ol... 10 Features in Java... Nagarro Yashi outlo... 5 Hidden Secrets in... Some Java 8 Strea... Interview Preparati... HRIS Oracle Applica... Timeshhet

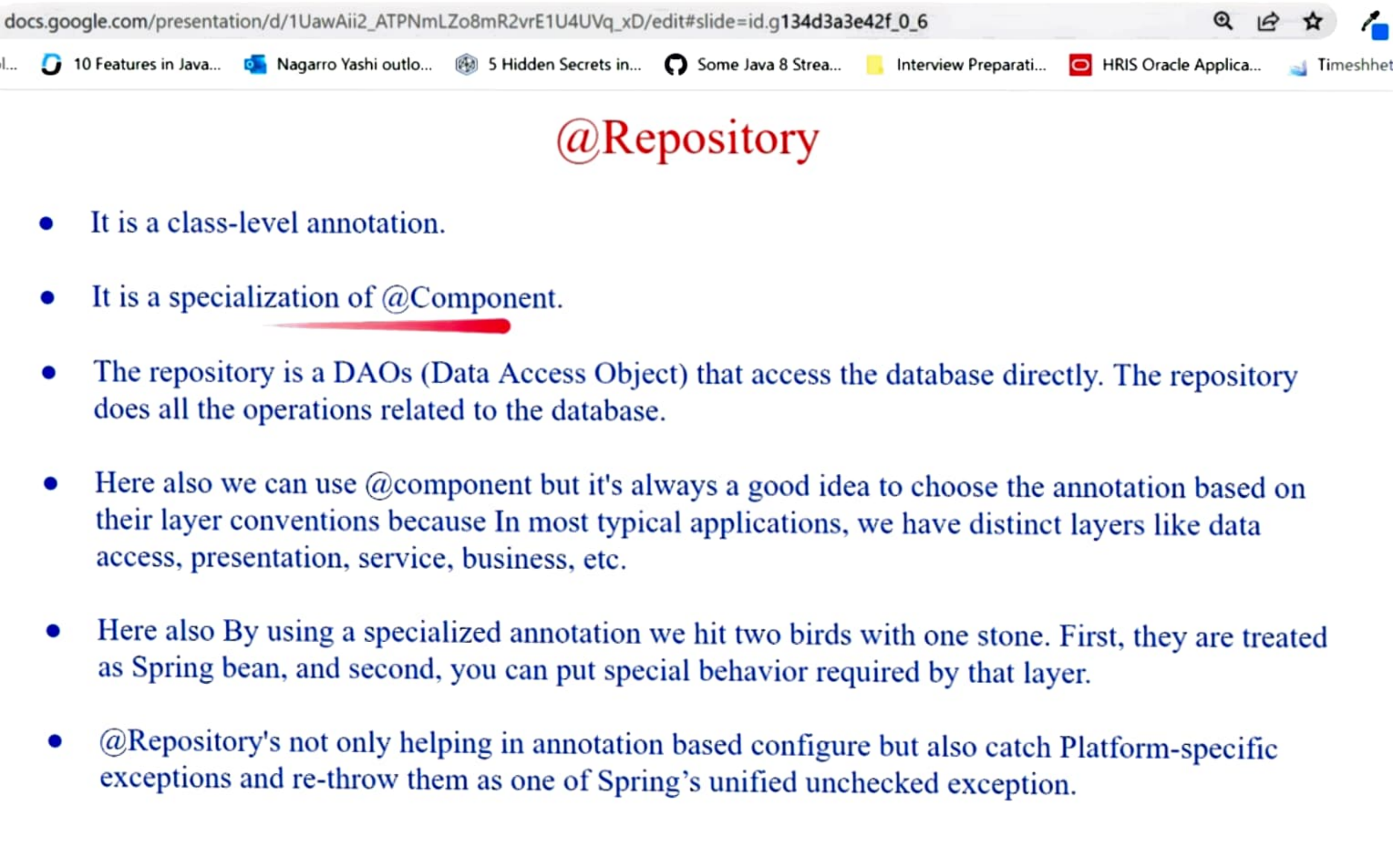
Summary

- @Component is a generic stereotype for any Spring-managed component or bean.
- @Repository is a stereotype for the persistence layer.
- @Service is a stereotype for the service layer.
- @Controller is a stereotype for the presentation layer (spring-MVC).

- All of them are used to auto-detect Spring beans when context scanning is enabled and essentially provide the same functionality with respect to dependency injection.

