Eğitim Kataloğu

2017

http://academy.eteration.com

academy@eteration.com

eteration bilişim çözümleri ve ticaret anonim şirketi itü arı-3 teknokent B202 maslak Istanbul 34469 türkiye tel: +90 212 328 0825 fax: +90 212 328 0521

1. ETERATİON HAKKINDA

2002 yılında kurulan eteration, Türkiye, Avrupa ve Asya'da yer alan çeşitli sektörlere üstün yazılım, danışmanlık ve eğitim hizmetleri sunmayı hedefleyen bir Türk yazılım ve AR-GE kuruluşudur. Kurumsal Java, Hizmet Tabanlı Mimari (SOA), Nesne Tabanlı Modelleme, Analiz, Web Teknolojileri ve Kurumsal Mobil Çözümleri konularında kapsamlı, disiplinli ve metodolojik yaklaşımı ile öne çıkarak tercih nedeni olmaktadır. eteration, ürün satış, danışmanlık ve çözüm üretimi konularında uluslararası IT tedarikçileri ile iş ortaklığı yapmaktadırlar. Kurulduğu günden beri istikrarlı bir büyüme gösteren eteration üç kişi ile başladığı serüvenine bugün uluslararası alanda IT uzmanlığına sahip kırkın üzerinde tecrübeli mühendisten oluşan bir ekiple yoluna devam etmektedir.

Hizmet verdiği firmalar arasında Türkiye'nin üç büyük GSM operatörü, büyük finans kuruluşları ve kamu kuruluşları yer almaktadır. eteration AR-GE şirketi kimliğiyle İTÜ ARI Teknokent 'te faaliyetlerine devam etmektedir.

Kurumlar, bir taraftan karmaşık iş problemleri ve teknolojik ortamlarla mücadele etmekte diğer taraftan baskı altında büyük kısıtlamalarla karşı karşıya kalmaktadırlar. eteration olarak, açık ve standartlar üzerine kurulu çözümlerimizi, kendi değerlerimizi, tutkumuzu ve yaratıcılığımızı müşterilerimizle paylaşmaya inanarak; bilgi birikimimizle, ruhumuzla, yeteneklerimizle, firmaların daha iyi, daha hızlı, sürekliliği devam eden çözümleri, kendi yöntemleri ile daha hızlı yönetmeleri için gerekli araç ve hizmetleri sağlamaktayız. Biz, birbirimizle, iş ortaklarımızla ve müşterilerimizle birlikte çalışmaktayız. Biz, yükümlülüklerimiz için sorumluluk duymaktayız. Bu şekilde daha yakın, sürekliliği devam eden, daha verimli ilişkiler kurabilmekteyiz.

Hizmet verdiğimiz firmaların, iş dinamiklerine sezgileri ile daha hızlı karşılık verebilmesi için;

- Açık platformlar ve yeni teknolojileri anlamalarını sağlayıp, etkin kullanarak,
- Daha çevik ve rekabetçi olmalarını sağlayarak, onları güçlendirmeyi hedeflemekteyiz.

Teknoloji uzmanlığı, proje geliştirme servisleri, danışmanlık, eğitim, yeni nesil platform ve araçlar konusundaki araştırma geliştirme hizmetlerindeki yetkinliklerimiz; bütünleşmiş hizmetleri yaratma konusundaki yaklaşımımızın temelidir.

Benzer hedefleri ve platformları paylaşan yazılım sağlayıcılar ve iş ortaklarıyla doğru birliktelikler kurarak, öncü teknolojilere erişmesine ve bu teknolojileri hizmetlerinin çevik ve etkin kullanılmasına imkân sağlarken, firmalara sunduğumuz hizmetlerin toplam sahip olma maliyetini de düşürmektedir.

Uzmanlık Alanlarımız

- Servis tabanlı çözümlerin mimari çalışmalarının ve üretiminin gerçekleşmesi
- Karmaşık web portal ve içerik çözümlerinin tasarlanması, üretilmesi

- Kurumsal mobil çözümlerin uygulamaların geliştirilmesi
- Kurumsal Middleware uygulamalarının entegrasyonunun gerçekleştirilmesi
- Yazılım proje yönetimi, mimari ve yazılım geliştirme süreçleri ve uygulamaları konularında eğitim ve danışmanlık hizmetleri
- Yazılım geliştirme süreçlerini hızlandıracak ve katkıda bulunacak altyapıların ve araçların geliştirmesi
- IBM WebSphere, IBM e-commerce, IBM Collaboration Solutions & Social Business , CA APM ürün ailelerine yönelik çözüm, danışmanlık ve destek hizmetleri sunulması
- B2B, B2C çözümlerinin oluşturulması

2. EĞİTİM HİZMETLERİMİZ

Eteration eğitimleri, yazılım ve uygulama geliştirmeye yönelik süreçler, analiz, tasarım ve programlamayı kapsayan konuları içermektedir

Uygulama geliştiricilerin proje öncesindeki teknik alt yapılarının oluşturulmasına yönelik olarak hazırlanan eğitimler, başarılı bir uygulama geliştirmek için gerekli olan teorik ve pratik donanımları katılımcılara kazandırmayı hedeflemektedir. Eğitimlerimiz, uzun yıllardır uygulama geliştirme ve proje tecrübesine sahip eğitmenler tarafından verilmesiyle klasik anlamdaki sınıf eğitimlerinden ayrılmaktadır.

Eteration eğitimleri , sizlerle yapılan kapsam ve içerik belirleme çalışmalarıyla proje ihtiyaçlarına cevap verecek şekilde düzenlenebilmektedir. Geniş katılımlı eğitimler is e, katılımcıların eğitim merkezine seyahatlerini ortadan kaldırmak ve yapılacak proje ile ilgili ortama yakın olabilmek için firmaların kendi lokasyonlarında verilebilmektedir..

Eğitimlerimiz aşağıda belirtilen formatlarda verilmektedir:

- **Firmanın Kendi Yerinde Eğitim (Onsite)**

Proje ihtiyaçlarına göre düzenlenmiş, firma tarafından sağlanan lokasyonda verilen kurslar.

- **Açık Kurslar (Public)**

Eteration Eğitim Merkezi'nde verilen sınıf eğitimleri.

Eğitim araçları, materyalleri ve ortamı

Eteration, yüksek nitelikte eğitmen yada eğitmenleri, tüm gerekli eğitim içerikve materyalleriyle eğitimin hedeflenen kalitede olmasını sağlamaktadır. (eğitim kitapları, CD ler, gerekli yazılımlar, vb.)

Firmanın kendi yerinde düzenlenecek eğitimler için ortamın, çalışma alanlarının, araçların sağlanmasında firma kendi kaynakları kullanılır. (Her bir katılımcı için uygun özelliklere sahip bilgisayar, sunum cihazı, günlük çalışma alanından ayrı bir mekân, vb…)

3. EĞİTİMLERİMİZ

Kategorilere göre egitimlerimizin listesi aşağıdaki gibidir.

3.1 Analysis and Design

3.1.1 Object Oriented Analysis and Design using UML

Code

M-OOAD-501-001

Overview

This 3-day intensive course is a language neutral introduction of object oriented software development. Object oriented analysis and design techniques will be trained using methods from the Unified Modeling Language (UML). What you will learn At the end of this course you will be familiar with the concepts of object oriented software development, and you will be able to apply object oriented analysis and design methods for simple applications. You will know the most important diagram techniques and will be able to join a development team and actively participate in the design of an application.

Description

- The OO life cycle
- Object oriented concepts in depth
- Encapsulation, Interfaces Inheritance, Polymorphism
- Requirements analysis
- Determine object candidates
- Assigning responsibilities
- CRC-Cards
- Design techniques and diagrams
- Use case diagrams
- Class diagrams
- Component diagrams
- Interaction diagrams
- Activity diagrams

- State/Transition diagrams
- Software design patterns overview
- Frameworks overview
- Software development processes

Audience

Designers, architects, developers and project managers who develop software in an object oriented language like Java or C++ Designers who want to apply proper analysis and design techniques in order to code more robust and better maintainable code *Lectures and demos combined with hands-on exercises using computer-based labs.

Duration

3 days

Format

Instructor Lead

Prerequisites

Experience in application development and basic OO skills such as those from the course eJava111 Developing Object-Oriented Programs in Java™.

3.2 ARCHITECTURE

3.2.1 Effective IT Architecture

Code

A-ARCH-521-001

Overview

Effective IT Architecture is a comprehensive training course that will jumpstart your way to becoming a IT architect. The course is an interactive introduction to IT architecture and what it means to be a IT architect. It's aimed at software developers who are looking towards their first IT architect role, developers who want to become more architecturally aware and IT architects who are new to the role. Throughout the course you'll reinforce everything you learn by defining the architecture for a small software system through a series of hands-on exercises.

What will you learn

- Understand what IT architecture is all about.
- Understand what it means to be a IT architect and the responsibilities associated with the role.
- Understand the trade-offs that are made when making architectural decisions.
- Experience what it feels like to be an architect on a bespoke software development project; including gathering non-functional requirements, determining the drivers for architecture and defining an architecture.
- Appreciate that even a little architecture can go a long way to building better software.

Description

1. Software Architecture Overview
   - The domain of architecture
   - Misconceptions about architecture
   - Architecture defined
   - Architecture Meta-model
   - Responsibilities of the architect
   - Process best practices
   - The development life-cycle
   - Life of an Architecture
   - Overview of the UML
2. Architectural Analysis and Design
   - Types of Architecture

- Architectural Design and Principles
- Architecture vs. Design
- Layering and Tiers
- Components and Connectors
- External Interfaces
- Use Case Maps
- Contextual Factors

3. Architectural Views
   - Cross functional mechanisms
   - Views of System Architecture: Logical View, Process View, Implementation View, Deployment View, Use Case View
   - Elements of Conceptual Architecture
   - Data Models
   - Execution Architecture
   - Concurrency
   - Application Components
   - Prototyping
4. Architectural Styles and Quality Attributes
   - Implementation Styles
   - Performance
   - Usability
   - Reliability
   - Security
   - Frameworks
   - Patterns
   - Tiers and Abstraction Layers
5. Service-Oriented Concepts
   - What is service-oriented architecture
   - What is a service
   - What is service-orientation
   - SOA Technologies & Platforms
   - Reference Architecture
6. Service Engineering
   - Current Enterprise Practices
   - Service Engineering Overview
   - Service Analysis
   - Service Architecture
   - Service Delivery
   - Service Management
   - SOA Governance
7. Service Virtualization and Enterprise Service Bus
   - Service Virtualization and ESB

- Service Integration Patterns

Audience

Project managers, Application Architects, Designers.

Duration

1 day

Format

Instructor Lead

Prerequisites

3.2.2 Building Microservices

Code

A-MSRVS

Overview

Microservice architecture, is a distinctive method of developing software systems that has grown in popularity in recent years. In fact, even though there isn't a whole lot out there on what it is and how to do it, for many developers it has become a preferred way of creating enterprise applications. Thanks to its scalability, this architectural method is considered particularly ideal when you have to enable support for a range of platforms and devices—spanning web, mobile, Internet of Things, and wearables—or simply when you're not sure what kind of devices you'll need to support in an increasingly cloudy future.

The big idea behind microservices is to architect large, complex and long-lived applications as a set of cohesive services that evolve over time.

Topics Covered

1. Microservices Architecture Context
    - What are Microservices?
    - SOA vs. Microservices
    - Examples of Microservices
    - Other Decompositional Techniques
        - Shared Libraries
        - Modules
    - Principles of Microservices
    - When Shouldn't You Use Microservices
    - Microservice Pros and Cons
    - Splitting the Monolith
    - Deployment
    - Testing
    - Monitoring
    - Security
    - MicroServices at Scale
    - The Future of Microservice Architecture
2. How to Implement Microservices ?
    - Building Microservices with Spring Boot
    - Building Microservices with Java POJOs
    - Building Microservices with OSGI
    - Building Microservices with Java9 Jigsaw
3. Introduction to Docker

Audience

Application Architects, Software Developers, Designers.

Duration

4 days

Format

Instructor Lead

Prerequisites

Core Java Syntax - (D-EJAVA-301-001) Effective JAVA Programming, JavaEE, Spring Framework

3.2.3 Docker

Code

D-DCKR

Overview

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud.

