TTSA05 : Design Technique for High Performance System

Description :

#รู้ไว้ก่อน #จะได้ไม่ต้องรื้อทีหลัง เรียนรู้เทคนิคกันแบบไม่กั๊ก! หลักสูตรดีๆสำหรับสาย #DEV ที่กำลังออกแบบและพัฒนาระบบที่ต้องการคุณภาพด้านประสิทธิภาพสูงๆ (High Performance) และต้องรองรับการ Scale ได้ในอนาคต

Instructor :

avatar
Image not found or type unknown

นายณรงค์ จันทร์สร้อย

-

Training Date : **10 พ.ย. 2569 - 12 พ.ย. 2569**

fee : **12,000 ฿** (ราคายังไม่รวม Vat 7%)

Days & Duration : **3 Day(s) | 18 Hour(s)**

Time : **10:00:00 - 17:00:00**

Language : **Thai**

Venue : **Software Park Training Room 3rd floor, Software Park Building Chaengwattana Road, Pakkred Nonthaburi**

Type : **Classroom**

Category : **Software Architecture and Design**

Objectives :

ปัจจุบันมีวิกฤติสำคัญหนึ่งในการพัฒนาระบบซอฟต์แวร์และแอพพลิเคชั่นมากมาย คือความสามารถในการใช้ทรัพยากรต่างๆ ได้อย่างคุ้มค่าและมีประสิทธิภาพ บ่อยครั้งที่ปัญหาด้านประสิทธิภาพมักพบหลังจากการทำ performance testing ซึ่งอาจช้าเกินแกง การย้อนกลับไปปรับแก้งานออกแบบและโค้ดจึงเป็นเรื่องยาก ทางออกที่มักทำกันหรือมักถูกบอกให้ทำคือการเพิ่มประสิทธิภาพด้านการประมวลผล? บางทีต้องตั้งคำถามกลับว่าการประมวลผลกับทรัพยากรอันไหนสำคัญกว่ากัน? เหมือนกับ CPU กับ memory อันไหนสำคัญกว่ากัน? การเพิ่มหน่วยความจำไม่ค่อยส่งผลนักกับ license ของซอฟต์แวร์ แต่ทำไมหลายครั้งทางออกมักเป็นการเพิ่มหน่วยประมวลผล เช่น เพิ่ม CPU หรือเพิ่มเครื่อง?

โดยเฉพาะอย่างยิ่งระบบมากมายในปัจจุบันมักเป็น component-based application ที่ทำงานบน middleware เช่น แอพพลิเคชั่นเซิร์ฟเวอร์ ระบบลักษณะนี้ต้องการทรัพยากรอย่างมาก การรองรับการขยายตัวของระบบ (scalability) ก็เป็นสิ่งสำคัญที่ต้องคำนึงถึง หรืออีกวิกฤติหนึ่งในปัจจุบันในหลายองค์กรคือการพึ่งพา container หรือเฟรมเวิร์กต่างๆ มากจนเกินไป ปัญหาที่พบบ่อยครั้งคือผู้ใช้มักขาดความเข้าใจในสิ่งเหล่านี้อย่างดีพอ ทำให้ไม่สามารถรับมือได้เนิ่นๆ ตั้งแต่ตอนออกแบบระบบ ฉะนั้นเราจะออกแบบระบบอย่างไรให้ใช้ทรัพยากรได้อย่างคุ้มค่าและมีประสิทธิภาพและรองรับการขยายตัวได้ โดยไม่ต้องมาพบปัญหาเอาตอนทำ performance testing ภายหลังพัฒนาระบบเสร็จหรือใกล้เสร็จ หรือมาเจอเอาตอนใช้งานระบบจริง และจะได้ไม่ต้องตกเป็นเหยื่อด้านค่าฮาร์ดแวร์และ license ของซอฟต์แวร์มหาโหดในปัจจุบัน และมักแฝงเร้นไม่ยอมบอกให้เคลียร์ตั้งแต่เนิ่นๆ มักมาบอกกันตอนเจอทางตันที่ไม่อาจปฏิเสธได้แล้ว