- Their only difference comes in their Specific purpose i.e. @Controller is used in Spring MVC to define controller, which are first Spring bean and then the controller. @Repository is used in the Data Access layer.

SUBSCRIBED



@Repository

- It is a class-level annotation.
- It is a specialization of @Component.
- The repository is a DAOs (Data Access Object) that access the database directly. The repository does all the operations related to the database.
- Here also we can use @component but it's always a good idea to choose the annotation based on their layer conventions because In most typical applications, we have distinct layers like data access, presentation, service, business, etc.
- Here also By using a specialized annotation we hit two birds with one stone. First, they are treated as Spring bean, and second, you can put special behavior required by that layer.
- @Repository's not only helping in annotation based configure but also catch Platform-specific exceptions and re-throw them as one of Spring's unified unchecked exception.

@Service

- It is also used at class level.
- It is a specialization of @Component.
- It tells the Spring that class contains the business logic.
- But at the end The main task of this annotation is also to mark the class capable to become a bean in Spring container similar to component. We can use @component also and it will work the same as this annotation does. They are technically the same

docs.google.com/presentation/d/1UawAii2_ATPNmLZo8mR2vrE1U4UVq_xD/edit#slide=id.g134d3a3e42f_0_6

10 Features in Java... Nagarro Yashi outlo... 5 Hidden Secrets in... Some Java 8 Strea... Interview Preparati... HRIS Oracle Applica... Timeshhet

What will happen if we replace @Controller with @Component

- By using @Controller annotation we do two things,
  - We declare that this class is a Spring bean and should be created and maintained by Spring ApplicationContext,
  - We indicate that its a controller in MVC setup. This latter property is used by web-specific tools and functionalities.

- DispatcherServlet will look for @RequestMapping on classes that are annotated using @Controller but not with @Component.

- This means @Component and @Controller are the same with respect to bean creation and dependency injection but @Controller is a specialized form of @Component. Even if you replace @Controller annotation with @Compoenent, Spring can automatically detect and register the controller class but it may not work as you expect with respect to request mapping.

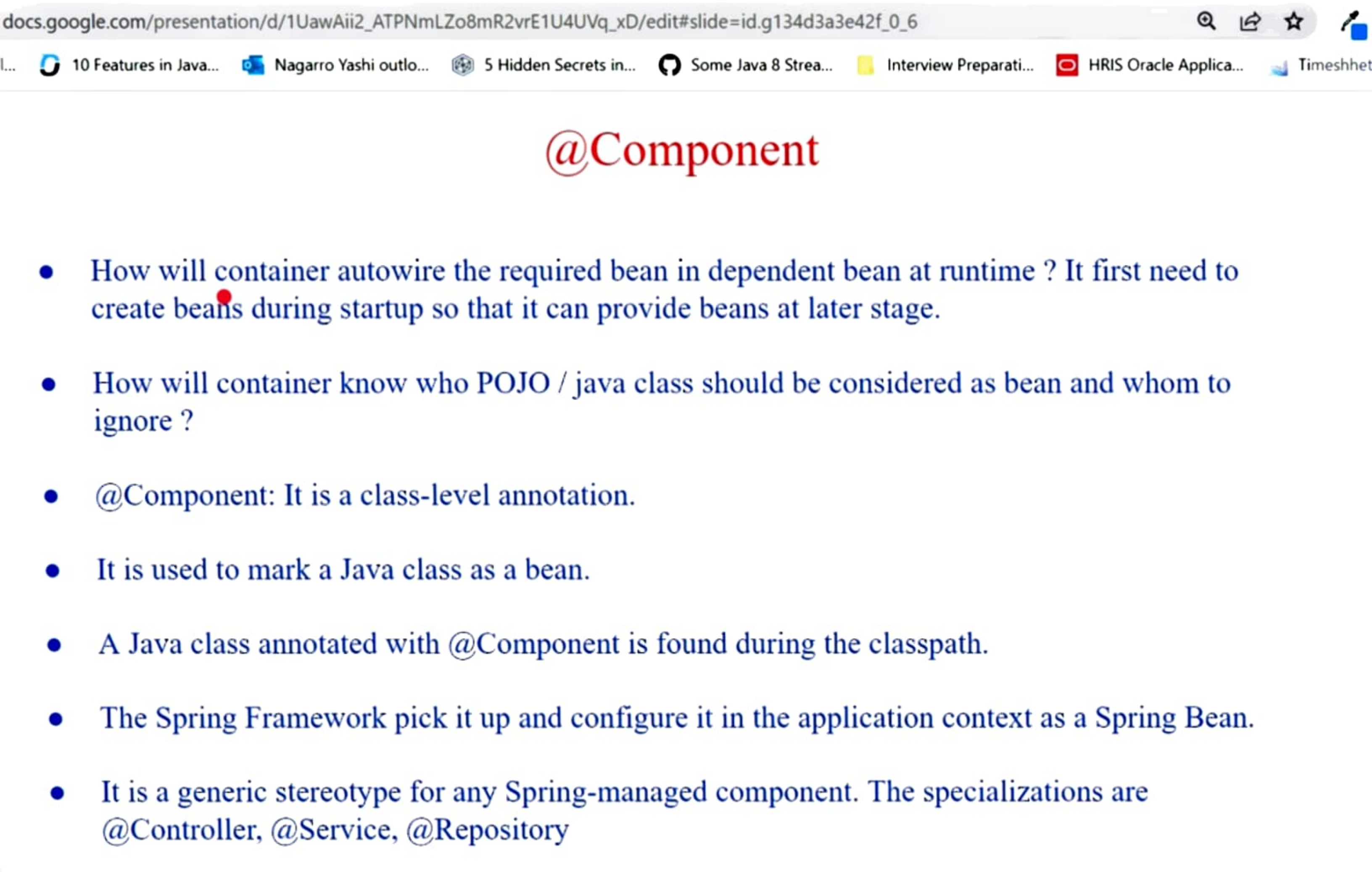
10 Features in Java... Nagarro Yashi outlo... 5 Hidden Secrets in... Some Java 8 Strea... Interview Preparati... HRIS Oracle Applica... Timeshhet