Topics in this course :

- Docker architecture
- Installing the Docker Engine
- Creating our first Docker container
- Building Docker images
- Storing and retrieving Docker images from Docker Hub
- Building containers from images
- Deploying applications with Docker
- Networking Docker containers
- Data persistence with Volumes
- Using Docker into a Continuous Integration and Deployment process

Audience

Web Developers, Application Developers

Duration

1 day

Format

Instructor Lead

Prerequisites

3.3 CLOUD

3.3.1 Developing Cloud Applications

Code

D-CLOUD-371-001

Overview

Cloud computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access.

This program is comprehensive hands on workshop for learning the basics of cloud computing technology and using more than one the most promising platform to develop applications and hosting them on cloud infrastructure. You will we developing and hosting a cloud web application by the end of the workshop.

Description

Topics in this course :

1) **Introduction to Clouds**
   a) What is cloud computing?
   b) Why it is popular now?
   c) How to design applications and datastores to run on clouds
   d) Cloud Platforms roundup; Sun Cloud, Amazon EC2, Google App Engine, Azure, Rackspace

2) **Cloud Development**
   a) Amazon EC2
      - Limitations
      - SDK
      - Database
        - Using SimpleDB
        - Using Amazon Relational Database Service
      - Messaging

        - Simple Queue Service
        - Simple Notification Service
    - Scaling & Elastic Load Balancing
    - Running your own custom server instances
    - Development and Deployment
    - Billing
    - Building a sample Application using APIs and Services and deployment

b) Google App Engine Java

    - Limitations & JRE Whitelist
    - SDK
    - Database
        - BigTable datastore model
        - Rethinking and Building up relations
        - Using JPA and JDO
        - Database workarounds
    - Services
        - Memcache
        - URL Fetch
        - Mail
        - XMMp
        - Scheduled Tasks
        - Task Queues
    - Development, Deployment and Versioning
    - Using admin console
    - Billing
    - Building a sample Application using APIs and Services and deployment

3) Developing Cloud Applications Using Eclipse

a) Google App Engine Plugin for Eclipse

- Eclipse App Engine Plugin
- Eclipse GWT plugin
- DataNucleus and Jetty
- Local development and testing
- Deployment from eclipse to clouds

b) AWS Toolkit for Eclipse

- AWS SDK for Java
- Amazon SimpleDB Management
- Amazon EC2 Management

Audience

Developers who aim to develop Cloud applications

Duration

3 days

Format

Instructor Lead
Three-day instructor-led class with approximately 50% hands-on labs

Prerequisites

Core Java Syntax - (D-EJAVA-301-001) Effective JAVA Programming

3.4 Code Quality

3.4.1 Writing Elegant and Readable Code

Code

D-CLN-C

Overview

If you're a developer, then there probably have been times when you've written code and, after a few days, weeks, or months, you looked back at it and said to yourself "What does this piece of code do?" The answer to that question might have been "I really don't know!" In that case, the only thing you can do is going through the code from start to finish, trying to understand what you were thinking when you wrote it

In this course, we will give you practical techniques for writing clean code. We won't be talking about specific architectures, languages, or platforms. The focus lies on writing better code.

Topics Covered:

- CODE SHOULD BE EASY TO UNDERSTAND
    - What Makes Code “Better”?
    - The Fundamental Theorem of Readability
    - Is Smaller Always Better?
    - Does Time-Till-Understanding Conflict with Other Goals

- SURFACE-LEVEL IMPROVEMENTS
    - Packing Information Into Names
    - Names That Can't Be Misconstrued
    - Aesthetics
    - Knowing What To Comment
    - Making Comments Precise And Compact
- SIMPLIFYING LOOPS AND LOGIC
    - Making Control Flow Easy To Read
    - Breaking Down Giant Expressions
    - Variables And Readability
- REFACTORING YOUR CODE
    - Principles in Refactoring
    - Bad Smells in Code
    - Building Tests
    - Toward a Catalog of Refactorings
    - Composing Methods
    - Moving Features Between Objects
    - Organizing Data
    - Simplifying Conditional Expressions
    - Making Method Calls Simpler
    - Dealing with Generalization
    - Extracting Unrelated Subproblems
    - One Task at a Time
    - Writing Less Code
    - Big Refactorings

Audience

Developers who aim to write better code.

Duration

1 day

Format

Instructor Lead

Prerequisites

Any programming language experience.

3.5 Eclipse

3.5.1 Developing Plug-Ins for Eclipse

Code

D-PLUGIN-315-001

Overview

Best way to extend the power of Eclipse-based tools is by building plugins. This course is based on the contents of the book, 'Eclipse: Building Commercial-Quality Plug-ins', by Eric Claybergand Dan Rubel. and the best education from eteration's eclipse experts. In addition to introducing the basics of plugin development, the course will cover how to turn them into commercial quality extensions. In this course you will learn:

- The fundamentals of plugin development
- Solutions to commons problems and challenges
- Latest Eclipse APIs Eclipse: Building Commercial-Quality Plugins
- Understand the concepts of the Eclipse platform
- Principles of extending the Eclipse platform with Plug-ins
- Be able to create your own plug-ins for Eclipse or WebSphere Studio
- Be able to test and deploy your plug-ins

Eclipse and Plugins:

Eclipse is an open source application development environment which is becoming the de-facto standard in Java development. The Eclipse project is also expanding its scope to other languages like C++ and other phases of the application development process like analysis, design and testing. The Eclipse platform is built on a mechanism for discovering, integrating, and running modules called plug-ins and provides an extensible framework for generic application development scenarios like Generating web sites, Embedded Java programs, C++ programs, Enterprise JavaBeans, etc. This 2-day advanced course focuses on how to extend your development tools to be plugged into Eclipse or IBM's WebSphere Studio.

Description

Topics Covered:

- The Eclipse Platform
- Architecture
- The Plug-in concept
- Development Environment
- Eclipse Basics
- Platform Runtime, Workspace, Projects
- Workbench, Views, Editors, Perspectives
- Java Development Tools
- Package Explorer
- Hierarchy View
- Outline View
- Java Source Code Editor
- Eclipse Front-End Frameworks
- JFace
- Standard Widget Toolkit (SWT)
- Layout Managers
- Plug-In Extension Points
- Views, Editors
- Menus, Action Sets

- The Plug-in Toolkit
- Plug-in-Wizards
- Plug-in Manifest Editor
- Deploying
- Testing

Audience

This course is for tool developers who would like to build new extensions or broaden the reach of their development tools for the popular Eclipse platform

- Be familiar with object oriented programming in Java.
- Be familiar with XML terminology.

Duration

2 days

Format

Instructor Lead
*Lectures and demos combined with hands-on exercises using computer-based labs.

Prerequisites

Excellent knowledge of Java, JavaEE and object-oriented programming • Experience using the Eclipse IDE is desirable.

3.5.2 Developing Eclipse RCP applications

Code

D-RCP-351-001

Overview

If you are familiar with the nuts and bolts of developing Eclipse RCP applications and now face a major project requiring specific RCP knowledge, this intensive three-day class covers advanced RCP development concepts that have proven relevant in challenging large-scale projects. Particular attention is paid to sharing best practices that our coaches' derive from their individual project experience. Throughout the training, participants will be given the opportunity to apply theoretical contents in several labs.

Description

This course covers:

- Plugin Philosophy
  - Compartimentation
  - Lazy loading
  - Loose Coupling
- Wizards
  - Contributing wizards
  - Defining a wizard extension
  - Implementing a wizard
  - Implementing a wizard page
  - Customizing a wizard
- Jobs
  - Concurrency
  - Monitoring progress
- Adapters
  - Using the extension object pattern the Eclipse way
  - Data Binding

- Synchronizing controls and (presentation) model
    - Virtual Trees and Tables
    - How to deal with large data sets
- Defining Extension Points
    - The flip side of the coin
- Forms API
    - Customizing the Look and Feel
    - Presentation API
- Headless Build
    - How to build Eclipse RCP products
- Help System
    - User guidance the Eclipse way
- The Next Generation
    - Provisioning mechanism
    - Update sites
    - Stealth updates

Audience

Developers who aim to develop eplipse RCP applications.

Duration

3 days

Format

Three-day instructor-led class with approximately 50% hands-on labs.

Prerequisites

Core Java programming skills and OOP language experience.

3.6 JAVA

3.6.1 Developing Object-Oriented Programs in Java

Code

D-JAVA-201-001

Overview

This intermediate course uses an example-based approach to provide an overview of the object-oriented paradigm and to illustrate the evolutionary development approach supported by Java™. At the end of this course you will be familiar with the core components and packages of the Java™ Standard Edition and you will be able to apply object-oriented programming principles with Java™, Java™ syntax and semantics. You will have a clear understanding of advanced Java™ topics and Java new features.

Course Content

1. Setup Development Environment
   1.1. Installing Java Standart Edition (JDK)
   1.2. Introduction to Eclipse
   1.3. Installing and running eclipse
   1.4. Using Eclipse as development environment
2. Object Oriented Concepts
   2.1. Encapsulation, Inheritance and Polymorphism
   2.2. OO analysis and design: "Is a" and "Has a"
   2.3. Designing an OO application step by step
3. Java SE Language Fundamentals
4. Primitive Data Types
5. Control Statements
6. Classes and Methods
7. Type Casting
8. Inheritance
9. Interfaces
10. Core Class Library
11. Collections and Streams
12. Exception Handling
13. Generics,Compile-time type safety.
14. Enhanced Iterators

15. Autoboxing/Unboxing
16. Typesafe Enums
17. Varargs
18. Annotations
19. JDBC
20. JavaDoc
21. Junit
22. Apache Ant
23. Debugging

Audience

This course is designed for developers, software and system architects and project managers involved with the development of Java™ applications.

Duration

4.5 days

Format

Instructor Lead

Prerequisites

Experience in the following areas is required: Some prior programming experience in a procedural or object-oriented language.

3.6.2 Effective JAVA Programming Using Design Patterns

Code

D-EJAVA-301-001

Overview

This is an advanced Java™ Programming training course that teaches Java developers how to use design patterns and the latest advanced Java language skills effectively.

With the advent of Java 5 and Java 6, the language has seen profound improvements of which not all developers are aware. This course highlights those improvements, as well as delving into a range of topics that an experienced Java developer needs, such as design patterns, performance (JVM, Garbage collection,memory leak, profiling etc), concurrency and refactoring issues: skills that underpin best Java development project practice worldwide.
Topics include:

- Study of best Java development project practice worldwide
- Effective use of advanced Java language constructs
- Study and applications of over 20 Design Patterns
- Performance, concurrency and refactoring

Description

Applying OO Concepts with Java

- Objects and Messages
- Encapsulation, minimizing accessibility and mutability
- Polymorphism
- Subtyping and Subclassing
- Composition versus Inheritance
- Design patterns with Java - an overview
- Structural patterns
  - The Composition pattern
  - The Adapter pattern
  - The Bridge pattern
  - The Decorator Pattern

Effective Java

- Creating and destroying Objects
- Creational Design Patterns
  - The Factory Method provides
  - The Abstract Factory Method
  - The Builder Pattern
  - The Prototype Pattern
  - The Singleton Pattern

- Common Methods and why they are important
  - equals and hashCode
  - toString
  - cloning, deep and shallow copying
  - comparing object

- Managing object behavior with patterns
  - Events and changes
  - The chain of responsibility
  - The Observer pattern
  - The Mediator defines
  - The Chain of Responsibility
  - The Command pattern

- Managing Object State and Function
  - The Visitor pattern adds function to a class
  - The State pattern

- Generics
  - Implementing typesafe heterogeneous containers
  - Generic types and methods
  - Lists versus arrays, foreach versus other loops
  - The Iterator pattern

- Enums and Annotations
  - Annotations versus Naming patterns
  - Defining your own annotations

- Exceptions
  - Exceptions for exceptional conditions
  - Programming errors
  - Recoverable conditions
  - Unnecessary usages
  - Standard versus Custom exceptions
  - Documenting Exceptions
  - Capturing failure information
  - Ignored exceptions

- Concurrency
  - Accessing shared mutable data
  - Effective use of synchronization
  - Executors, tasks and threads
  - Lazy initialization and concurrency

Refactoring Java Code

- The basics of refactoring: Detect, characterize, design, modify
- When to refactor
- Tools
- Moving a class
- Extracting methods
- Extracting supertypes
- Conditionals vs Polymorphism

Audience

This course is designed for developers, software and system architects and project managers involved with the development of Java™ applications.

Duration

2 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of Java is a prerequisite for this course.

