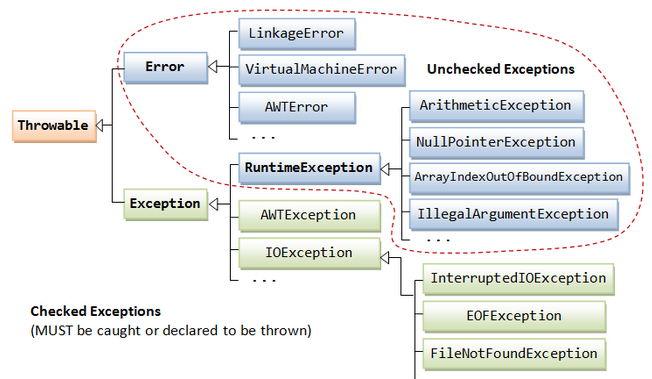
**Q)How many action instances will be created for 10 request?**

**Q) Structure of System.out.println ?**

Out (PrinttStream type) is static member of System class and println is overloaded method on it.

**Q) JSF request flow?**

**Q) Difference between error and exception ?**



***Checked exceptions*** are generally those from which a program can recover & it might be a good idea to recover from such exceptions programmatically. Examples include FileNotFoundException, ParseException, etc. A programmer is expected to check for these exceptions by using the try-catch block or throw it back to the caller

***Unchecked exceptions*** These are those exceptions that might not happen if everything is in order, but they do occur. Examples include ArrayIndexOutOfBoundException, ClassCastException, etc. Many applications will use try-catch or throws clause for RuntimeExceptions & their subclasses but from the language perspective it is not required to do so. Do note that recovery from a RuntimeException is generally possible but the guys who designed the class/exception deemed it unnecessary for the end programmer to check for such exceptions.

***Errors*** are also unchecked exception & the programmer is not required to do anything with these. In fact it is a bad idea to use a try-catch clause for Errors. Most often, recovery from an Error is not possible & the program should be allowed to terminate. Examples include OutOfMemoryError, StackOverflowError, etc.

**Q) Why use finally block?**

**Q) Why use EJB and JSf at the same time ?**

If JSF supports ajax then why use Myfaces or other faces implementation ?

**Q) Filters ?**

**In web.xml**

**<filter>**

<filter-name>HeaderFilter</filter-name>

<filter-class>com.example.HeaderFilter</filter-class>

**</filter>**

**<filter-mapping>**

<filter-name>HeaderFilter</filter-name>

<servlet-name>AdminConsole</servlet-name>

**</filter-mapping>**

The Filter class defines three methods—init, doFilter and destroy—

It is possible to have more than one filter defined for a servlet, known as a *filter chain*; the order of the filter-mapping tags in the deployment descriptor defines the order of the chain for execution.

Note that the Filter interface is not part of the javax.servlet.http package, and the request and response parameters passed to doFilter are ServletRequest and ServletResponse, not the HttpServletRequest and HttpServletResponse subclasses.

Some examples of filter use include:

* **Logging all requests**—The filter logs the request and immediately passes the request to the next filter in the chain.
* **Authentication**—A stronger form than that described in [Listing 13-3](http://mmlviewer.books24x7.com/book/id_62584/viewer.asp?bookid=62584&chunkid=928783834#ch13lst03): If a user is not authenticated, the request could be redirected to a login page first.
* **Serving static content**—Images or other content such as JavaScript or CSS files could be loaded without the servlet needing to be called.

**TOMCAT'S COMPONENTS**

Tomcat is built from several components, including one for the servlet container, and one for processing JSPs. The servlet container component is called *Catalina*, and the JSP component is called *Jasper.*

**Q) In Spring : How do you define an application context in Java code?**

Introduced in Spring 3.0, it is now possible to have Spring scan the classpath for a configuration.

@Configuration

public class SpringJavaConfiguration {

@Bean

public Dictionary dictionary() throws IOException {

return new FileDictionary("/usr/share/dict/words");

}

@Bean

public SpellCheckApplication spellCheckApplication() throws IOException {

return new SpellCheckApplication(dictionary());

}

}

to initialize the application context.

final ApplicationContext context =

new AnnotationConfigApplicationContext(SpringJavaConfiguration.class);

or

<context:component-scan

base-package="com.wiley.javainterviewsexposed.chapter16"/>

**What is autowiring?**

*Autowiring* is the process of letting the application context figure out the dependencies for you.

**Hibernate**  
<Entity>.hbm.xml // mapping file.

hibernate.cfg.xml. //config file