**About Spring Training**

The Spring Framework is an open source application framework for Java. This framework has taken the Java software community by storm. Spring provided the technology to develop everything from small, stand-alone applications to large complex, enterprise systems out of simple POJOs (plain old Java objects).

In this class, students are exposed to the light-weight Spring container, configuration, foundational API, and general Spring architecture. Not just a class that focuses on theory, this course is loaded with practical labs and deals with configuration, maintenance and architectural issues. After taking the class, developers will immediately be able to utilize the Spring Framework in their new or existing applications.

**Spring Course Prerequisite**

* Students should have a good understanding of the Java programming language. A basic understanding of relational databases and SQL is very helpful. A basic understanding of XML is also useful.  Students that have attended Complete Java have the necessary background for this course.

**Spring Training Course Objective**

* Learn how to download, setup and configure the Spring Framework
* Explore the Spring Container and Modules
* Discover the Spring philosophies and principles and how they impact application development
* Understand dependency injection
* Learn aspect oriented programming and how it is used to provide cross cutting concerns
* See how to accomplish data access with Spring’s DAO Module
* Understand how Spring deals with transaction management
* Examine Spring’s unit testing framework

## Spring Training Course Duration

* 20 Working days, daily one and half hours

## Spring Training Overview

### SPRING

* Introduction to Spring Framework
* POJO and POJI Model Programming
* Introduction to Modules of Spring
* Core Module
* JDBC/DAO Module
* WEBMVC Module
* ORM Module
* AOP Module
* JEE Module

### SPRING CORE

* Introduction to IOC
* Introduction to Spring Container
* Dependency Injection
* Setter Injection
* Constructor Injection
* Injecting Primitive Data types and Collection classes
* Autowiring
* Inner beans
* Alias Names
* Lazy Initialization
* Bean Inheritance
* Factory methods Configuration
* Bean Life Cycle
* Bean Scopes
* Dependency Check
* depends-on attribute
* Aware Interfaces
* Working with properties files
* Multiple Configuration files
* Pre processor and Post Processors
* I18N
* Annotations

### Spring JDBC/DAO

* Drawbacks of plain JDBC
* JDBC Template
* NamedParameter JDBC Template
* Simple JDBC Template
* All the methods defined in Templates
* DAOSupport classes
* Callback interfaces
* DrivermanagerDatasource
* BasicDataSource
* Combopooled DataSource
* ServerSuppiled DataSource
* Examples of database access operations using spring DAO
* Examples of accessing stored procedures using spring DAO

### Spring ORM

* Drawbacks of Hibernate
* Spring with Hibernate without HibernateTemplate
* Spring with Hibernate with HibernateTemplate
* HiberanteTemplate
* HibernateDAOSupport
* Callback interfaces
* Spring with JPA and other ORM integration

### Spring AOP

* Introduction
* Need of AOP
* Aspect
* JoinPoint
* PointCut
* Advice
* Target
* Weaving
* Dynamic Proxy.
* Spring AOP Architecture
* Before Advice
* After Returning Advice
* Throws Advice
* Around Advice
* After Advice
* XML based AOP
* Schema based AOP
* Annotation based AOP
* Covers Spring 2.0 AOP, @AspectJ Style AOP support

### Spring MVC

* Introduction
* Spring MVC Resources
* Commands or Models
* DispatcherServlet
* Controller Classes
* Spring MVC flow
* Steps to develop the SpringMVC application
* HadlerMappings
* BeanNameUrlHandlerMapping
* SimpleUrlHandlerMapping
* ControllerClassNameHandlerMapping
* Validations
* ViewResolvers
* XmlViewResolver
* ResourceBundleViewResolver

### Spring with JMS Integration

### Spring 3.0 Annotations

### Spring Security

### Spring with Quartz Integration (Scheduler)

### Spring OXM

### Spring WebFlow Overview

### Spring Batch Overview

### Struts-Spring-Hibernate Integration

### IDE : Eclipse