3.6.3 Java8 New Features

Code

D-JAVA8

Overview

Java 8 introduces a number of revolutionary capabilities - many of them centered on lambda expressions and functional-style programming. These capabilities add powerful new programming techniques to the language, but also add complexity.

This concise course is focused on introducing the new capabilities and how to use them. It includes numerous code examples and programming labs that illustrate all of the new capabilities.

The course is hands on, and requires that students be comfortable with writing general Java code at an intermediate level, including the use of interfaces.

Description

Topics Include:

1. What is Java 8?
2. Java Lambda Expressions
    - What are lambda expressions?
    - Formal syntax for lambda expressions
    - Lambda expression simple syntax
    - Lambda expressions that return a value
    - Lambda expressions with multiple arguments
    - Lambda expressions and scope
    - Common usage scenarios
3. Method enhancements
    - Method references
    - Constructor references
    - Default methods
    - Static methods in interfaces
4. New Functional Interfaces
    - Function
    - Predicate
    - Consumer
    - Supplier
    - BinaryOperator
    - Additional new functional interfaces
5. Streams
    - Sequential vs. parallel streams

    - Immediate vs. terminal operations
    - Stream example
    - Lazy evaluation
    - A closer look at immediate and terminal operations
    - Primitive specialized streams
6. Enhanced Collections API
    - Iteration
    - New methods in List
    - New methods in Map
7. Enhanced concurrency API
    - ConcurrentHashMap
    - CompletableFuture
    - CountedCompleter
    - Adders and accumulators
8. Additional new features
    - Time
    - IO / NIO additions
    - Reflection and annotation changes
    - Nashorn JavaScript Engine

Audience

Any Java developer who needs to get up to speed with the latest features of Java platform / language

Duration

1 day

Format

Instructor Lead.

Prerequisites

Delegates should be comfortable with Java language, syntax and object oriented application development principles.

3.6.4 Effective Agile Java Development

Code

D-E504

Overview

This is an advanced and pragmatic workshop which will cover the latest agile development practices and tools that are used in Java™ Development.

Description

This course is an advanced pragmatic workshop that teaches latest agile development practices and tools. It provides practical experience across the full scope of agile development activities, including requirements gathering, acceptance test driven development (ATDD), behavior driven development (BDD), test driven development (TDD), agile architecture and design, clean coding practices, continuous integration and agile development teamwork and collaboration. Students will build a small application from the ground up using ATDD and TDD practices and getting exposure to innovative tools such as Maven, Jenkins/Hudson, Subversion, JUnit, Mock Testing, Selenium, Spock, JBehave.

Automated testing techniques are covered in detail in this workshop. Indeed, learning how to write more effective tests is an excellent way to write better designed, more maintainable and more reliable code. The course covers fundamental TDD and BDD practices for Java Developers. Continuous Integration, or CI, is a cornerstone of modern software development best practices.

Topics Include:

- **Apache Maven**
  Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
  Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:
    - Making the build process easy
    - Providing a uniform build system
    - Providing quality project information
    - Providing guidelines for best practices development
    - Allowing transparent migration to new features
- **Principles of Behavior Driven Development (BDD) and Agile Requirements:**

An introduction to the fundamental concepts and motivations behind Behavior-Driven Development and related techniques. BDD principles can be applied to requirements gathering and analysis activities. In this module, we learn how techniques such as Acceptance Test Driven Development, Executable Specifications and Feature Injection can significantly improve the quality, reliability and relevance of the application being built, and provide a much better traceability back to, and understanding of, the core business requirements. The course uses high-level BDD tools such as JBehave to build a working set of executable specifications from the ground up.

- **Agile Development Principles**

  Architecture is just as important in Agile projects as it is in conventional software projects. In this module we discuss the key principles of Agile development and design, including the role of architecture in Agile projects, when, how and by whom architecture is specified, implemented and validated.

- **Test Driven Development**

  Test Driven Development (TDD) is a key Agile development practice, and is arguably the single most effective way to improve code quality and reliability. In this section, we focus on core TDD and BDD practices at the coding level, and learn how to effectively apply BDD practices in Java both by using advanced JUnit features and testing approaches such as Mock testing. Unit testing vs. Integration testing is covered. Web testing frameworks such as Selenium is introduced. Specific BDD libraries such as Spock are covered as well as topics such as good test design and organization and testing databases.

- **Refactoring and Clean Coding**

  Refactoring is an essential part of maintaining high code quality and reducing maintenance costs. And a solid understanding of clean coding principles is vital to writing maintainable and understandable code. This course explores refactoring principles and patterns, and also discusses effective techniques to help make your code clean, readable and highly maintainable.

- **Team Programming and Collaboration**

  Team collaboration and communication techniques are discussed along with source code control and version management systems. Subversion and distributed systems such as Git are covered.

- **Continuous Integration and Delivery**

This module covers how Continuous Integration and Delivery practices can be used to enhance team communication and accelerate feedback. Jenkins, an Open Source Continuous Integration tool, is by far the most popular . This course will teach you how to build a powerful and robust CI infrastructure using Maven and Jenkins and automating the build process with Jenkins and provide a wealth of best practices and real-world tips

Audience

This course is in advanced series and designed for senior developers, software & system architects, project managers involved with the development of Java™ applications.

Duration

2 days

Format

Instructor Lead.

Prerequisites

A fundamental knowledge of Java is a prerequisite for this course.

3.6.5 Developing Reports with JasperReports and Eclipse

Code

D-JASPER

Overview

JasperReports is a popular open-source reporting engine whose main purpose is to help creating page oriented, ready to print documents in a simple and flexible manner. JasperReports is written in 100% Java and can be embedded in any Java application. JasperReports has the ability to deliver rich content in various formats such as PDF, HTML, XLS, CSV, XML files, or directly on the screen or printer. This comprehensive course enables software and report developers to develop reports using JasperReports and iReport through a combination of lecture, demos, and hands-on exercises, and includes the use of the Jaspersoft training environment.

Description

Topics

1. An Overview of JasperReports
    - What is Jasper Reports
    - Features of Jasper Reports
    - Flexible Report Layout
    - Subreports
    - Exporting Capabilities
2. Adding Reporting Capabilities to Java Applications
    - Downloading Jasper Reports
    - Installing Jaspersoft Studio
3. Creating Simple Report
    - Creating a JRXML Report Template
    - Creating a Binary Report Template
    - Generating Report
    - Displaying Report on a web browser
    - Elements of a JRXML Report Template

4. Creating Dynamic Database Reports
    - Generating Database Reports
    - Embedding SQL Queries into a Report Template
    - Modifying a Report Query via Report Parameters ...
    - Database Reporting via a Datasource
5. Working with Other Datasources
    - Empty Datasources
    - Map Datasources
    - Java Objects as Datasources
    - TableModels as Datasources
    - XML as Datasource
    - Custom Datasources
    - Writing a Custom JRDataSource Implementation
6. Report Layout and Design
    - Controlling Report-Wide Layout Properties
    - Styles
    - Setting Text Style for Individual Report Elements
    - Setting a Report's Background
    - Grouping Report Data
    - Report Expressions
    - Report Variables
    - Setting the Size and Position of a Report Element
    - Subreports
7. Adding Charts and Graphics to Reports
    - Adding Geometrical Shapes to a Report
    - Adding Lines to a Report
    - Adding Images to a Report
    - Adding Charts to a Report
8. Advanced JasperReports Features

- Report Localization
- Scriptlets
- Crosstabs
- Adding Hyperlinks and Anchors to Reports
- Bookmarks
- Handling Very Large Reports

9. Exporting to Other Formats
    - Exporting to PDF
    - Exporting to RTF
    - Exporting to Excel
    - Exporting to CSV
    - Export Reports to a Browser

Audience

Report developers, data analysts, data architects, system architects, and software developers

Duration

1 day

Format

Instructor Lead

Prerequisites

A fundamental knowledge of Java is a prerequisite for this course.

3.6.6 Building Secure Applications with Java and JavaEE

Code

D-Secure-java

Overview

This advanced cource is designed for building secure applications with Java and Java EE. Main topics are

- Software Development Life Cycle
- Java Application Security
- OWASP Java Best Practices

Description

Course include:

1. OWASP
2. Web Application Security Consortium
3. OpenSAMM
4. Enterprise Security Concepts
    a. Basic Vulnerability Terminology
    b. Enterprise Security APIs
    c. Software Development Life Cycle & Security Guideline
    d. Software Assurance Maturity Model
5. Security in Software Development Lifecyle
    a. Security Requirements
    b. Threat Modeling
    c. Secure Design Guidelines
    d. Secure Coding Guidelines
    e. Testing for web application security
    f. Secure administration and Security within Change Management
    g. Deployment WebApp Security Controls
    h. Secure Development Life Cycle
    i. Web Application Security Roles and Responsibilities
6. OWASP Top 10 Web Application Security & Vulnerabilities
    a. A1: Injection
    b. A2: Broken Authentication and Session Management

c. A3: Cross Site Scripting
d. A4: Insecure Direct Object Reference
e. A5: Security Misconfiguration
f. A6: Sensitive Data Exposure
g. A7: Missing Function Level Access Control
h. A8: Cross Site Request Forgery (CSRF)
i. A9: Using Known Vulnerable Components
j. A10: Unvalidated Redirects and Forwards
k. A9: Using Known Vulnerable Components
l. A10: Unvalidated Redirects and Forwards
m. A9: Using Known Vulnerable Components
n. A10: Unvalidated Redirects and Forwards

7. Testing for Vulnerabilities
    a. Web Application Security
    b. Software Security Assurance (SSA)
    c. Find Vulnerabilities
    d. Testing for application vulnerabilities
    e. Black Box vs. Gray Box
    f. Tools of the trade
    g. WebGoat
    h. The Zed Attack Proxy
    i. LAPSE+
8. Secure Development Practices
    a. Validating User Input
    b. Authentication
    c. Authorization
    d. Session Management
    e. Using Interpreters
    f. Crypto
    g. Catching Errors
    h. File System
    i. Configuration
    j. Web 2.0
9. Java Security Overview
    a. Information Security Principles
    b. Controls for Information Security
    c. Java EE Security Needs

d. Java EE Security Components
e. Securing EJBs and Web Applications

10. Enterprise Security API (ESAPI)
    a. ESAPI - Goals
    b. ESAPI to OWASP Top 10 Mapping
    c. ESAPI Maturity
    d. ESAPI Approach
11. SQL Injection Protection
    a. SQL Injection Attacks
    b. Finding SQL Injection Bugs
    c. Mitigating SQL Injection
    d. Methods to prevent SQL Injection

Audience

Java Developers

Duration

2 day

Format

Instructor Lead

Prerequisites

An advanced knowledge of Java is a prerequisite for this course.

3.6.7 Performance Monitoring and Tuning for Java Applications

Code

D-J-PERF

Overview

At some point of your career you will reach the situation when you will have to consider your enterprise application environment – server hardware, other applications running on your server and other servers running in your network.

You may for example want to know why disk operations were so quick on your development box, but became a major issue on the production box.

Don't leave performance to chance.

You will recognize and correct performance problems throughout the entire enterrpise application development lifecycle with this advanced course.

This course will provide you with the skills you'll need to quickly performance tune your Enterrpise Java applications.

The reasons driving this can range from a slow service, JVM crashes, hangs, deadlocks, frequent JVM pauses, sudden or persistent high CPU usage or even the dreaded OutOfMemoryError (OOME).

In this course we will be going through some of the open source tools that are available. Some of these tools come with the JVM itself, while some are third party tools

Description

Course include:

1. Java development performance tuning tips
2. The Java Virtual Machine
3. Analyzing and understanding the memory use of an application
4. JVM performance optimization tips
5. Garbage collection and application performance
6. Application Server Performance tips
7. Heap profiling
8. CPU profiling
9. Thread profiling
10. Profiling Tools
    - **Jmap :** prints shared object memory maps or heap memory details of a given process or core file or remote debug server
    - **VisualVM:** visual tool integrating several commandline JDK tools and lightweight profiling capabilities. Designed for both production and development time use, it further enhances the capability of monitoring and performance analysis for the Java SE platform
    - **Btrace:** safe, dynamic tracing tool for the Java platform. can be used to dynamically trace a running Java program
    - **EurekaJ:** profiler tool for Java applications. accept incoming statistics and provide a view to visualize the statistics in a consistent manner, parse the BTrace output, convert it to JSON and forward it to the EurekaJ Manager application's REST interface.
    - **Eclipse Memory Analyzer :**can help provide details of an application's memory use. The tool is useful for both tracking memory leaks and for periodically reviewing the state of your system

Audience

This course is designed for senior java developers, software and system architects, Operation managers

Duration

2 days

Format

Instructor Lead

Prerequisites

Advanced Java language knowledge and enterprise application development experience.

