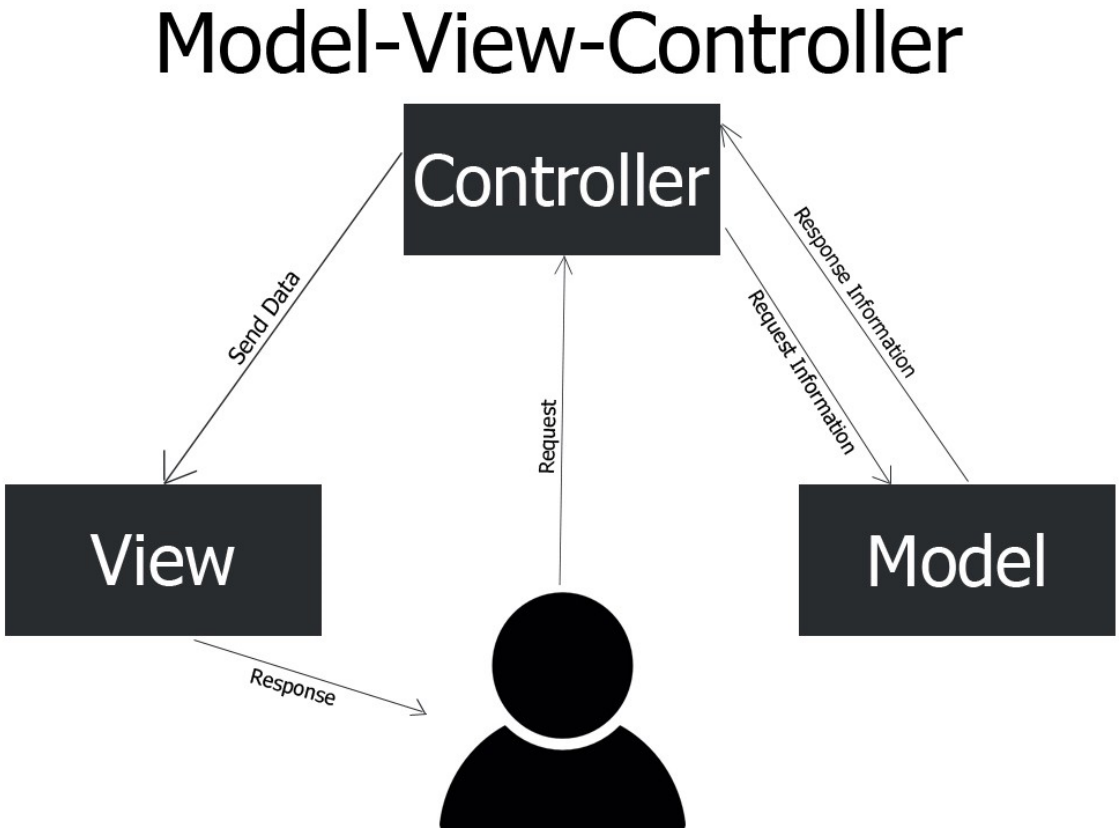
วัตถุประสงค์

เป้าหมายของ Yesod คือการขยายจุดแข็งของ Haskell ไปสู่การพัฒนาเว็บ Yesod พยายามทำให้โค้ดของคุณกระชับที่สุด ทุกบรรทัดของโค้ดของคุณจะถูกตรวจสอบความถูกต้อง ณ เวลาคอมไพล์ให้มากที่สุดเท่าที่เป็นไปได้แทนที่จะต้องใช้ไลบรารีขนาดใหญ่ของการทดสอบหน่วย (unit testing) เพื่อทดสอบคุณสมบัติพื้นฐาน คอมไพเลอร์ทำทุกอย่างเพื่อคุณ โดยเบื่องหลัง Yesod ใช้เทคนิคประสิทธิภาพขั้นสูงมากเท่าที่เราสามารถรวบรวมเพื่อให้โค้ดระดับสูงของคุณทำงาน

