Spring Core Annotations

@Component Annotation

The @Component annotation indicates that an annotated class is a “spring bean/component”.

The @Component annotation tells Spring container to automatically create Spring bean.

@Autowired Annotation

The @Autowired annotation is used to inject the bean automatically

The @Autowired annotation is used in constructor injection, setter injection and field injection

@Qualifier Annotation

@Qualifier annotation is used in conjunction with Autowired to avoid confusion when we have two or more beans configured for same type.

@Primary Annotation

We use **@Primary** annotation to give higher preference to a bean when there are multiple beans of the same type.

@Bean annotation

- @Bean annotation indicates that a method produces a bean to be managed by the **Spring container**. The @Bean annotation is usually declared in Configuration class to create Spring Bean definitions.

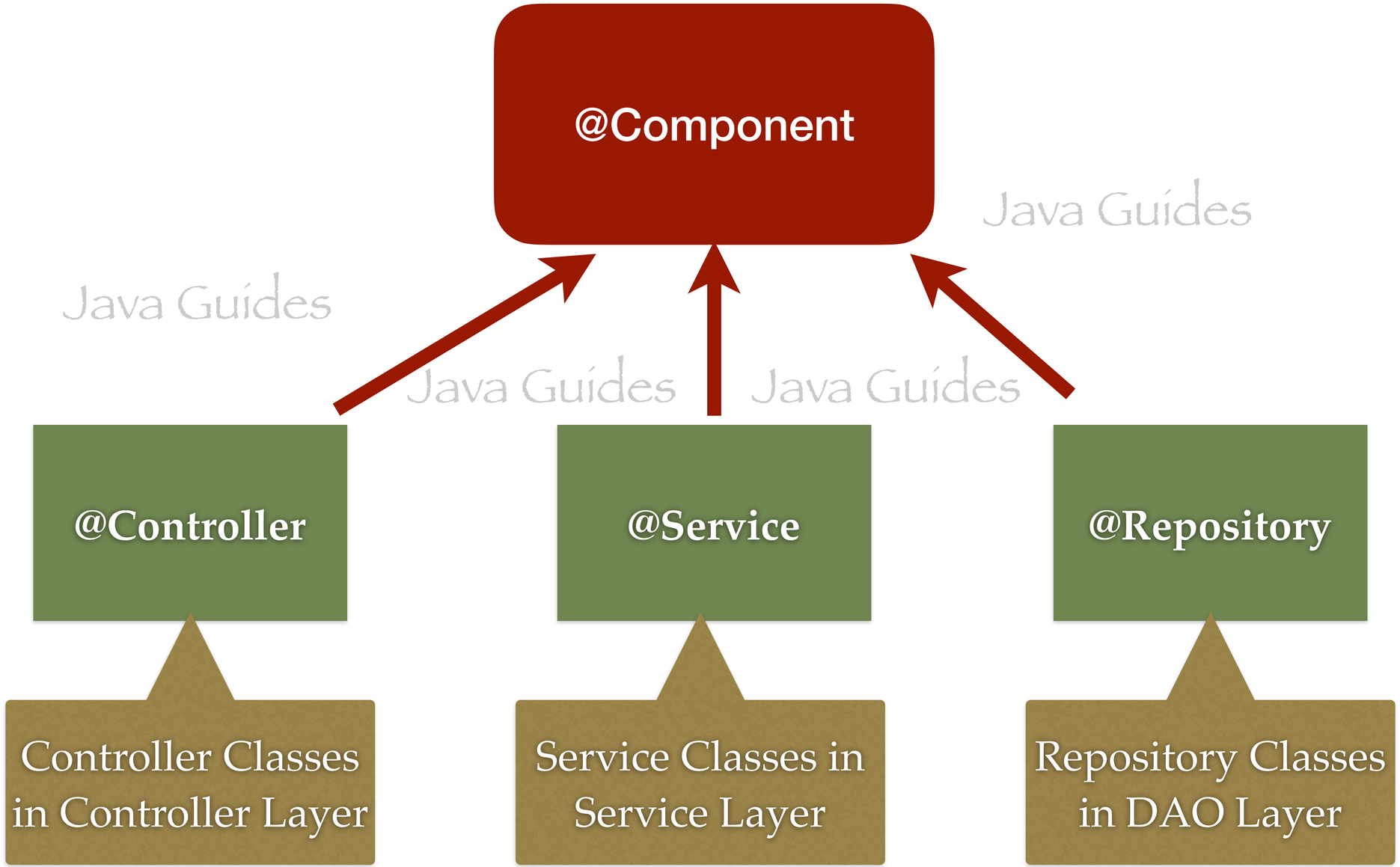
- By default, the bean name is same as **method name.** We can specify bean name using **@Bean(name = “beanName”)**.