3.7 JAVAEE

3.7.1 Developing Enterprise Applications with JavaEE

Code

D-JEE-401-001

Overview

This course is designed for Java™ developers who need to learn how to develop Web based applications based on the Java™ Enterprise Edition (JavaEE). This is a best-practices course that will guide the students through building a complete end-to-end web application.

At the end of this course you will understand the best practices in building Internet Applications using JavaEE as well as the core JavaEE Technologies including Servlets, JSP and JNDI. You will understand distributed Web-based JavaEE architectures and Advanced Web Application topics including graphics, security, internationalization and multi-access.

This course takes you through the basics of developing a Java EE enterprise application and demonstrates some of the EJB 3 technology features that were introduced as part of the Java EE specification.

This course teaches students how to build Web Services and Web Service clients using Java technologies. The class includes a high-speed introduction to XML syntax, namespaces, XML Schema, SOAP, and WSDL before exploring Web Service client or server-side development in Java APIs and tools. Specifically, this class focuses on JAX-WS and JAX-RS web service and client development.

Description

JEE Web Tier Technologies

1. Internet Technologies, Overview
2. Distributed Web Architectures and JavaEE
3. HTML Concepts
4. Use of XML in Web Applications
5. JavaEE Web Application Organization and Assembly (WAR)
6. Servlets/Filters
7. Session Management - Cookies
8. Java™ Server Pages (JSP)
9. TagLibs, TagFiles, JSP-EL, JSTL
10. Web Application Design Patterns and Frameworks(MVC)
11. Java Server Faces 2
    - 11.1. JSF Architecture
    - 11.2. JSF Quick Start
    - 11.3. Facelets
    - 11.4. Managed Beans
    - 11.5. Unified Expression Language
    - 11.6. Message Bundles-Internalization
    - 11.7. Navigation
    - 11.8. Events, Actions and Listeners
    - 11.9. JSF Tables and Table Models
    - 11.10. Conversion & Validation
    - 11.11. Custom Tags and Components
    - 11.12. Ajax and JSF
    - 11.13. JSF Implementations and Component Libraries
12. JEE Security

XML Technologies

- xml, xsd
- jaxb
- xslt
- xquery
- xpath

Java And Web Services

1) Web Services Technologies
   a) Web Service Styles
   b) REST / SOAP
   c) Enterprise Web Services - JAX-WS
2) Java Web Services and JAX-WS
   a) Java First Web Services
   b) Web Service Clients
   c) JAX-WS Annotations
   d) WSDL – Web Services Definition Language
3) Rest style services

Java Persistence API (JPA)

- Object-Relational Mapping - ORM
- Java Persistence API – JPA
- Configuration and Project Setup
- Simple Mapping
- Relational Mapping (OneToOne, ManyToOne, OneToMany, ManyToMany)
- Query, JPQL and Criteria

Context and Dependency Injection (CDI)

- DI and CDI
- Injection
- Beans and Bean Scopes
- Injecting Objects by Using Producer Methods
- Qualifiers
- Alternatives
- Stereotypes
- Java EE Resources
- Events
- Interceptors

EJB Technologies

1) What is a EJB
2) Enterprise Java Server (EJS)
3) EJB3
4) EJB components
5) Types of EJB
    a) Stateless session bean - example scenario
    b) Stateful session bean - example scenario
    c) JPA Entity - example scenario
    d) Message driven beans - example scenario
6) EJB containers and container contracts
7) EJB Context
8) Annotations and Deployment descriptors

9) EJB application packaging and deployment
10) Exception and exception handling
11) EJB Security
    a) Java Authentication and Authorization Service (JAAS)
    b) JAAS used in EJB
    c) Container managed vs. Bean managed security
    d) Role based security
    e) Method permissions explained with example
12) EJB Transactions
13) EJB Timers
14) Interceptors in EJB
15) Message Driven Beans
    a) JMS-Messaging Domain
    b) Developing MDB
16) Transactions, Security

Audience

Suitable for developers, software architects, system architects and project managers involved with the development of Internet applications.

Duration

5 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of Java is a prerequisite for this course.

3.7.2 Developing EJB3 Components and Services

Code

D-JEE-401-002

Overview

This is a 3-day course consisting of lectures and demos combined with hands-on exercises using computer-based labs. Suitable for developers, software architects, system architects and project managers involved with the development of Internet applications. This advanced course is designed for Java developers who need to learn how to develop components based on Enterprise Java Beans standards (EJB3) and Java-WS using Java Enterprise Edition (JavaEE). This course will guide the students through building an enterprise application using distributed component-based JavaEE architecture. The technologies that are covered in this course are EJB3, JAX-WS along with related JavaEE technologies- including JAAS, JNDI, JDBC, JMS, JTS/JTA.

Course Goals:

- Understand the essential concepts of JavaEE Components and EJB3
- Understand the essential concepts of Web Services and JAX-WS
- Apply these concepts to the development of highly modular client/server systems
- Learn the best practices for developing Service based distributed Enterprise Applications

Description

EJB Technologies

1) What is a EJB
2) Enterprise Java Server (EJS)
3) EJB3
4) SOA and JEE
5) EJB components
6) Types of EJB
    a) Stateless session bean - example scenario
    b) Stateful session bean - example scenario
    c) JPA Entity - example scenario

d) Message driven beans - example scenario

7) EJB containers and container contracts
8) EJB Context
9) Annotations and Deployment descriptors
10) Clustering
11) EJB application packaging and deployment
12) Exception and exception handling
13) EJB Security
    a) Java Authentication and Authorization Service (JAAS)
    b) JAAS used in EJB
    c) Container managed vs. Bean managed security
    d) Role based security
    e) Method permissions explained with example
14) EJB Transactions
15) EJB Timers
16) Interceptors in EJB
17) Message Driven Beans
18) JMS-Messaging Domain
19) Developing MDB
20) Transactions, Security, Clustering

Java WebService Technologies

1) Web Services Technologies
    a) Web Service Styles
    b) REST / SOAP
    c) Enterprise Web Services - JAX-WS
2) Java Web Services and JAX-WS
    a) Java First Web Services
    b) Web Service Clients

c) JAX-WS Annotations

d) WSDL – Web Services Definition Language

e) Contract First Web Services

Audience

Java developers who are (or will be) involved in enterprise Java development. This course is intended for experienced Java programmers who are familiar with the advanced aspects of Java like serialization, sockets, RMI, and JDBC.

Previous exposure to Java EE (including JMS), web development, and a working understanding of database fundamentals and SQL is also strongly suggested.

EJB3 training course will build upon these prerequisites to gain the specific skills necessary to develop, deploy, and run distributed applications using Enterprise JavaBeans (EJB3).

Duration

3 days

Format

Instructor Lead

Prerequisites

Solid Java programming skills and understanding of OO Java and Java-5.0 language features is essential. Experience with developing Java web applications is very helpful for this course, but not strictly required. Some knowledge of XML will be useful for writing the occasional deployment descriptor, but is not required.

3.7.3 Effective Enterprise Applications with JavaEE

Code

D-E502

Overview

This is an advanced Java™ Programming training course that teaches Java developers how the use the latest JavaEE technologies and the best enterprise development practices and design patterns skills effectively.

Description

This course is designed for Java™ developers who need to learn how to develop Enterprise applications based on the latest Java™ Enterprise Edition (JavaEE) specification. This is a best-practices course that will use the latest JavaEE technologies and the best enterprise development practices and design patterns skills through building a complete end-to-end enterprise application.

At the end of this course you will understand the best practices in building Internet Applications using JavaEE as well as the core JavaEE Technologies including Web, Service Components and Persistence layers. You will understand distributed Web-based JavaEE architectures and Advanced Web Application topics including graphics, security, internationalization and multi-access.

This course takes you through the advanced details of developing a Java EE enterprise application and demonstrates some of the latest Web, Services and Components, EJB, and JPA technology features that were introduced as part of the Java EE specification.

The class includes standards such as JSF, JPA, EJB, Web Services, REST, XML, XSD APIs and tools.

This course also includes Apache Maven. Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Audience

This course is in advanced series and designed for senior developers, software & system architects, project managers involved with the development of Java™ applications.

Duration

2 days

Format

Instructor Lead.

Prerequisites

A fundamental knowledge of Java is a prerequisite for this course.

3.8 MOBILE

3.8.1 Developing Android Applications

Code

D-ANDROID-391-001

Overview

Android is a mobile applications platform that has the backing of Open Handset Alliance, which includes Google, Verizon, and other companies. Android platform allows developers to write managed code in the Java language, controlling the device via Java libraries. Android is under the Apache License, a free software and open source license. Android is becoming a major platform for mobile applications due to its rich features and portability: now you can build apps that will work across several devices. The course also covers advanced topics such as creating custom widgets, create animations, working with the camera, using sensors, create and use advanced content providers and much more. The course covers testing and deployment of Android applications.

In this course you will build applications for the Android platform, including:

- Build a working mobile application
- Use Android controls, forms, and dialogs
- Create a local database persist appplication data
- Use advanced device features such as motion and location-awareness
- Integrate with other applications such as maps

Description

1. Introduction
   - What is Android?
   - History
   - Android Overview
   - Application Frameworks
   - Android Application Architecture
   - Dalvik VM vs. JVM
2. Developing Mobile Applications
   - Why Mobile?
   - Mobile Development Approaches
   - Mobile Sites

- Native Mobile Applications
- Hybrid Mobile Applications

3. Quick Start-First Project
   - Development Overview
   - Installing
     1. Android SDK & Tools
     2. IDE
     3. Developing an Application
        - Project setup
        - Running
          - On the emulator and device
          - Emulator configuration
4. Understanding the Internal Structure
   - Short history and overview
   - Framework overview
   - Android for Mobile, Android for Machines, Android as an OS
5. Applications & Activities
   - Applications
     1. Lifecycle
     2. Activities
        - Lifecycle
        - Reacting to Configuration Changes
        - Application Manifest
6. User Interfaces & Layouts
   - Introduction to UI
   - Layouts
   - Android Screen Size Diversity
     - Different types of layouts
     - LinearLayout
     - RelativeLayout
     - TableLayout
     - FrameLayout
     - GridLayout
     - Nesting layouts to create complex views
     - Layout Performance
     - Merge and Include with reusable layouts
7. Views & Basic UI Controls

- Overview Of UI Elements
- Common Android Widgets
    - TextView
    - EditText
    - Button
    - RadioButton
    - RadioGroup
    - CheckBox
    - ToggleButton
    - ImageView
    - ImageButton
    - Progress Bar
    - Spinner
    - DatePickerDialog
    - ListView
    - Adapters
    - Toasts
    - Interacting with the widgets
    - References UI Elements in Java
    - Event Listener

8. Fragments
    - What is a Fragment?
    - Fragment Lifecycle
    - Steps for creating fragment
    - Fragment Manager
9. Custom Views & Basic 2D Graphics
    - Custom Views
    - Modify Existing Views
    - Customize Views
    - Compound Views
10. Animations
    - Layout Animations
    - Animations for Activity Transitions
    - Animating Objects
    - AnimationListener
    - Animating the transitions

11. 2D Graphics
    - 2D Graphics using
        - ImageView
        - Canvas
12. 12 Dialogs
    - DialogBox Forms
        - AlertDialog
        - Toast
        - Custom Dialogs
13. Connectivity & Networking
    - Android Networking Support Classes
    - Networking Permissions
    - Sockets
    - URL Connections
    - HttpClient
    - JSON
    - Parsing JSON
    - Parsing XML Data
14. Intents
    - What are Intents?
    - What are Intents used for?
    - Making a call
    - Web browser Intent
    - common Intents
    - Intents and Application Components
    - Intent properties
    - Passing Data
    - Returning Data
    - Intents and Bundles
    - Intent Filters
15. BroadcastRecivers
    - Broadcast Receiver
    - Registering
        - Dynamic Registration
        - Static Registration
        - Event Broadcast