Architectural Patterns / Styles

โดยทั่วไปแล้ว Yesod มีความคล้ายคลึงมากกว่าแตกต่างจากเฟรมเวิร์กชั้นนำเช่น Rails และ Django โดยทั่วไปแล้วจะเป็นไปตามกระบวนทัศน์ Model-View-Controller (MVC) มีระบบเทมเพลตที่แยกมุมมองออกจากตรรกะ จัดเตรียมระบบ Object-Relational Mapping (ORM) และมีแนวทางควบคุมด้านหน้าเพื่อกำหนดเส้นทางปีศาจอยู่ในรายละเอียด . Yesod มุ่งมั่นที่จะผลักดันให้เกิดข้อผิดพลาดในขั้นตอนการคอมไพล์แทนรันไทม์ และตรวจจับจุดบกพร่องและข้อบกพร่องด้านความปลอดภัยโดยอัตโนมัติผ่านระบบประเภท แม้ว่า Yesod จะพยายามรักษา API ระดับสูงที่เป็นมิตรต่อผู้ใช้ แต่ก็ใช้เทคนิคที่ใหม่กว่าจำนวนหนึ่งจากโลกแห่งการเขียนโปรแกรมที่ใช้งานได้จริงเพื่อให้ได้ประสิทธิภาพสูง และไม่กลัวที่จะเปิดเผยข้อมูลภายในเหล่านี้แก่นักพัฒนา

สถาปัตยกรรมหลักใน Yesod คือการสร้างสมดุลระหว่างเป้าหมายทั้งสองที่ดูเหมือนจะขัดแย้งกัน ตัวอย่างเช่น ไม่มีการปฏิวัติวิธีการกำหนดเส้นทางของ Yesod (เรียกว่า type-safe URLs) ในอดีต การใช้โซลูชันดังกล่าวเป็นกระบวนการที่น่าเบื่อและเกิดข้อผิดพลาดได้ง่าย นวัตกรรมของ Yesod คือการใช้เทมเพลต Haskell (รูปแบบหนึ่งของการสร้างโค้ด) เพื่อทำให้ต้นแบบอัตโนมัติที่จำเป็นในการบูตกระบวนการ ในทำนองเดียวกัน HTML ที่ปลอดภัยสำหรับการพิมพ์นั้นมีมานานแล้ว Yesod พยายามรักษาลักษณะที่เป็นมิตรกับนักพัฒนาของภาษาเทมเพลตทั่วไปในขณะที่รักษาพลังของความปลอดภัยของประเภท

