SOA and Microservice Design Patterns and Domain-Driven Design

This course provides an easy to understand, end-to-end overview of contemporary service concepts and technologies pertaining to modern-day microservices and service-oriented computing, as well as business and technology-related topics pertaining to service-oriented architecture (SOA). Providing an in-depth exploration of the overarching models and underlying mechanics of service-oriented technology architecture also will be covered. A wide range of topic areas is covered to provide techniques, insights and perspectives of the inner workings of service and composition architectures, including messaging, microservice deployments, service contracts, API gateways, containerization and many more.

Domain-Driven Design has attracted quite a lot of attention in recent years. This course explains the concepts of DDD, and explores how it can help us model complex software systems. We'll lift the lid on key concepts such as Bounded Contexts, Aggregates and Ubiquitous Language, and take a pragmatic look at how to apply these concepts to address real challenges.

The principles of BDD during this course also will be covered.

The course also covers the Command-Query Responsibility Segregation (CQRS) pattern, and describes how it fits in with DDD.

Duration: 27 hours

Prerequisites:

• Experience in large scale software design
• Familiarity with Object Oriented Design principals

Course outline:

– Business and Technology Drivers for SOA, Services and Microservices

– SOA vs. Traditional Architectures

– Understanding Service and Composition Architectures

– Strategic Goals and Benefits of Service Oriented Computing

–Services vs. Microservices

– Fundamental Characteristics of a Service Oriented Architecture

– Understanding Service Orientation as a Design Paradigm, including the Four Pillars of Service Orientation

– Introduction to Service Layers, Service Models and Service Compositions

– Service Inventories, Service Layers and Service API Governance and Management

– Introduction to Common Service Technologies, including API Gateways, Virtualization, Containerization

– Adoption Impacts, including considerations for Governance, Infrastructure, Performance and Standardization

– Fundamental Application Design with SOA

– Service Orientation vs. "Silo" Based Design

– Service Oriented Application Design with Microservices

– Understanding Services and Service Capabilities

– Understanding the Functional Context of Microservices

– Complex Service Composition Design, Composition Runtime Roles and Responsibilities

– Composition with Microservices

– Distinguishing Characteristics of the SOA Model

– The Eight Design Principles of Service Orientation

– Contract First Design, Standardized Service Contracts and Uniform Contracts

– Service Loose Coupling and Coupling Types, Service Abstraction and Information Hiding

– Service Reusability and Agnostic Design, Service Autonomy and Runtime Control

– Service Statelessness and State Deferral, Service Discoverability and Interpretability

– Design Guidelines for REST Services

– Logic Centralization, Schema Centralization and Canonical Schemas

– Dual Protocols, Canonical Resources and Inventory Endpoints

– Contract Centralization, Official Endpoints and Services with Concurrent Contracts

– Lightweight Endpoints, Reusable and Uniform Contracts

– Service Façades, Legacy Wrappers and Service Data Replication

– Microservice Deployments and Containerization

– Redundant Implementations, Content Negotiation and Idempotent Capabilities

– Messaging Metadata, State Messaging and Event Driven Messaging

– Service Instance Routing, Endpoint Redirection, Service Agents and Intermediate Routing

– API Gateways and Asynchronous Queuing

– Data Format Transformation, Data Model Transformation and Protocol Bridging

– Service Brokers and the Enterprise Service Bus

– Orchestration and Compensating Service Transactions

– Composition Autonomy, Entity Linking and State Repositories

Also the following SOA Patterns will be introduced:

• Agnostic Capability

• Agnostic Context

• Functional Decomposition

• Non-Agnostic Context

• Service Encapsulation

• Agnostic Service Declaration

• Atomic Service Transaction

• Enterprise Service Bus (ESB)

• Federated Endpoint Layer

• Orchestration

• Service Façade

• Service Callback

• Multiple Service Contracts

• Authentication Broker

• Message Origin Authentication

• Message Screening

• Transactional Service

– **Domain-Driven Design**

- **Introduction to Domain-Driven Design:**
    - What is Domain-Driven Design (DDD);
    - What challenges does DDD address;
    - Essential patterns and practices of DDD

- **Problem Domains:**
    - Understanding the problem domain;
    - Techniques and practices;
    - Core domains;
    - Defining clean boundaries

- **Effective Model-Driven Design:**
    - Domain models;
    - Defining a Ubiquitous Language;
    - Domain model implementation patterns;
    - Bounded Contexts;
    - Applying DDD in practice

- **Bounded Context Integration:**
    - Integrating Bounded Contexts;
    - Integrating distributed Bounded Contexts;

- **Overview of DDD Patterns:**
    - Entities, value objects, and domain services;
    - Aggregates, factories, and repos;
    - Domain events;
    - Event sourcing

- **Value Objects and Entities:**

    - When to use a value object;
    - Persistence options;
    - How to implement entities;
    - Entity best practices

- **Domain Services and Domain Events:**
    - When and how to use domain services;
    - Domain services vs. application services;
    - Domain service patterns;
    - Domain event actions;
    - Implementation options

- **Aggregates:**
    - The importance of aggregates in DDD;
    - Defining aggregate boundaries;
    - Implementing aggregates;
    - Persistence

- **Factories and Repositories:**
    - The purpose of factories;
    - The purpose of repositories;
    - Patterns and anti-patterns

- **Introduction to CQRS and Event sourcing:**
    - What is CQRS;
    - Eventual consistency;
    - Commands and command buses;
    - Repositories;
    - Events and event buses

- Event sourcing

– Behavior Driven Development (BDD)

- The fundamental principles and practices of BDD
- Feature, Scenario and Behavioral specifications: Given – When – Then Approach
- Example Mapping & Discovery Workshops to achieve shared understanding
- The importance of a ubiquitous language for problems and solutions
- Key differences in TDD and BDD
- Writing first scenario for your product
- Using parameters in step definitions
- Roles and responsibilities on a BDD team