        - Normal Broadcast
        - Ordered Broadcast
        - Sticky Broadcast
        - Event Delivery
16. Data Persistence: Files, State Preferences
    - Client Data Persistence
        - Files
        - SharedPreferences
        - SQLite database
17. Services
    - Service
    - Service Class
    - Example Services
        - Logging Service
        - Music Player Service
        - ID Service
        - Implementing a Service
        - AIDL Syntax
        - Implement Remote Methods
        - Implement Service Methods
18. Content Providers
    - ContentProvider
    - ContentResolver
    - Querying
    - Cursor
    - Deleting Data
    - Inserting Data
    - Creating a Content Provider
19. SQLite
    - What is SQLite?
    - SQLite and Android
    - Opening a Database
    - Using a Database
        - Inserting
        - Deleting
        - Querying

- Examining the Database Remotely

20. Android Threads and Asynchronous Tasks
    - Android Threading
    - Thread Methods
    - Thread Example
    - Implications
    - AsyncTask
    - Handler
21. Notifications & Alerts
    - Notifications
    - Toast Notifications
    - Dialogs
    - Notification Manager
    - Creating Notifications
    - Send Notification
    - Alarm Manager
22. Multimedia
    - MultiMedia Features
        - AudioManager
        - SoundPool
        - RingtoneManager & Ringtone
        - MediaPlayer
        - MediaRecorder
        - Camera
23. TouchGesture
    - MotionEvents
    - Touch Handling
    - Gestures
24. Google Cloud Messaging 25 Sensors
    - Sensors
    - SensorManager
25. Location and Maps
    - Location Services
    - Maps
    - Overlay
    - GeoPoint

26. Account Manager
    - Account Manager API

Audience

Developers who aim to develop Android applications

Duration

4.5 Days

Format

Instructor Lead

Prerequisites

Basic Java programming skills, or equivalent OOP language experience

3.8.2 Developing iPhone/iPad Applications with Objective-C

Code

D-IPHONE-361-001

Overview

iPhone and iPad are popular platforms revolutionizing the mobile platforms. While these platform offer standart buildin gps, connectivity, multitouch controls and accelerometer, they also implement a unique and yet different programing model. To build any application and publish them to famous App Store, developers need to code in Objective-C. Objective-C also comes with a full featured devolopment IDE called XCode. Besides the coding facilities in XCode, another tool is also integrated to design and build user interfaces, the Interface Builder. The course covers topics from getting familiar with Objective-C syntax, controlling phone resources, memory management, using APIs and built in patterns, designing UIs with Interface Builder, building and debugging your application, using the simulator, using certificates to sign and publish your application. The course also covers tips on following Apple's HIG (Human Interface Guide) for getting approval to get published in iPhone App Store.

In this course you will build applications for the iPhone/iPad platform, including:

- Using XCode to build applications
- Learning Objective-C syntax
- Build a working mobile application
- Building User Interfaces with Interface Builder
- Using IB components

- Create a local database persist appplication data
- Use advanced device features such as camera, motion and location-awareness

Description

Course Topics

1. iOS Technologies
    - Overview
    - Short history
    - SDK
2. Architecture layer
    - Layers
    - Overview of frameworks
3. Developer Tools
    - Xcode Environment
    - Xcode features and tools
    - Instruments
    - Developer Library
4. How to start developing an iOS app?
    - Introduction
    - Setup development environment
    - Quick Start: First iOS app
    - Running and Debugging an application
    - Using Simulator
5. Write Objective-C Code
    - Overview
    - Syntax
    - Class and interface structure
    - Sending messages to objects
    - Memory management
    - Built-in patterns
    - Foundation Framework
6. Building User Interfaces
    - Storyboards

- Bind code and UI
- ViewController Lifecycle
- Multiple MVCs

7. Navigation
8. ScrollView
9. TableView
10. Handling Events
    - Using delegates
    - Taps, Touches, Gestures
11. CollectionViews
12. Protocols
13. AutoLayout
14. Handling Different Screen Sizes
    - Building apps for different hardware resources: iPhone vs. iPad
15. Device APIs
    - Using Camera and Photo library
    - Core Location
    - Using Accelerometer
16. Connectivity
    - Working with services
    - Best practices
    - Multithreading
17. Graphic Libraries
    - Basic graphics using Quartz
18. Internationalization and Localization
    - Preparing Nib files for Localization
19. Automated Testing
    - Why automates tests?
    - Unit testing
    - Instrumentation testing
20. Publishing Applications

- How to stay HIG compliant
- Getting and using developer certificates
- Signing and sending your application for approval
- First refusal: How to analyze incomplete parts
- Users always right: Understanding user comments

Audience

Developers who aim to develop Iphone applications.

Duration

4.5 days

Format

Instructor Lead

Prerequisites

Experience in the following areas is required: Some prior programming experience in a procedural or object-oriented language.

Mac computer should be provided by participants

3.8.3 Developing Hybrid Mobile Applications

Code

D-MBHB-001

Overview

Hybrid development combines the best (or worst) of both the native and HTML5 worlds. Hybrid apps primarily built using HTML5 and JavaScript, that is then wrapped inside a thin native container that provides access to native platform features. PhoneGap is an example of the most popular container for creating hybrid mobile apps and enabling native apis to be used from javascript.

For the most part, hybrid apps provide the best of both worlds. Existing web developers that have become gurus at optimizing JavaScript, pushing CSS to create beautiful layouts, and writing compliant HTML code that works on any platform can now create sophisticated mobile applications that don't sacrifice the cool native capabilities. In certain circumstances, native developers can write plugins for tasks like image processing. Apps can package HTML and JavaScript code inside the mobile application binary, in a manner similar to the structure of a native application. In this scenario you use REST APIs to move data back and forth between the device and the cloud

Description

Topics Include

- Modelling Hybrid Mobile UI
- Structure of Hybrid apps
- Overview of Mobile Frameworks
- HTML5, CSS3 and Javascript
- Web Design Strategies
- PhoneGap/Apache Cordova
- JqueryMobile
- BackboneJS
- Development Setup
- Best Mobile Design Practices
- Useful Metatags
- Working with Local Data
- Working with Remote Data
- Working With Media
- Integration with Device APIs
- Debugging

- Offline Apps
- Single Sourcing
- Simple Security for Hybrid Mobile Applications

Audience

This course is in advanced series and designed for senior developers, software & system architects

Duration

3 days

Format

Instructor Lead

Prerequisites

Advanced experiement on Java Language, JavaScript and JQuery.

3.9 OSGI

3.9.1 Developing OSGi Applications with Equinox

Code

D-OSGI-321-001

Overview

Based on the book, OSGi and Equinox: Creating Highly Modular Java Systems by Jeff McAffer,et. al., this is a three-day course that enables you to leverage the capabilities of Equinox and OSGi for developing and deploying enterprise applications. The course covers OSGi fundamentals and advanced topics as well as specific features of Eclipse Equinox, everything you need to develop highly modular Java applications. Along the way they provide deep insights and context to help you to start your own development efforts and keep on track.

Course participants incrementally develop a comprehensive application involving clients, servers and embedded devices, HTTP, servlets, JPA persistence and other web technologies in a series of hands-on focused sessions. You will gain a broad understanding of Eclipse, Equinox and OSGi. This intensive course covers fundamentals of OSGi as well as intermediate topics and specific features of Eclipse Equinox. You will apply the theoretical knowledge in several labs which will give you the practical experience necessary for your projects. All attendees will receive a copy of the OSGi book.

Equinox and OSGi:

Equinox is the core of the Eclipse platform and a major contributor to the success of Eclipse on the clientside(RCP). Bundles (plug-ins) and Extension Points based on the OSGi runtime are solid architectural building blocks. However, Equinox is not limited to the client-side. It can be applied in a general way to build any kind of application, especially servers. Equinox-based servers use the OSGi runtime which provides a service abstraction. OSGi is a dynamic environment where bundles can be installed, started, stopped and uninstalled at runtime. The Equinox vision is to model a community and a repertoire of bundles specifically built for servers, similar to the Eclipse SDK where bundles cover a vast array of functions and purposes for the client. Imagine extending your server with additional functionality by installing some extra bundles, for example a log analyzer for your HTTP service.

Description

The course includes:

1. **OSGi Introduction**

- What is a bundle
- Modularity
- Bundle lifecycle
- How to develop, run and debug bundles

2. **Services**
   - Inter-bundle collaboration
   - Defining services
   - Discovering services
3. **Dynamic Systems**
   - Dynamic awareness
   - Trackers, listeners, activators
   - Best practices for being dynamic
4. **Declarative Services**
   - Compared to traditional service techniques
   - Concepts: components, immediate, cardinality, optional, ...
   - POJO techniques
   - Best practices for naming, markup
   - PDE Tooling
5. **Select OSGi Standard Services**
   - HTTP
   - ConfigAdmin
   - Log
   - Preferences
6. **Classloading**
   - Classloading in OSGi and Equinox
   - Dependencies and classloading
   - Buddy Classloading policies
   - Context classloader integration

7. **Server Side Equinox and OSGi**
    - Solo: Jetty in Equinox
    - Embedded: Equinox embedded in app servers
    - Servlet bridge
    - Servlets, JSPs, AJAX content
    - OSGi-enabled WAR development and deployment
    - Security contexts and JAAS integration
8. Tooling
    - Target platforms and cross development
    - Using third-party libraries
    - Dependency management
9. **Provisioning**
    - Introduction to p2
    - Adding dynamic provisioning to applications
    - Provisioning servers
    - Extending p2

Audience

This course is for:

- Software developers looking to leverage Equinox and OSGi
- Eclipse plug-in wanting a deeper understanding of how Eclipse works
- System architects interested in leading edge Java modularity
- OSGi developers wanting to round out their knowledge and learn about Equinox extensions to OSGi and the Eclipse tooling suite

Duration

3 days

Format

Instructor Lead

Three-day instructor-led class with approximately 50% hands-on labs.

Prerequisites

- Excellent knowledge of Java, JavaEE and object-oriented programming
- Experience using the Eclipse IDE is desirable

3.10 Project Management

3.10.1 Agile Methodology & Scrum& Agile Testing

Code

P-AG-TST-001

Overview

Agile Methodologies

- History of Agile
- Agile Methodologies
    - Scrum
    - Kanban
    - Extreme Programming
- Manifesto for Agile Software Development
- Scrum
    - How it works?
    - What are the roles?
        - Team
        - Product Owner
        - Scrum Master
    - What are scrum ceremonies?
        - Release Planning
        - Sprint Planning
        - Sprint Review
        - Sprint Retrospective
        - Daily Scrum

- Artifacts
  - Product Backlog
    - Creating
    - Managing
    - Sprint Backlog
    - Burn Down Chart
    - Impediment List
- Scrum from the view of Scrum Master
- Scrum from the view of Product Owner
- Scrum from the view of Team
- Estimating & Planning
- Measuring & Reporting

Test Methodologies

The Role & Characteristics of a Tester

Test Methods

- Static - Dynamic Methods
- Black Box - White Box
- Visual Testing

Testing Levels

- Unit testing
- Integration testing
- System testing
- Acceptance testing

Test Objectives

- Compatibility Tests
- Smoke Tests
- Regression Tests
- Alpha Tests

- Beta Tests
- Functional Tests
- Non Functional Tests
- Performance Tests
- Usability Tests
- Security Tests

Agile Methodologies & Testing

- What is your "Done" Definition?
- Testing at Agile
- Difference of Testing at non Agile Team & Agile Team
- Test Effort Estimation
- Different Perspectives for Agile Testers
  - Continuous Integration
  - Version Management
  - Pairing
  - Acceptance Criteria
  - Regression Testing
  - Defect Management
  - Test Driven Development
  - Test Automation
  - Agile Performance Testing
- Behaviors that might cause Agile Testing To Fail
- Improvement Process

Audience

Quality & Test Specialists, Developers, Project Managers, Product Owners

Duration

3 days

Format

Intructor Lead

Prerequisites

3.10.2 Agile Methodology & Scrum

Code

P-AG-001

Overview

Agile Methodologies

- History of Agile
- Agile Methodologies
    - Scrum
    - Kanban
    - Extreme Programming
- Manifesto for Agile Software Development
- Scrum
    - How it works?
    - What are the roles?
        - Team
        - Product Owner
        - Scrum Master
    - What are scrum ceremonies?
        - Release Planning
        - Sprint Planning
        - Sprint Review
        - Sprint Retrospective
        - Daily Scrum
    - Artifacts
        - Product Backlog
            - Creating
            - Managing
            - Sprint Backlog

        - Burn Down Chart
        - Impediment List
- Scrum from the view of Scrum Master
- Scrum from the view of Product Owner
- Scrum from the view of Team
- Estimating & Planning
- Measuring & Reporting

Audience

Duration

2 days

Format

Instructor Lead

Prerequisites

3.11 SOA

3.11.1 Effective Service and API Development with Java

Code

D-E503

Overview

This is an advanced and pragmatic workshop covering Service and API design patterns and development practices.