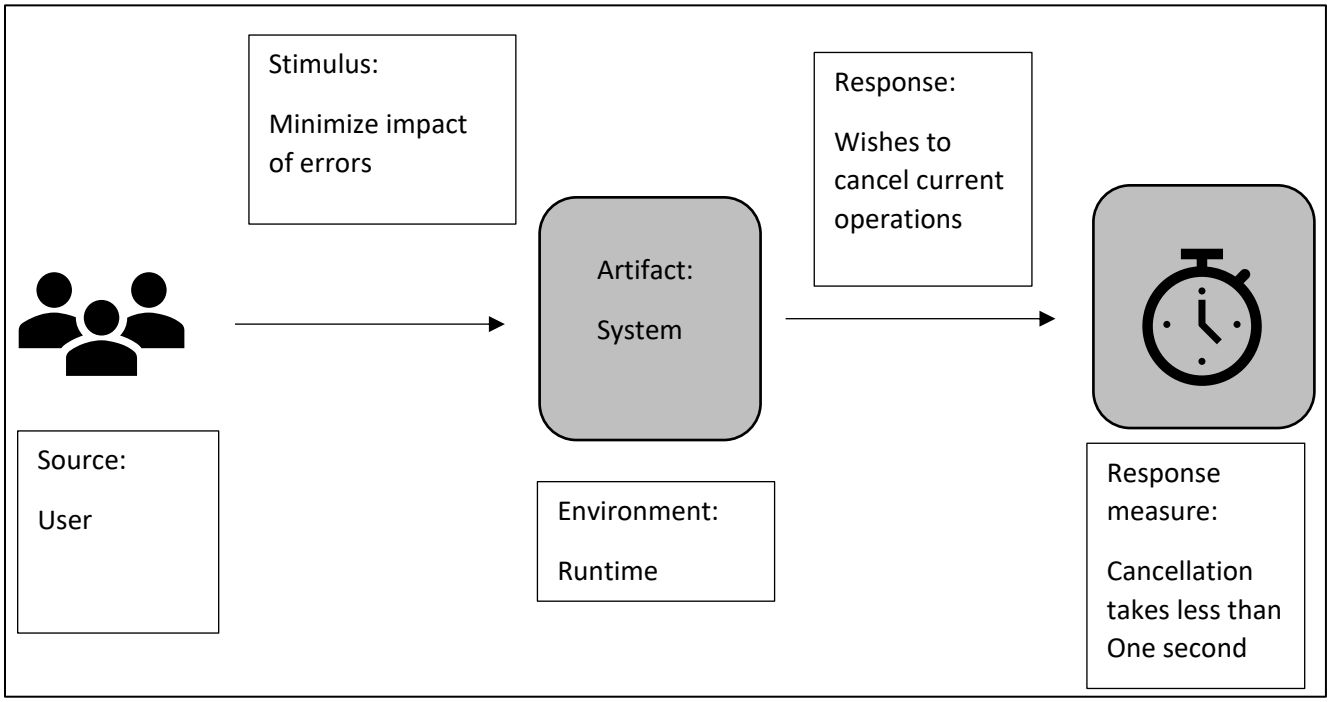


แหลงที่มา https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5

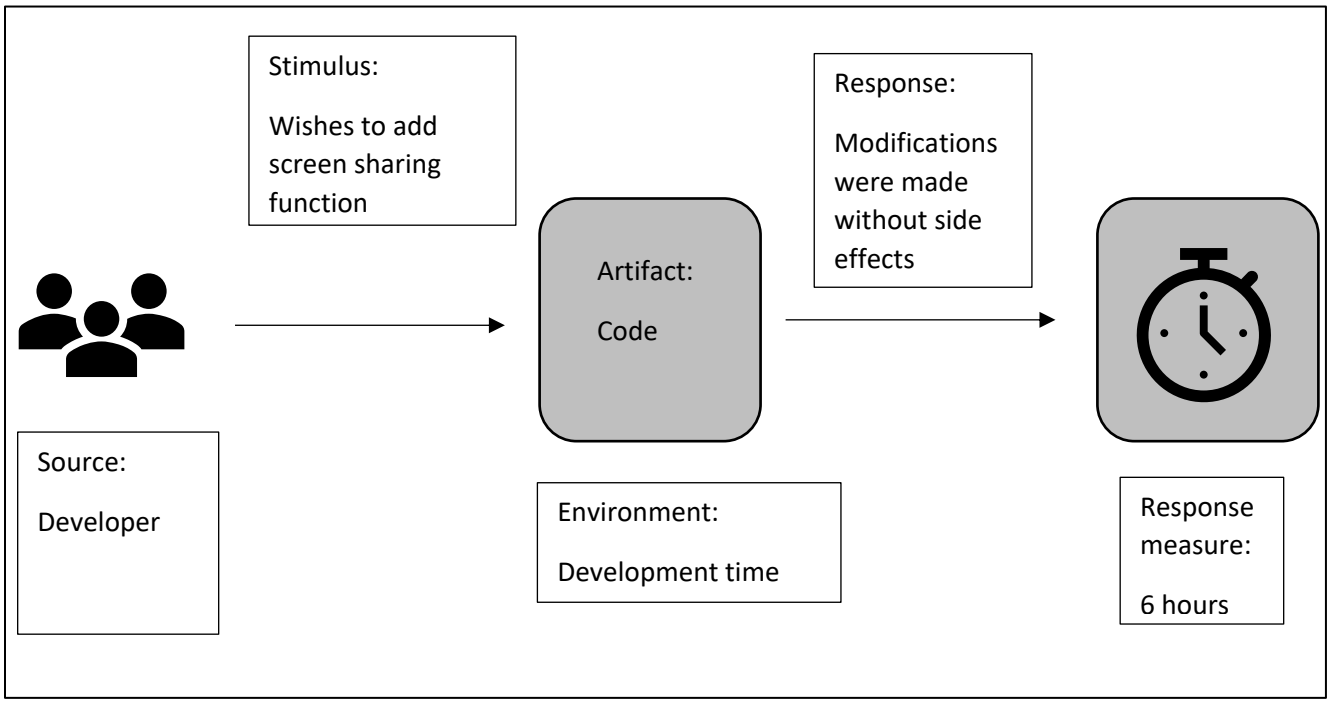
MVC เป็นรูปแบบการออกแบบที่ใช้เพื่อช่วยสร้างเฟรมเวิร์กสำหรับแอปพลิเคชัน ประกอบด้วยสามส่วนที่เรียกว่า Model, View และ Controller มีรายงานว่า MVC ถูกคิดค้นโดย Trygve Reenskaug ในปี 1970 เขาใช้การออกแบบนี้เพื่ออธิบายโครงสร้างซอฟต์แวร์ในแง่ของความรับผิดชอบและนำไปใช้อย่างมีประสิทธิภาพ เป็นการออกแบบที่ได้รับความนิยมและถูกใช้โดยภาษาโปรแกรมทุกประเภท เช่น Java, C#, Ruby และ PHP ในโพสต์นี้ฉันจะพูดถึงแต่ละส่วนและพูดถึงสิ่งที่คุณในฐานะนักพัฒนาซอฟต์แวร์ควรรู้และรวมไว้ในแต่ละส่วน เมื่อฉันอธิบายแต่ละส่วนของโมเดล MVC ฉันจะเชื่อมโยงมันกับการเปรียบเทียบของร้านอาหาร คิดว่ารูปแบบ MVC เป็นร้านอาหารโดยที่นางแบบเป็นคนทำอาหาร มุมมองคือลูกค้า และผู้ควบคุมคือพนักงานเสิร์ฟ

