Spring Framework

--Santosh Kumar Mondal

ASSOCIATION

COUPLING

OOP

INHERITENCE

POLYMORPHISM

CONSTRUCTOR

SINGLETON

STATIC

REFLECTION

DI

INJECTION

IOC

AOP

Environment

ECLIPSE IDE

MAVEN

Spring Bean

- A Java Object.
- A Java Object created and managed by Spring Container.

```
<bean name="bean1" class="com.dac.Hello" />
```

Spring Bean Attribute - Basic

- **id**
  - Unique bean identifier.
- **name**
  - Bean identifier, can have multiple aliases separated by comma.
- **class**
  - Fully qualified name of the java class.

Spring Bean Attribute - Continue

- **abstract**
  - instance can't be created.
- **parent**
  - Bean inheritance.
- **primary**
  - **Default bean** to autowire. Used in DI.

Spring Bean Attribute - Continue

- **autowire**
  - Inject dependency. Or collaborate multiple bean.
  - **byName**
    - Autowire by property name.
  - **byType**
    - Autowire by property data type
  - **Contructor**
    - Autowire applies to constructor argument.

Spring Bean Attribute - Continue

- **lazy-init**
  - Default value is false. If true, will create bean on **first** request.
- **init-method**
  - This method will be called after bean instance is created by container.
- **destroy-method**
  - This method will be called when bean instance is removed from the container.

Spring Bean Attribute Continue

- **scope**
  - Application Context Aware
    - **singleton**
    - **prototype**
  - Web Context Aware
    - **request**
    - **session**
    - **global-session**

Spring Bean Property Element

- **property**
  - Inject value into bean property
  - Attributes
    - **name**
      - name of the bean property
    - **Value**
      - Primitive value of the bean property
    - **ref**
      - Reference value of the bean property.

Spring Bean Constructor Element

- **constructor-arg**
  - Inject value into bean during construction.
  - Attributes
    - **name**
    - **value**
    - **ref**
    - **type**
    - **index**

Injecting Collections

- **List**
- **Set**
- **Map**
- **Properties**

Dependency Injection

- Injecting bean of **reference** type.
- Injecting bean - autowire byName
- Injecting bean - autowire byType
- Injecting bean - @autowire Annotation

Annotating & Auto-Discovery Bean

- @Component
  - A general purpose annotation indicating class is a Spring Component.
- @Controller
  - Indicate defined class as Spring MVC Controller.
- @Repository
  - Indicate defined class as Spring data repository.
- @Service
  - Indicate defined class as Service. Used in Restful Web Service.

SPRING MVC

STEP -1 Create Dynamic Web Project. (Note : Do create web.xml)

STEP -2 Convert to Maven Project.

STEP -3 Add Dependency. CORE/CONTEXT/WEBMVC/JDBC/mysql-connector

STEP -4 Update web.xml, Add Servlet and Servlet Mapping.
(org.springframework.web.servlet.DispatcherServlet)

STEP -5 Create spring configuration xml file. At the location of web.xml.

SPRING MVC

STEP -6 File name of spring configuration file. {SPRING_SERVLET_NAME}-servlet.xml

STEP -7 Add spring-mvc schema at root tag of spring configuration file.

STEP -8 Update Spring Configuration file. Add following tags.
<context:component-scan base-package="“com.dac.servlet”>
<mvc:annotation-driven>

STEP -9 Update Spring Configuration file with view resolver.
org.springframework.web.servlet.view.InternalResourceViewResolver

SPRING MVC

STEP -9 Create view folder inside the WEB-INF to add jsps

STEP -10 Initialize npm and install bootstrap at Webcontent folder.

STEP -11 Create first Controller using Spring.

STEP -12 ${pageContext.request.contextPath}. *Use the command to include bootstrap css in jsp.*

```
<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
```

Aspect Oriented Programming

- AOP helps to modularize Cross Cutting Concern.
- **Cross Cutting Concern** are **common functionality** that affects the multiple point of an application.
  - Security
  - Logging
  - Validation

AOP Terminology

- ASPECT
  - The key unit of AOP.
  - A java class with cross cutting concern or common functionality.
  - Application can have multiple number of aspects

AOP Terminology

- ADVICE
  - An actual action to be taken, before or after method execution.
  - Job of an aspect is Advice.
- Types of Advice
  - Before
    - Functionality takes place before method is invoked.
  - After
    - Functionality takes place after method is invoked
  - Around
    - Functionality takes place before and after method is invoked.

AOP Terminology

- Join Point
  - A point in the execution of an application, where Aspect will be plugged.
  - In simple, **where** to apply the aspect.
- Pointcut
  - A set of one or more Joint Point.

Selecting JoinPoint

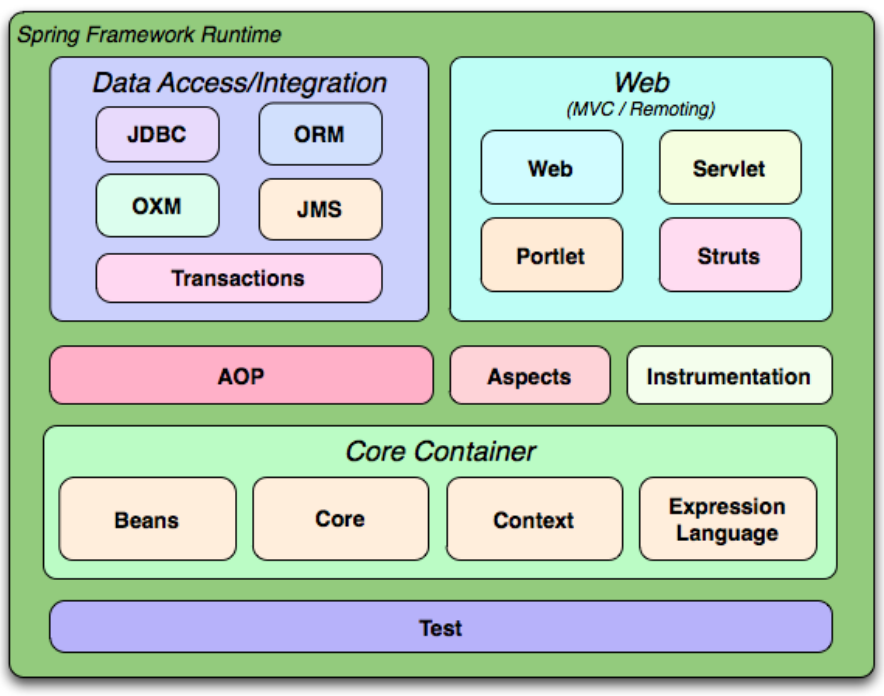
- **execution()**
  - Matches join point that are method execution.
- **within()**
  - Limits matching to join point with certain type.
- **args()**
  - Limits joint point to the execution of method, whose argument are instance of given type

Writing JoinPoint

execution(* com.dac.Student.**program**(..))

execution(* com.dac.Student.**program**(String, ..) and args(name, ..))

Spring Framework Overview



Thank You