Description

There are many API sets we use in any development cycle coming from different frameworks or libraries. API quality depends on designer knowledge and design capability . A Simple, maintainable, functional and consistent API does not happen by accident. Service Oriented Architectures (SOA) have been around for years. Web services is the latest and greatest implementation of SOA. What does SOA really mean and how is it successfully implemented using Web services? Eteration's class on Effective Services and API development is aimed to help developers, architects, implementation managers to effectively use the core concepts and building blocks of services and how to design highly interoperable and scalable enterprise systems using the latest in services technologies such as and technologies such as XML, JSON, WS-*, REST and OAuth.

We focus on design practices that facilitate the development of better services and APIs, using SOA or different services architectures: SOA, REST, Modern API technologies. APIs such as those used by twitter, facebook and foursquare are discussed. Interoperability, enterprise level security and other advanced topics such as OAuth are also covered in the course.

API Design

- General Design Tips and Best Practices

- Modular Architectures
- Use of Interfaces
- Runtime aspects
- Commitment and contracts

RESTfull APIs

- Design rules for addressing resources with URIs
- Design principles to HTTP's request methods and response status codes
- Guidelines for conveying metadata through HTTP headers and media types
- Design tips to address the needs of client programs
- Understand why REST APIs should be designed and configured, not coded
- JAXRS and Java REST

Web Services

- Service Oriented Architecture
- SOAP, WSDL, UDDI
- Standard Java™ XML
- JAXWS and Web Services APIs

Design Patterns and Best Practices

- General Design Tips and Best Practices
- WSDL First
- RPC vs. Message Driven Thinking
- Validation
- Designing Good Schemata
- Namespace Guidelines and Design Tips
- Patterns and Anti-patterns
- Versioning

Audience

This course is in advanced series and designed for senior developers, software & system architects, project managers involved with the development of Java™ applications.

Duration

2 days

Format

Instructor Lead.

Prerequisites

A fundamental knowledge of Java is a prerequisite for this course.

3.11.2 Developing Service-Oriented Applications with Java

Code

D-WSOA-460-001

Overview

This advanced course is designed for Java developers who need to learn how to develop Service Oriented Applications using Java and Web services.

What you will Learn

You will become familiar with service-oriented architectures and supporting technologies. At the end of this course you will be able to implement Web services using Java, Process and transform XML.

Course Goals

- Understand SOA and Service-Orientation
- Learn how to build SOA with Java and Web Services
    - XML Technologies
    - Web Services Technologies
    - Developing components with Java EE

Description

Concepts

- What is a Service
- What is a Web Service
- What is Service-Orientation
- What is Service-Oriented Architecture
    - Principles
    - IT Challenges and SOA Maturity Models
    - Components and Services
    - Service Orchestration and Choreography
    - Enterprise Service Bus

XML Technologies

- XML Technologies

- XML
- Web Standarts
- Grammars for XML Documents - DTD & XSD - Document Type Definitions and Schema

Definitions

- XSLT - Extensible Style Sheet Language Transformations
- XQuery – XML Query Language
- XPath - XML Path Language
- Integration of XML into Applications
  - Java and XML
    - Java XML Binding and Parsing- Java Architecture for XML Binding JAXB
    - Frameworks for XML processing
  - XML Data Representation & Validation

Web Services Technologies

- Web Services Technologies
  - Web Service Styles
    - REST / SOAP
    - REST and JAX-RS
  - Web Services and SOA
  - WSDL – Web Services Definition Language
    - Java First Web Services
    - Web Service Clients
    - Annotations
    - Contract First Web Services
  - SOAP – Simple Object Access Protocol
  - UDDI – Universal Description, Discovery and Integration
  - Java and WS
    - JAX-WS

        - JAX-RS
        - Apache Axis and Others
    - Advanced
        - Dynamic Invocation
        - Asynchronous Web Services
        - WS-Security and WS-Policy
        - WS-Attachments & SOAP with Attachments
        - WS-Reliable Messaging
    - Integration of Web Services into Applications

Audience

This advanced course is designed for Java developers who need to learn how to develop service-oriented applications using Java and Web services.

Duration

3 days

Format

Instructor Lead

Prerequisites

This advanced course is designed for Java developers who need to learn how to develop service-oriented applications using Java and Web services.

3.12 Software Project Management

3.12.1 Apache Maven

Code

D-MVN

Overview

This course covers all of the basic fundamentals of Maven. It covers dependencies, plugins, repositories, IDE integrations, and all the basic commands of Maven.

Description

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Providing guidelines for best practices development
- Allowing transparent migration to new features

Topics Include:

1. Introduction to Maven
    a. What is Maven?
    b. How Maven works?
    c. What does Maven do?
    d. Maven plugin architecture
    e. Conceptual Model of a project
    f. İnstalling Maven
    g. Eclipse and Maven
2. Maven Quickstart
    a. Create a simple Maven project
    b. Archetypes
    c. Project structure
    d. Super pom
    e. Building Maven Project
3. Maven Core Concepts

    a. Maven plugins and Goals
    b. Maven Lifecycle
    c. Maven Coordinates
    d. Dependency Management
    e. Maven Repositories
    f. Site Generation and Reports
4. Customizing a Maven Project
    a. Customizing compiler
    b. Customizing project information
    c. Managing dependencies and scopes
    d. Managing classpath resources
    e. Customizing tests
    f. Integration testing
5. Multi-module Projects
    a. Module Layout Strategies
    b. Parent and Submodule Configuration
    c. Web Projects
    d. Building Multi-Module Projects
6. Dependency Management in Depth
    a. Transitive Dependencies
    b. Limiting Dependencies
        i. Dependency Mediation
        ii. Dependency Scopes
        iii. Dependency Management
        iv. Excluded Dependencies
        v. Optional Dependencies
    c. Importing Dependencies
    d. Bill Of Materials (BOM)
7. Release Management
    a. Releasing Software
    b. Maven Release Plugin
    c. Distribution Repositories
    d. Deploy to Nexus Instance
    e. Introduction to Nexus
8. Maven Best Practices

Audience

Ideal for programmers who want to use Maven on their projects. This class is also appropriate for the existing Maven user who is interested in developing a greater understanding of the Maven fundamentals.

Duration

1 day

Format

Instructor Lead.

Prerequisites

prior experience of JAVA and eclipse

3.12.2 Gradle

Code

D-GRDL

Overview

This course is for build masters and developers who are authoring their builds. Participants will learn how to use the Gradle build system to substantially increase their productivity.

Topics Include:

- Introduction
- Groovy for Gradle
- Installing Gradle
  - Environment variables
  - Testing your installation
- Quick Tour of Gradle
  - Creating build scripts
  - Declaring dependencies
  - Accessing repositories
  - Using plugins
  - Configuring the directed acyclic graph
- Building Java projects
  - Standard project structure
  - The Java plugin
  - Running tests
- Building Groovy projects
  - The Groovy project structure
  - Working with both Groovy and Java
  - Executing tests with both JUnit and Spock
- Defining Tasks
  - Declaring tasks
  - Defining project properties

    - Setting dependencies
    - Using doFirst and doLast
    - Using the built-in task types
- The Gradle Daemon
    - Usage and troubleshooting
    - Configuring the daemon
- Web projects
    - Standard web layout
    - The war and jetty plugins
    - Customizing web projects
- IDE Integration
    - Eclipse projects
- The Gradle wrapper
    - Specifying versions
    - Generating the scripts
- Multi-project builds
    - Using settings.xml
    - Consolidating configuration properties
    - Making one subproject depend on another

Audience

Ideal for programmers who want to use Gradle on their projects. This class is also appropriate for the existing Gradle user who is interested in developing a greater understanding of the Gradle fundamentals

Duration

2 days

Format

Instructor Lead.

Prerequisites

prior experience of JAVA and eclipse

3.12.3 Modern SCM Practices and GIT

Code

D-SCM

Overview

A source control management system (SCM) is software that provides coordination and services between members of a software development team. At the most basic level, it provides file management and version control so that team members don't write over each other's changes, and only the newest versions of files are identified for use in the workspace.

SCMs also give developers the ability to work concurrently on files (in branches that may or may not converge), to merge changes with other developers' changes, to track and audit changes that were requested and made, to track bug-fix status and to perform releases. In some cases, SCMs may include other components to assist in managing a software process throughout the entire lifecycle.

Git is the new fast-rising star of version control systems. Initially developed by Linux kernel creator Linus Torvalds. Git offers a much different type of version control in that it's a distributed version control system. With a distributed version control system, there isn't one centralized code base to pull the code from. Different branches hold different parts of the code. Other version control systems, such as SVN and CVS, use centralized version control, meaning that only one master copy of the software is used.

Topics Covered:

1. Introduction to SCM systems
    1.1. Overview of Software Version Control

1.2. Version Control Concepts
1.3. Comparing centralized vs. distributed systems
1.4. Proprietary/Open-Source SCM Tools

2. GIT
   2.1. What is Git?
      - Understanding version control
      - The history of Git
      - About Distributed Version control
   2.2. Git Concepts and Architecture
   2.3. Getting Started
      - Setting up a repository
      - Initializing a repository
      - Saving changes
      - Inspecting a repository
      - Viewing old commits
   2.4. Making Changes to Files
      - Adding files
      - Editing files
      - Viewing changes with diff
      - Viewing only staged changes
      - Deleting files
      - Moving and renaming files
   2.5. Undoing Changes
      - Undoing working directory changes
      - Unstaging files
      - Amending commits
      - Retrieving old versions
      - Reverting a commit
      - Using reset to undo commits
      - Removing untracked files
   2.6. Ignoring Files
   2.7. Navigating the Commit Tree
   2.8. Branching & Merging
      - Branching can Change Your Life
      - Working With Branches
      - Publishing a Local Branch
      - Deleting Branches
      - Saving Changes Temporarily
      - Checking Out a Local Branch
      - Merging Changes
      - Branching Workflows
      - Cherry-Picking
   2.9. Stashing Changes
   2.10. Sharing Work via Remote Repositories
      - Introduction to Remote Repositories

- Connecting a Remote Repository
- Inspecting Remote Data
- Integrating Remote Changes

2.11. Advanced Topics

- Undoing Things
- Inspecting Changes with Diffs
- Dealing With Merge Conflicts
- Rebase as an Alternative to Merge
- Submodules
- Workflows with git-flow
- Authentication with SSH Public Keys

2.12. Tools & Services

- Desktop GUIs
- Diff & Merge Tools
- Code Hosting Services

2.13. Migrate GIT from SVN

3. 3rd Party Tools
   3.1. Code Review Tools: Gerrit
   3.2. Code Quality Management Tools – SonarSource
   3.3. Code Analyzer Tools: FindBugs, Jlint etc...
4. Best Practices of Good Source Code Management
5. Selecting a SCM Tool that is rigth for you

Audience

Developers, architects

Duration

1 day

Format

Instructor Lead

Prerequisites

Any programming language experience.

3.13 Spring

3.13.1 Developing Enterprise Applications using Spring Framework and JPA

Code

D-SPRING-381-001

Overview

This advanced course is designed for Java developers who need to learn how to use the Spring Framework to create well-designed, testable business applications in an agile manner. During this course, you will learn Spring's features and how to use them. You will also become familiar with the fundamental architectural issues you will need to be aware of when developing with the Spring framework. It is important to know how to use certain parts of the Spring framework, but even more important to be able to decide when to use them!

Topics Covered:

- Work with the Spring Inversion of Control (IoC) Container
- Effectively use JDBC , Hibernate and JPA for data access
- Use JUnit, Spring, stubs and mocking frameworks to effectively implement automated unit and integration tests
- Take advantage of Aspect-Oriented Programming (AOP) to keep code clean and maintainable

- JMS, JMX, Spring Batch
- And much more...