@Controller

- The @Controller is a class-level annotation.
- It is a **specialization** of @Component.
- It marks a class as a **web request handler**. It is often used to serve web pages. It is used in conjunction with @RequestMapping annotation.
- We also use @RestController when we need to send response in JSON format directly.
- By default, @Controller returns a string that indicates which route to redirect. Its used mostly with JSPs. It returns URL to new JSP page where it has to be redirected
- They are nothing but the specialized form of @Component annotation for certain situations. Instead of using @Component on a controller class in Spring MVC, we use @Controller, which is more readable and appropriate.

@Controller

- The @Controller is a class-level annotation.
- It is a **specialization** of @Component.
- It marks a class as a **web request handler**. It is often used to serve web pages. It is used in conjunction with @RequestMapping annotation.
- We also use @RestController when we need to send response in JSON format directly.
- By default, @Controller returns a string that indicates which route to redirect. Its used mostly with JSPs. It returns URL to new JSP page where it has to be redirected
- They are nothing but the specialized form of @Component annotation for certain situations. Instead of using @Component on a controller class in Spring MVC, we use @Controller, which is more readable and appropriate.

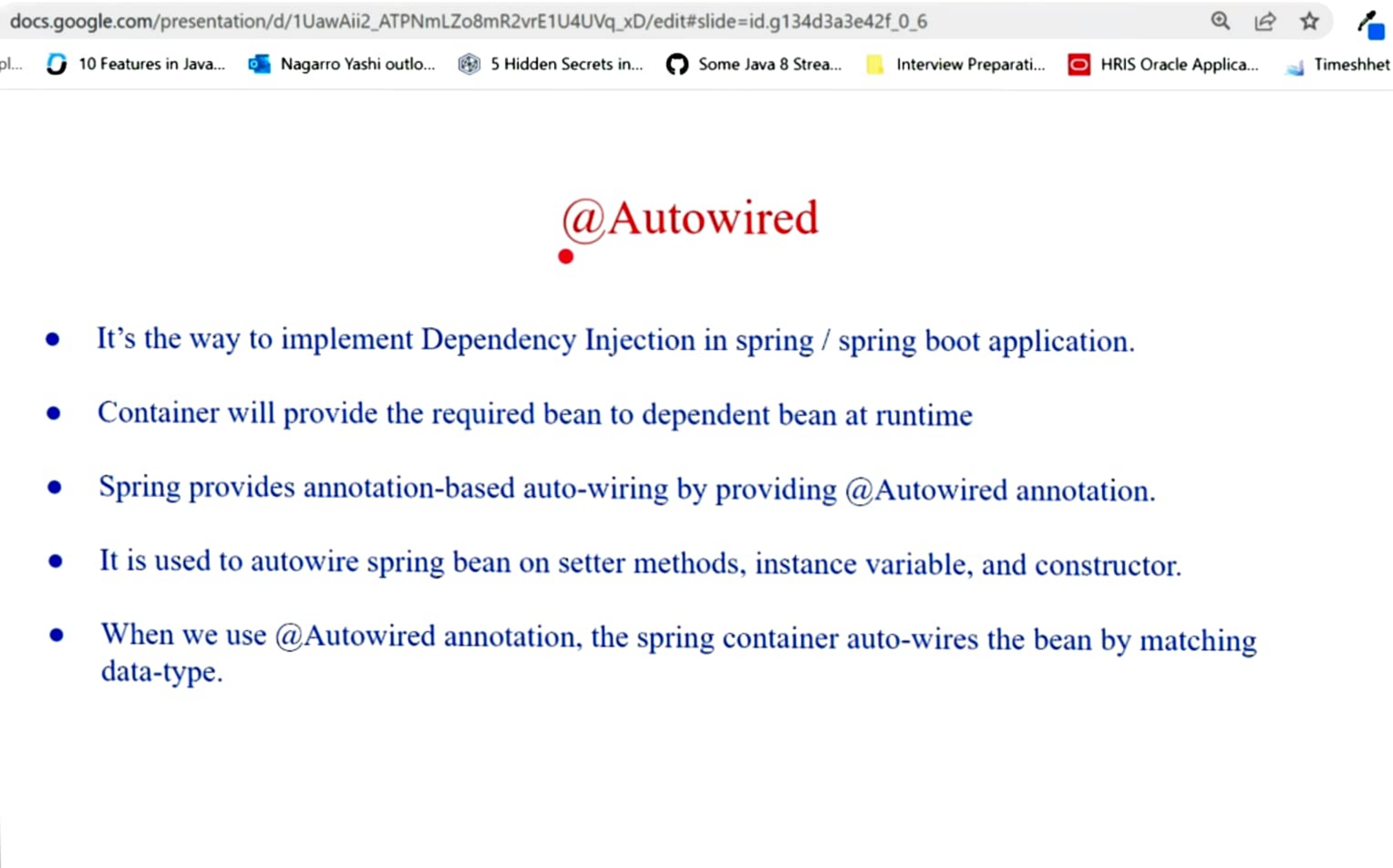