หลักสูตรนี้จึงออกแบบมาเพื่อให้ผู้เรียนมีความรู้ความเข้าใจในคุณภาพด้านประสิทธิภาพ (performance) และการจัดการทรัพยากร (resource management) และนำเสนอแนวทางสำหรับวิเคราะห์และออกแบบระบบที่ต้องการคุณภาพด้านประสิทธิภาพสูง ไม่ว่าจะเป็นระบบประเภท embedded system, mobile application, business application, enterprise system และอื่นๆ เพื่อให้ผู้เรียนตระหนักถึงประเด็นสำคัญ (concerns) ที่ต้องคำนึงเวลาออกแบบระบบที่ต้องการประสิทธิภาพสูงๆ รวมถึงการจัดการด้าน scalability (การรองรับการขยายตัวของระบบ) และสามารถรับมือและจัดการได้ โดยไม่ต้องเสียค่าใช้จ่ายฟุ่มเฟือยที่เกินความจำเป็นภายหลัง

หลักสูตรนี้มุ่งเน้นการสร้างความเข้าใจในหลักการด้านการออกแบบระบบที่ต้องการคุณภาพด้านประสิทธิภาพสูง ซึ่งจำเป็นต้องใช้ทรัพยากรต่างๆ อย่าง 'คุ้มค่า' โดยเน้นทั้งพื้นฐานและหลักการวิเคราะห์และออกแบบ โดยมุ่งหวังให้ผู้เรียนสามารถนำความรู้ที่ได้กลับไปประยุกต์กับงานได้ ไม่ว่าผู้เรียนจะออกแบบและพัฒนาระบบประเภทใดหรือใช้เทคโนโลยีใดอยู่ และหากมีข้อสงสัยต้องการปรึกษากับผู้สอนภายหลังการอบรม อันเนื่องจากจำนวนวันอบรมที่จำกัด ก็สามารถทำได้โดยผ่านเว็บไซต์สังคมออนไลน์ต่างๆ และการติดต่อผ่านทางช่องทางต่างๆ ตามตกลงกันระหว่างผู้สอนและผู้เรียนระหว่างอบรม

Target Group :

- การอบรมนี้ออกแบบมาสำหรับ solution architect, system analyst, software architect, enterprise architect, IT architect, requirement engineer, IT manager, programmer, developer, performance engineer, tester และผู้ที่สนใจทั่วไป

พื้นฐานของผู้เข้ารับการอบรม (Prerequisities)

- ผู้เรียนควรมีความรู้และทักษะพื้นฐานด้านการพัฒนาซอฟต์แวร์, Object-Orientation, การวิเคราะห์และออกแบบระบบ และ Software Development Life Cycle (หลักสูตรนี้มีการใช้ UML บ้าง แต่ไม่มากนัก)

Benefits :

- เข้าใจพื้นฐานด้าน performance และ resource management
- เข้าใจการวิเคราะห์ปัจจัยต่างๆ ที่มีผลต่อการใช้ทรัพยากรและการทำงานของ process ต่างๆ ในระบบ เช่น business process, rules, client behavior, service, resource, transaction data, layer, tier, session, object, data model ฯลฯ
- เข้าใจการออกแบบระบบที่ต้องมีประสิทธิภาพสูงและรองรับการขยายตัวได้
- ได้เรียนรู้ architectural tactic ต่างๆ เพื่อการจัดการด้าน performance โดยเฉพาะ
- ได้เรียนรู้ architectural pattern ต่างๆ ทางด้านการจัดการทรัพยากรสำหรับระบบที่ต้องการประสิทธิภาพสูง
- เข้าใจประเด็นสำคัญอื่นๆ ที่สำคัญต่อระบบที่ต้องการคุณภาพด้านประสิทธิภาพสูงๆ รวมถึงการจัดการด้าน scalability อีกด้วย
- สามารถนำความรู้ที่ได้รับกลับไปประยุกต์กับงานได้
- สามารถปรึกษากับผู้สอนภายหลังการอบรมได้อีกด้วย

Course Outline :

Performance and Resource Management Overview

- What is Performance?
- Understanding the Resource
- Demand and Supply
- Event Source and Arrival Patterns
- Understanding Multi-Process and Multi-Thread
- Resource Consumption
- Resource Constraint and Other Constraints
- Overview of Resource Management
- Scope of Resource Management
- Scheduling Strategies
- What is Granularity?
- Performance Attributes (เช่น latency, response time, responsiveness, throughput, jitter, miss rate, data lost ฯ)
- Understanding the Cost Effect
- Understanding the ACID
- Overview of State Transition and State Management
- Overview of Unit of Work, Transaction and Sub Transaction
- What is Scalability?
- Scalability Methods