Description

Introduction

- Spring IO Platform Overview
- Spring Projects
- Basic environment

Spring Architecture

- Spring Framework definition
- Spring Framework design principals
- Sprint interfaces

Spring setup

- Setting classpath and jar files
- Setting configuration

Spring Core Concepts

- Spring quick start
- Writing bean definitions
- Configuring objects
- Creating an application context

Design Patterns

- Inversion of Control
- Dependency Injection

Bean Life-Cycle

- Application context life-cycle
- Initialization of beans
- Using beans
- Destruction
- Other considerations

Spring Application Configuration

- Bean definition inheritance
- Inner beans

- Property editors
- Bean naming
- Importing configuration files
- Custom XML Namespaces

Testing Spring Application

- Unit Testing vs. Integration Testing
- Unit Testing
- Stubs vs. Mocks
- Integration Testing

AOP with Spring

- Spring AOP
- Implementing Aspect
- Configure the Aspect as a Bean
- Property changes with context
- Useful spring JoinPoint Context
- Context Selecting pointcuts
- Advices with spring Aop
- Advice best practices
- Spring AOP XML

Spring Data Access

- Spring in enterprise data access
- Spring resource management
- Spring data access support
- Data access in a layered architecture
- Common data access configurations

Spring JDBC Access

- Spring JDBC Template
- JDBC Template Internals
- Creating JDBC DAO
- Configuring the Application

Spring Transaction Management

- Why use transactions?
- Local transaction management

- Programmatic JTA
- Declarative transactions
- Spring transaction management
- Transaction propagation

Object-Relational Mapping with Spring and JPA

- Object-Relational Mapping - ORM
- Java Persistence API – JPA
- Configuration and Project Setup
- Relational Mapping
- Query and JPQL
- Spring with JPA

Spring MVC

- MVC and Spring MVC
- Spring MVC configuration
- Dispatcher Servlet, request life cycle
- Creating controllers
- Mapping requests to controllers
- Validating input

RESTful Services

- RESTful web services with Spring MVC

Spring Boot

- Spring Boot Overview
- Installations
- Configuration
- Starters
- A Simple example. Running example
- Creating executable jar

Audience

Developers who aim to develop Java applications within the Spring framework.

Duration

3 days

Format

Instructor Lead

Prerequisites

Core Java Syntax - (D-EJAVA-301-001) Effective JAVA Programming..

3.13.2 Spring Boot

Code

D-SPR-BOOT

Overview

This course will introduce you to Spring Boot which takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

Topics Covered:

I. Introduction to Spring Boot
   a. Introducing Spring Boot
   b. Installing Spring Boot
   c. Developing your first Spring Boot application
II. Using Spring Boot
   a. Structuring your code
   b. Configuration Classes
   c. Running Application
III. Customizing Configuration
IV. Testing with Spring Boot
V. Spring Boot Actuator: production-ready features
VI. Microservices with SpringBoot
VII. Deploying Spring Boot applications

Audience

This course is in advanced series and designed for senior developers, software & system architects, project managers involved with the development of Spring Boot applications

Duration

3 days

Format

Instructor Lead

Prerequisites

Prior experience of Spring Framework and Maven are required

3.13.3 Building Modern Web Applications with Spring Framework, HTML5 and JavaScript Technologies

Look At: 3.4.5. Building Modern Web Applications with Spring Framework, HTML5 and JavaScript Technologies

3.13.4 Developing RESTful Services with Spring

Code

D-SPR-REST

Overview

This course enables the experienced Java developer to use the Spring MVC framework to create RESTful web services. We begin by developing fluency with the Spring container and configuration practices, and then learn the annotation-driven MVC system for REST controllers. We will introduce you to Spring Boot which takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

Spring Data's mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store.

Spring Data REST is part of the umbrella Spring Data project and makes it easy to build hypermedia-driven REST web services on top of Spring Data repositories.

Topics Covered:

1. Introduction to Spring Boot
    1.1. Introducing Spring Boot
    1.2. Installing Spring Boot
    1.3. Developing your first Spring Boot application
2. Using Spring Boot
    2.1. Structuring your code
    2.2. Configuration Classes
    2.3. Running Application
3. Spring MVC
    3.1. MVC and Spring MVC
    3.2. Spring MVC configuration
    3.3. Dispatcher Servlet, request life cycle
    3.4. Creating controllers
    3.5. Mapping requests to controllers
    3.6. Validating input
4. RESTful Services
    4.1. RESTful web services with Spring MVC
    4.2. Handling Requests

4.3. Producing Responses
4.4. Error Handling and Validation

5. Object Relational Mapping-ORM
   5.1. Introduction to JPA
   5.2. Entities
   5.3. Multiplicity in Entity Relationships
   5.4. Entity Inheritance
   5.5. Managing Entities
   5.6. Spring and JPA
   5.7. Java Persistence Query Language (JPQL)
   5.8. Criteria API
6. Introducing Spring Data Project
7. Persistence with Spring Data JPA
   7.1. Spring Data Project
   7.2. Working with Repositories
   7.3. Defining Query Methods
   7.4. Query Creation from Method Names
   7.5. Custom Repositories
8. Exporting Spring Data JPA Repositories as REST Services: Spring Data Rest
   8.1. Configuring Spring Data REST
   8.2. Basic settings for Spring Data REST
   8.3. Repository resources
   8.4. Paging and Sorting
   8.5. Domain Object Representations

Audience

This course is in advanced series and designed for senior developers, software & system architects, project managers involved with the development of Restful applications with Spring MVC.

Duration

3 days

Format

Instructor Lead

Prerequisites

Prior experience of Spring Framework and Maven are required.

3.14 SQL

3.14.1 PL/SQL Programming

Code

D-SQL

Description

Topics in this course :

- What is SQL?
- What is PL/SQL?
- Data Types
- Table basics
- Creating tables
- Selecting data
- Inserting records
- Updating records
- Deleting records
- Drop/alter table
- Primary Keys
- Foreign Keys
- Creating Index
- Some important SQL Functions
- SQL Joins
- SQL Functions, Procedures and Triggers
- Some Important performance issues

Audience

Duration

1 day

Format

Instructor Lead

Three-day instructor-led class with approximately 50% hands-on labs

Prerequisites

3.15 WEB

3.15.1 Building Rich Internet Applications with Html5, CSS3 and JavaScript

Code

D-J-HTML5

Overview

HTML5 is a markup language used for structuring and presenting content for the World Wide Web and a core technology of the Internet. It is the fifth revision of the HTML . Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices. HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). This course provides an accelerated introduction to HTML5, CSS3, and JavaScript and helps students learn fundamentals of HTML5/CSS3/JavaScript programming skills. It focuses on using HTML5/CSS3/JavaScript to implement programming logic, define and use variables, perform looping and branching, develop user interfaces, capture and validate user input, store data, and create well-structured applications. There seems to be a consensus of opinion that HTML5 will be the next generation web platform to be embraced by all the major players

Course Content

1. Modern Web Application Design Strategies
    a. Graceful Degradetion/Progressive Enhancement Strategies
    b. Web Design Approaches: Fixed, Fluid, Adaptive, Responsive
    c. JavaScriptMVC and SPA Concerns
2. Overview of Html5 Technologies
3. Advanced JavaScript Features
    a. Functions, Closures, Inner Functions, Self Invoking Functions
    b. Object and prototypes
    c. Object-Oriented JavaScript: Inheritance, Encapsulation, singletons
    d. Global and local variables
    e. Namespaces
    f. Module pattern
4. Developing Client-Side web applications with JavaScript MVC Frameworks
    a. AngularJS
    b. KnockoutJS
    c. RequireJS

    d. SammyJS
5. JavaScript Libraries: JQuery
6. Server-Side
    a. Restful APIs
    b. JSON, JSON Mapping
7. HTML5 – CSS3 New Features
8. HTML5 APIs
9. Offline Applications

Audience

This course is in advanced series and designed for java developers, software & system architects

Duration

2 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of Java, JavaEE and Html is a prerequisite for this course

3.15.2 Developing Client-Site Web Applications with AngularJS

Code

D-ANG

Overview

The course teaches how to develop modern web applications using a combination of JavaScript and AngularJS. You will learn how to apply these various techniques to create modern client side web applications that use the power of the client via the rich document object model and the JavaScript runtime environment the browser provides.

Course Content

1. Advanced JavaScript Features
    - Functions, Closures, Inner Functions, Self Invoking Functions
    - Object and prototypes
    - Object-Oriented JavaScript: Inheritance, Encapsulation, singletons
    - Global and local variables

- Namespaces
- Module pattern

2. JavascriptMVC and SPA (Single Page Application) concepts
3. Developing Client-Side web applications with Angular JS
   - What is Angular?
   - Directives
   - Modules, Controllers and Scopes
   - Routing and Templating
   - Communicating with Servers
   - Forms, Models and Validations
   - Custom directives
   - AngularUI package
4. JavaEE Restful web services

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

1 day

Format

Instructor Lead

Prerequisites

A fundamental knowledge of JavaScript, Ajax, HTML and CSS is required

3.15.3 Building Responsive Web Apps with Angular 2

Code

D-ANG2

Overview

Google's Angular 2 and AngularJS are extremely productive frameworks for complex “Single Page Apps”.

Angular 2 is a complete re-write of the entire framework and will not be backwards-compatible with AngularJS 1.x apps. ES6 and "Evergreen" modern browsers will be targeted as part of the move towards next-gen web development.

The course teaches how to develop modern web applications using a combination of JavaScript and Angular2. You will learn how to apply these various techniques to create modern client side web applications that use the power of the client via the rich document object model and the JavaScript runtime environment the browser provides.

Course Content

- Introduction
    - Angular 2 Architecture
    - Building your first Angular2 App
    - Overview of TypeScript and ECMAScript 6
    - Classes and Imports
    - Displaying data
    - Using Directives
    - User Input
- FORMS
    - Template-driven forms
    - Model-driven forms
    - Control, ControlGroup & FormBuilder
    - Validation and error messages
    - Custom validators
    - Asynchronous validators
- Dependency Injection
    - Creating services
    - Configuring providers
    - Defining provider recipes
    - Understanding the injector tree
    - Injecting dependencies using tokens

- Views and Templates
    - Interpolation
    - Property Binding
    - Attribute, Class, and Style Binding
    - Event Binding
    - Additional techniques
- Routing
    - Configuring routes
    - Linking to routes
    - Guards
    - Child routes
    - Sibling routes
    - Lazy loading routes
- HTTP
    - Performing GET, PUT & POST requests
    - Getting started with observables
    - Configuring request headers
- Angular 2 Pipes
    - Overview of Pipes
    - Built-in Pipes.
    - Parameterising a Pipe
    - Chaining Pipes
    - Stateful Pipes
    - Custom Pipes
- Directives
    - Overview of Directives in Angular 2.0
    - Component Directives
    - Decorator Directives
    - Attribute Directives
    - Structural Directives
    - Template Directives
    - Controllers

Audience

Angular 1 and other JavaScript developers who need to get up to speed on Angular 2 in two days.

Duration

2 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of JavaScript, Ajax, HTML and CSS is required

3.15.4 Developing Client-Site Web Applications with ReactJS

Code

D-RCT

Overview

React is a library developed by Facebook, and it is designed to build large applications with data that changes over time. Developers can use ReactJS to create user interfaces (UI) with high performance, where React would automatically manage all UI updates. React is all about building reusable web components, and it also renders on the server using Node.js.

Course Content

1-What is React?
2-Why React ?
3-Installation Types
4-The Core of React
5-Properties of React
6-React Component Design
7-"props" and "state" Keywords
8-React Component Lifecycle and Rendering
9-Usage of "Mixins"
10-What is JSX?
11-JSX Fundamentals
12-React Web Component Samples
13-Introduction to Flux
14-Use React in Flux Architecture

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

2 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of JavaScript, Ajax, HTML and CSS are required

3.15.5 Next-generation Web Applications with full stack JavaScript and HTML5

Code

D-N-WEB

Overview

The Modern Web Development course teaches how to develop modern web applications using a combination of JavaScript, JQuery, AngularJS, Ajax, Knockout, RequireJS, SammyJS frameworks. You will learn how to apply these various techniques to create modern client side web applications that use the power of the client via the rich document object model and the JavaScript runtime environment the browser provides.

Description

1. Modern Web Application Design Strategies
    a. Graceful Degradetion/Progressive Enhancement Strategies
    b. Web Design Approaches: Fixed, Fluid, Adaptive, Responsive
    c. JavaScriptMVC and SPA Concerns
    d. HTML5 Principles
2. Advanced JavaScript Features
    a. Functions, Closures, Inner Functions, Self Invoking Functions
    b. Object and prototypes
    c. Object-Oriented JavaScript: Inheritance, Encapsulation, singletons
    d. Global and local variables
    e. Namespaces
    f. Module pattern
3. Developing Client-Side web applications with JavaScript MVC Frameworks
    a. JQuery
    b. AngularJS
    c. Knockout
    d. RequireJS
    e. SammyJS
    f. Bootstrap
4. Server-Side
    a. NodeJS

    b. Restful APIs
    c. JSON, JSON Mapping
5. NoSQL Database: MongoDB
6. Security in new generation web applications

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

4 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of JavaScript, Ajax, HTML and CSS is required.