- @Bean annotation provides **initMethod** and **destroyMethod** attributes to perform certain actions after bean initialization or before bean destruction by a container.

Stereotype annotations

1. These annotations are used to create Spring beans automatically in the application context (Spring IoC container)
2. The main stereotype annotation is @Component.
3. By using this annotation, Spring provides more Stereotype meta annotations such as @Service, @Repository and @Controller
4. @Service annotation is used to create Spring beans at the Service layer
5. @Repository is used to create Spring beans for the repositories at the DAO layer
6. @Controller is used to create Spring beans at the controller layer

Spring Stereotype Annotations



@Lazy Annotation

- By default, Spring creates **all singleton beans eagerly** at the startup/bootstrapping of the application context.

- You can load the Spring beans lazily (on-demand) using **@Lazy** annotation

- @Lazy annotation can used with @Configuration, @Component and @Bean annotations

- **Eager intialization is recommended**: to avoid and detect all possible errors immediately rather than at runtime.

Spring Bean Scopes

- The latest version of the Spring framework defines 6 types of scopes:
    - singleton
    - prototype
    - request
    - session
    - application
    - websocket
- The last four scopes mentioned, request, session, application and websocket, are only available in a web-aware application.

@Scope Annotation

- @Scope annotation is used to define a scope of the bean
- We use **@Scope** to define the scope of a **@Component** class or a **@Bean** definition.
- **Singleton:** only one instance of the bean is created and shared across the entire application. This is the default scope.
- **Prototype:** a new instance of the bean is created every time it is requested.

@Value Annotation

- Spring @Value annotation is used to assign default values to variables and method arguments.

- @Value annotation is mostly used to get value for specific property key from the properties file.

- We can read spring environment variables as well as system variables using @Value annotation.

@PropertySource and @PropertySources Annotations

- Spring **@PropertySource** annotation is used to provide properties file to Spring Environment.
- Spring **@PropertySources** annotation is used to provide multiple properties files to Spring Environment.
- These annotation is used with **@Configuration** classes.
- Spring **@PropertySource** annotation is repeatable, means you can have multiple PropertySource on a Configuration class.
- We can use @Value annotation and Environment class to read the Property file

@ConfigurationProperties Annotation

- @ConfigurationProperties bind external configurations to a strongly typed bean in your application code. You can inject and use this bean throughout your application code just like any other spring bean.

@ConfigurationProperties Annotation

Properties file

```
app.name=ConfigurationPropertiesDemo
app.description=${app.name} is a spring boot app that demonstra
app.upload-dir=/uploads

## Nested Object Properties (security)
app.security.username=javaguides
app.security.password=123456
app.security.roles=USER,ADMIN,PARTNER   # List Property
app.security.enabled=true

## Map Properties (permissions)
app.security.permissions.CAN_VIEW_POSTS=true
app.security.permissions.CAN_EDIT_POSTS=true
app.security.permissions.CAN_DELETE_POSTS=false
app.security.permissions.CAN_VIEW_USERS=true
app.security.permissions.CAN_EDIT_USERS=true
app.security.permissions.CAN_DELETE_USERS=false
```

Java Class (Spring Bean)

```
@ConfigurationProperties(prefix = "app")
@Component
public class AppProperties {
    private String name;
    private String description;
    private String uploadDir;
    private Security security = new Security();
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }
    public String getUploadDir() { return uploadDir; }
    public void setUploadDir(String uploadDir) { this.uploadDir = uploadDir; }
    public Security getSecurity() { return security; }

    public static class Security{
        private String username;
        private String password;
        private List<String> roles = new ArrayList<>();
        private boolean enabled;
        private Map<String, String> permissions = new HashMap<>();

        public String getUsername() { return username; }

        public void setUsername(String username) { this.username = username; }

        public String getPassword() { return password; }
```