Analysis

- Analyzing Business Processes and Rules
- Analyzing Client Behaviors
- Analyzing Service (application service and/or business service)
- Analyzing Resource Systems and Connection
- Analyzing Transaction Data and Data Transformation
- Analyzing Architectural Layers and Tiers
- Using Scenario for Detailing Performance Concerns
- Using Performance General Scenario
- Analyzing variability of Performance and Scalability
- Categorizing Relevant Resources
- Modeling Processes
- Modeling Processes and Resources Matrix
- Analyzing Process and Resource Consumption

Design

- Architectural Tactics
    - Two basic contributors to the response time: Resource Consumption and Blocked Time
    - Resource Demand
        - Increase computational efficiency
        - Reduce computational overhead
        - Manage event rate
        - Control frequency of sampling
        - Bound executions times
        - Bound queue sizes
    - Resource Management
        - Introduce concurrency
        - Maintain multiple copies of either data or computations
        - Increase available resources
    - Resource Arbitration
        - First-in/First-out
        - Fixed-priority scheduling: semantic importance, deadline monotonic, rate monotonic
        - Dynamic priority scheduling: round robin, earliest deadline first
        - Stat scheduling
- Architectural Patterns (from POSA 3):
    - Resource Acquisition
        - Lookup

        - Lazy Acquisition
        - Eager Acquisition
        - Partial Acquisition
    - Resource Lifecycle
        - Caching
        - Pooling
        - Coordinator
        - Resource Lifecycle Manager
    - Resource Release
        - Leasing
        - Evictor
- Short Brief for Recommended Architectural Patterns (from POSA 2):
    - Service Access and Configuration
        - Wrapper Façade
        - Component Configurator
        - Interceptor
        - Extension Interface
    - Event Handling
        - Reactor
        - Proactor
        - Asynchronous Completion Token
        - Acceptor-Connector
    - Synchronization
        - Scoped Locking
        - Strategized Locking
        - Thread-Safe Interface
        - Double-Checked Locking Optimization
    - Concurrency
        - Active Object
        - Monitor Object
        - Half-Sync/Half-Async
        - Leader/Followers
        - Thread-Specific Storage

Additional Concerns

- Handling Variations and Budget
- Recheck about Clustering: Need for Performance or Availability?
- Compromising with Other System Qualities, Especially Modifiability, Scalability and Availability
- Interesting Patterns for Handling Performance and Scalability (เช่น Transaction Script, Unit of Work, Facade, Business Delegate, Proxy, Gateway, Optimistic Offline Lock, Pessimistic Offline Lock, Coarse Grained Lock, Implicit Lock, Decorator, Bridge, Strategy, Composite, Flyweight, Command, Chain of Responsibility, State, Template Method, Domain Model, Repository, Prototype, Repository, Lazy Load, Mapping, State, Memento, Observer, Interpreter, Adapter, Abstract Factory, Factory Method, Builder เป็นต้น)

Payment Condition :

Payment can be made by:

1. Cash or Credit Card or Bank Cheque payable to **"สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ"** (a post-dated cheque is not accepted) on the first day of the service or within the last day of the service.
2. **Account transfer** and send the proof of the payment (the deposit slip) to email ttd@swpark.or.th

    • **ธนาคารกรุงเทพ สาขาอุทยานวิทยาศาสตร์**
    Saving Account Number: **080-0-00001-0**
    Account Name: **สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ**
    • **ธนาคารกรุงไทย สาขาตลาดไท**
    Saving Account Number: **152-1-32668-1**
    Account Name: **สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ**

Notes:

- Withholding tax (3%) is exempt.
- Should you need to withdraw, you must send the notice of the withdrawal in writing no later than 7 working days before the commencement date. The cancellation less than 7 days will be subject to a fine of 40% of the fee.
- Software Park Thailand reserves the rights to cancel courses due to unforeseen circumstances.

Contact Person :

For more information, contact our course coordinator on:

Namfhon Pongyat

Tel: +66-2583-9992 Ext. 81427

Email: namfhon@swpark.or.th

SOFTWARE PARK THAILAND

You are encouraged to use the course schedule as a guide to plan your training.

The schedule is accessible at www.swpark.or.th for more information.