3.15.6 Essential Tools & Libraries for Building Client Side Applications

Code

D-M-JS

Overview

The Web Platform has gone a long way since HTML5 was first made popular and people started looking into JavaScript as a language that could do build complex apps. Many APIs have emerged and there is an abundance of content out there about how the browser can leverage all this.

JavaScript is the most popular, and probably the most important, programming language today. Learning JavaScript is a great journey, but knowing the language and its syntax is only the beginning. JavaScript is a scripting language initially designed to enhance web pages, but is now used in almost every way imaginable.

Advances have been made that allow JavaScript to run on the server-side as well as be compiled into native phone application code. Today's JavaScript developer is part of a rich ecosystem filled with hundreds of IDEs, tools, and frameworks.

Once the novice becomes a journeyman, they are confronted with a seemingly endless number of tools and libraries which are used to create, manage and maintain modern JavaScript code.

In this advanced course we will discuss modern JavaScript development tools for building, automation, testing, packaging, modules& dependecy management

Description

1. Modules & Dependency Management
    a. RequireJS
    b. RequireJS Optimizer
    c. Browserify
    d. Webpack
2. Build and Automation
    a. Gulp
    b. Grunt
    c. WebPack build features
3. Testing
    a. Jasmine
    b. Karma
4. Packaging
    a. Bower

    b. Npm
5. Server Side JavaScript: NodeJS
6. Code Analysis
    a. JSLint
    b. JSHint

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

2 days

Format

Instructor Lead

Prerequisites

Advanced knowledge of JavaScript is required.

3.15.7 JS NEXT

Code

D-J-NX

Overview

JavaScript is everywhere. Once relegated to an Internet fad, the malleable programming language has evolved along with the Web and now finds itself entrenched in modern browsers, complex Web applications, mobile development, server-side programming, and in emerging platforms like the Internet of Things.

Underlying that browser-centric user and developer shift, JavaScript has developed a robust ecosystem of third-party and open-source libraries, frameworks, tools, implementations and superset languages woven into the backbone upon which Web and mobile development is now built

Over the past decade, starting with jQuery empowering Web developers with client-side scripting, every popular plug-in has filled another gap in the language and its capabilities.

After more than 15 years without a major update, the international standards organization Ecma is finally set to release ECMAScript 6—a comprehensive update to standardized JavaScript—in June of this year. With ECMAScript 6, JavaScript is getting a ton of new features targeted towards making life easier for people building transpilers, i.e. languages that compile down to JavaScript, like CoffeeScript and C++.

Standardised HTML elements come with semantics, and browsers can do all kinds of smart things with them. The idea of Web Components and specifically their custom elements, is that you no longer depend on the browser to provide you with elements: you can bring your own. With Web Components, browsers (will) give website owners the power to bring their own elements. This gives a lot of freedom and is a big addition to the web as a platform. It is innovation we should welcome.

Course Topics

1. ECMAScript 2015
    - Constants
    - Scoping
    - Arrow Functions
    - Template Literals
    - Destructuring Assignment
    - Modules
    - Classes
    - Iterators

- Generators
- Map/Set & WeakMap/WeakSet
- Typed Arrays
- Promises
- Modules

2. Polymer & Web Components
   - Web Component Polyfills
   - Templates
   - Custom elements
   - Shadow dom
   - HTML Imports
   - Polymer, Mozilla-Brick, X-Tag
3. ReactJS

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

3 days

Format

Instructor Lead

Prerequisites

Advanced knowledge of JavaScript is required.

3.15.8 Building Modern Web Applications with Spring Framework, HTML5 and JavaScript Technologies

Code

D-M-WEB-SPR

Overview

This course teaches how to develop modern web applications using a combination of Spring Framework, Html5 and JavaScript technologies. You will learn how to apply these various techniques to create modern client side web applications that use the power of the client via the rich document object model and the JavaScript runtime environment the browser provides.

Description

1. Modern Web Application Design Strategies
    - Graceful Degradetion/Progressive Enhancement Strategies
    - Web Design Approaches: Fixed, Fluid, Adaptive, Responsive
    - JavaScriptMVC and SPA Concerns
    - HTML5 Principles
2. Advanced JavaScript Features
    - Functions, Closures, Inner Functions, Self Invoking Functions
    - Object and prototypes
    - Object-Oriented JavaScript: Inheritance, Encapsulation, singletons
    - Global and local variables
    - Namespaces
    - Module pattern
3. JavaScript Frameworks
    - JQuery
    - AngularJS
    - Knockout
    - RequireJS
    - Bootstrap
4. JSON, JSON Mapping
5. Security in new generation web applications
6. Spring Framework
    - Spring Architecture
    - Setting configuration

- Spring Core Concepts
- Writing bean definitionsn (annotation based / XML Based)
- Creating an application context
- Design Patterns: Inversion of Control&Dependency Injection
- Spring Data Access
- Spring JDBC Access
- Object-Relational Mapping with Spring and JPA
- Introduction to Spring MVC
- RESTful web services with Spring MVC
- Spring Boot

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

3 days

Format

Instructor Lead

Prerequisites

A fundamental knowledge of Java, JavaScript, Ajax, HTML and CSS is required.

3.15.9 Developing Effective Rich Client Apps with JavaScript and NoSQL Technologies

Code

D-E506

Overview

This course is designed for application developers who needs to learn how to develope rich client applications with javascript and NoSQL databases.

Course covers mongodb, serverside javascripting with node.js , expose nosql data with rest services , building rich client user interface with javascript. At the end of the course you will understand how to build end to end rich client applications with lightweight backend services with rich frontend features.

Description

Course Content

1. JavaScript
    - Introduction
    - Core language features
    - Javascript variables
    - Functions : Anonymous Function, Immediately-Invoked Function Expression (IIFE)
    - Encapsulation
    - Objects, Object Oriented JS
    - Prototypes
    - Inheritance
    - Closures
    - Patterns (Namespace, Module Pattern)
2. JSON
3. jQuery
    - Introduction
    - jQuery Selector
    - Events
    - Ajax
    - jQuery and Ajax
    - JavaScript best practices
4. Mongo DB
    - What is noSQL Databases and why?
    - Introduction to MongoDB
        - What is MongoDB and Why?
            - Core concepts

        - Environments
      - Different deployment models
      - Installation
    - The CAP Theorem and MongoDB
    - CRUD and the MongoDB Shell
      - Introduction to the MongoDB API and the core concepts of documents and collections
      - Indexing and Data Modeling
      - Indexing, query profiling and the query optimizer
      - Some schema design case studies
      - Practice modeling various domains
      - Schemas for atomic operations
    - Drivers
      - How the drivers work in general
      - Driver APIs with examples
    - How to Query from Mongo DB
5. NodeJS
    - node.js and mongodb integration
    - What Is Node.js?
    - When To Use Node.js
    - Some important Node.js Modules (expressjs/npmjs/requestjs etc)
    - Understanding The Node.js Event Loop
    - Installation
    - Quick Start
    - Building A Node.js Web Server
    - Node.js Web Applications Using Express
    - Node.js and MongoDB using MongoJS or Mongoose
    - Templating (ejs or jade)
    - Optional (if time available): security (connect/passport/everyauth)
6. Rest services
    - Rest Concepts
    - Building rest services with NodeJS

Audience

This course is in advanced series and designed for senior developers, software & system architects

Duration

3 days

Format

Instructor Lead

Prerequisites

This course requires basic java and javascript knowledge

3.15.10 Surveying Web 2.0 Applications

Code

D-WEB20-402-001

Overview

The convergence of Windows and the Web is upon us. Google Maps, Gmail, Flickr and a variety of new AJAX and Rich Internet applications have begun to legitimize moving beyond HTML to deliver interactive applications that deliver the best of the web and the best of the desktop experience. We will show how these techniques are changing the way designers think about their application user interface designs. Today's customers are more demanding and have high expectations from your applications. Implementing effective user interfaces can mean the difference between success and failure in your software development strategy.

This course set is designed to provide students with the technological knowledge and skills to build Modern Web applications. Students will learn how to build rich interactive browser-based user interfaces, the building of hybrid offline/online web applications, and how to 'syndicate' content. Students will also be taught how to use and create a web API, how to combine data and services from different sources into a single application, including video streams, and how to provide forums for users to interact

HTML5 is a markup language used for structuring and presenting content for the World Wide Web and a core technology of the Internet. It is the fifth revision of the HTML . Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices. HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). This course provides an accelerated introduction to HTML5, CSS3, and JavaScript and helps students learn fundamentals of HTML5/CSS3/JavaScript programming skills. It focuses on using HTML5/CSS3/JavaScript to implement programming logic, define and use variables, perform looping and branching, develop user interfaces, capture and validate user input, store data, and create well-structured applications. There seems to be a consensus of opinion that HTML5 will be the next generation web platform to be embraced by all the major players

JEE Core Concepts

- Internet Technologies, Overview
- Distributed Web Architectures and JavaEE

XML Technologies

- xml, xsd, jaxb

Java And Web Services

- Web Services Technologies
- Web Service Styles:REST / SOAP
- Enterprise Web Services - JAX-WS
- Rest style services

Developing Modern Web Applications

- Modelling Modern Web Applications
- Structure of Web apps
- HTML5 Overview
- Advanced HTML5 Features
- CSS3 Features
- CSS & HTML Best Practices
- Web Design Strategies
- Best Design Practices
- CSS Preprocessors and SASS
- JavaScript
- Ajax
- JSON
- JSONP
- Overview of Frameworks
- Advanced JavaScript
    - Object Oriented JavaScript
    - Enterprise JavaScript, MVC with javascript: Backbone.js/ Knockout.js
    - JavaScript Libraries: JQuery
- NodeJS
- Using web services in the web applications
    - Jax-RS 2 Client API
    - Jax-RS 2 and JSON-P entegration
- Useful Metatags
- Working with Local Data
- Working with Remote Data
- Working With Media
- Debugging
- Security for Web Applications

Audience

- Project managers who are responsible for establishing or managing a web project or Internet strategy.
- Project leaders who need to know proven steps for web-enabling existing client/server applications.
- GUI designers who need to know how and when to use the new Web controls when creating user interfaces.
- Software Developers who are looking to expand their knowledge of advanced UI design.
- Interaction Designers who are responsible for managing and implementing UI design
- Business Analysts responsible for documenting requirements for complex user interface applications

Duration

4.5 days

Format

Instructor Lead

Prerequisites

Experience with HTML, JavaScript, Java and web application programming.

3.15.11 JQuery Fundamentals

Code

D-JQRY

Overview

This course builds a strong foundation in jQuery, a JavaScript library that helps developers build robust, interactive web applications. In this class, attendees will gain a foundational knowledge of jQuery and JQueryUI by learning about each of the interactions and widgets. Attendees will learn how these components can work together to build a user interface that is interactive while easy to create.

Description

1. jQuery Core
    a. Installing jQuery
    b. The jQuery Object
    c. Selecting DOM Elements
    d. Creating DOM Elements
    e. Traversing Data
    f. Manipulating the DOM
    g. Adding Content to Elements
    h. Changing the Appearance of Elements
    i. Manipulating Attributes
    j. Adding Event Handlers to Elements
2. jQuery UI
    a. Effects
    b. Animations
    c. jQuery UI Library
    d. Interactions
    e. Widgets
    f. Themes
    g. Using the APIs
    h. Creating Your Own Theme
3. jQuery Data
    a. JQuery and Ajax
    b. REST Overview
    c. Loading Data with $.ajax, get and post
    d. Data Types (Text, JSON, XML)
    e. Loading JSONP
    f. Data Loading Helpers
    g. CORS
    h. Sending Data
4. jQuery Plugins
    a. Finding and Using Plugins
    b. Writing Your Own Plugin
    c. Creating a Widget with the Widget Factory

d. Using Widgets as Views

Audience

This course is in advanced series and designed for web developers, software & system architects

Duration

1 day

Format

Instructor Lead

Prerequisites

A fundamental knowledge of JavaScript, Ajax, HTML and CSS are required.