docs.google.com/presentation/d/1UawAii2_ATPNmLZo8mR2vrE1U4UVq_xD/edit#slide=id.g134d3a3e42f_0_6

l... 10 Features in Java... Nagarro Yashi outlo... 5 Hidden Secrets in... Some Java 8 Strea... Interview Preparati... HRIS Oracle Applica... Timeshhet

@Component

- How will container autowire the required bean in dependent bean at runtime ? It first need to create beans during startup so that it can provide beans at later stage.
- How will container know who POJO / java class should be considered as bean and whom to ignore ?
- @Component: It is a class-level annotation.
- It is used to mark a Java class as a bean.
- A Java class annotated with @Component is found during the classpath.
- The Spring Framework pick it up and configure it in the application context as a Spring Bean.
- It is a generic stereotype for any Spring-managed component. The specializations are @Controller, @Service, @Repository

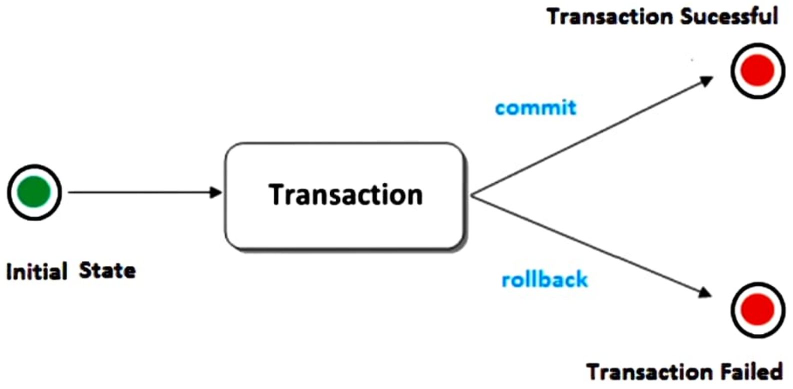


@Autowired

- It's the way to implement Dependency Injection in spring / spring boot application.
- Container will provide the required bean to dependent bean at runtime
- Spring provides annotation-based auto-wiring by providing @Autowired annotation.
- It is used to autowire spring bean on setter methods, instance variable, and constructor.
- When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.

What is Transaction?

- Every operation should be atomic in nature either do or die that means either commit or rollback.


Transaction Sucessful
commit
Initial State
Transaction
rollback
Transaction Failed