Quality Attribute Scenarios

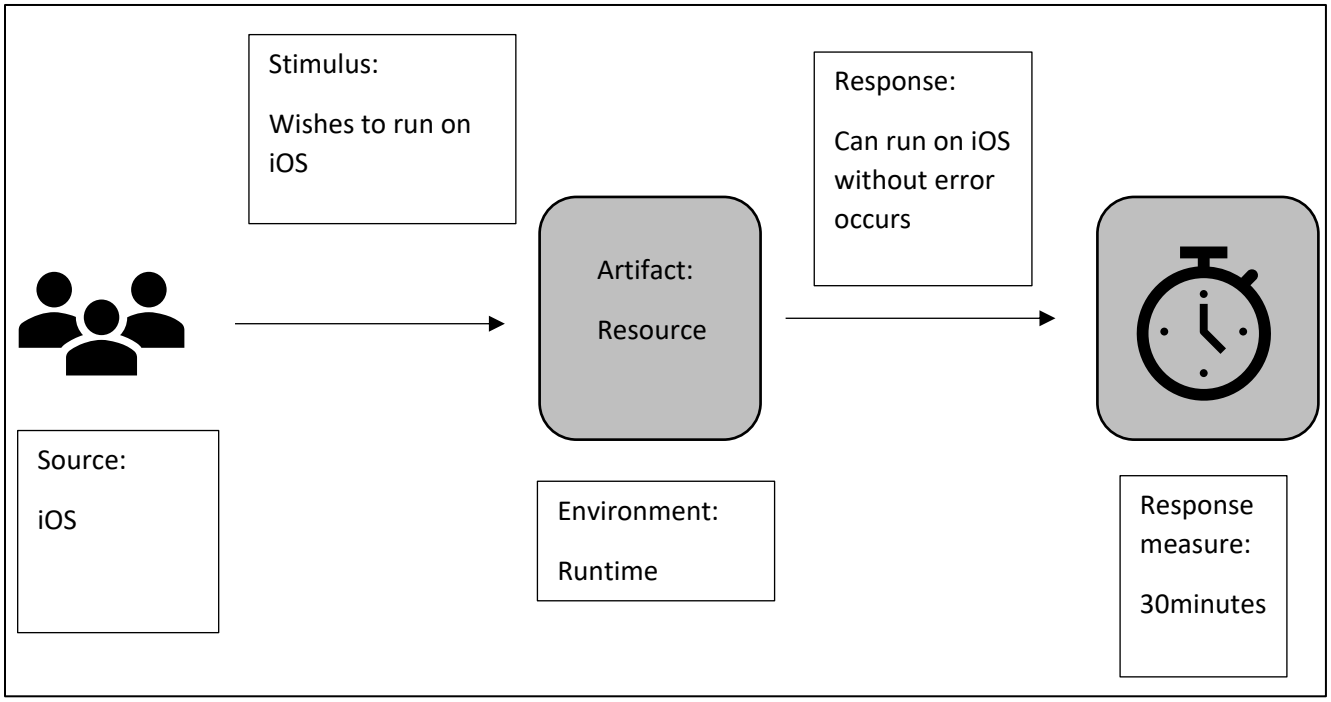
Usability



Modifiability



Portability



แหล่งอ้างอิง

- https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5
- https://en.wikipedia.org/wiki/Yesod_(web_framework)