12 factor Principles

Monday, 29 July 2024 9:31 PM

1) Codebase- Repository should not be shared with others microservic
2) Dependencies:twelve-factor app should always explicitly d dependencies. We should do this using a dependency declaration Object Model) will be used to active as part of Maven.
3) Configurations: Basically managing the database URL, user sensitive details use environment variables.
4) Backing Services : We need to define the repository and if c we need to change the driver file

```
@Repository
public interface MovieRepository extends JpaRe
Long> {
}
```

As we can see, this is not dependent on MySQL directly. Sp MySQL driver on the classpath and provides a MySQL-spec this interface dynamically. Moreover, it pulls other details fr directly.

5) Build, Release and Run: The twelve-factor methodology str process of converting codebase into a running applicationa

   Maven compile, build and run.

6) Port Binding : We should be able to run the application indep deploy the application on tomcat web server.

7) Concurrency : The twelve-factor methodology suggests app for scaling. What this effectively means is that applications shou distribute workload across multiple processes. Individual proces to leverage a concurrency model like *Thread* internally.

A Java application, when launched gets a single process which is

ces
eclare all its
n manifest. POM(Project

name ,pwd and other

database changes then

epository<Movie,

ring detects the
cific implementation of
om configurations

ictly separates the
s three distinct stages:

pendently not like

os to rely on processes
uld be designed to
sses are, however, free

bound to the underlying

JVM. What we effectively need is a way to launch multiple instanc
with intelligent load distribution between them. Since we've alread
application as a Docker container, Kubernetes is a natural choice f

Filters in Spring Security.

When implementing Spring Security with JWT (JSON Web Token) for authentication and aut
are commonly used to handle different aspects of the JWT lifecycle. Here are the primary ty
implement or configure:

1. **JWT Authentication Filter**: This filter is responsible for extracting the JWT from the HT
   setting the authentication context. Typically, this filter will:
   - Extract the JWT from the Authorization header of the request.
   - Validate the token (e.g., check its signature, expiration, etc.).
   - Parse the token to extract user details or authorities.
   - Create an Authentication object and set it in the SecurityContextHolder.
     Example: JwtAuthenticationFilter or JwtTokenFilter.
2. **JWT Authorization Filter**: Although often combined with the authentication filter, this
   authorize requests based on roles or permissions embedded in the JWT. It ensures tha
   roles or authorities to access a particular resource.
3. **JWT Token Filter**: This is a more general term and can encompass both authentication
   responsible for handling all JWT-related processing in the security filter chain.
4. **Username and Password Authentication Filter**: This filter is used to authenticate user
   password. It typically processes login requests and generates JWT tokens upon succes
   user is authenticated, a JWT is issued and returned in the response.
   Example: UsernamePasswordAuthenticationFilter (extended to support JWT issuance)
5. **JWT Token Filter for Refresh Tokens**: If you use refresh tokens in your system, you mi
   handle the issuance of new access tokens when a refresh token is presented.
6. **Exception Handling Filter**: This filter handles exceptions thrown during the authentica
   such as expired or invalid JWT tokens. It can be used to send appropriate HTTP respon
   Example: ExceptionHandlingFilter.
7. **Cors Filter**: Though not exclusive to JWT, this filter handles Cross-Origin Resource Shar
   application can handle requests from different origins. It's especially important when
   hosted on different domains.
   Example: CorsFilter.
8. **Custom Filters**: Depending on your specific requirements, you might implement additi
   various aspects of security or token management. For instance, you might create a filt
   perform additional security checks.

Example Configuration

Here is a simplified example of how you might configure a JWT authentication filter in a Spri

ces of the application
dy packaged our
or such orchestration

horization, several types of filters
pes of filters you might

TTP request, validating it, and

s filter can be used specifically to
at the user has the necessary

n and authorization filters. It's

rs based on their username and
sful authentication. After the

).
ght have a separate filter to

tion and authorization process,
ses when authentication fails.

ring (CORS) to ensure that your
your frontend and backend are

ional custom filters to handle
er to log specific events or

ng Security configuration class:

Here is a simplified example of how you might configure a JWT authentication filter in a Spr

java
Copy code
```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecur
import org.springframework.security.config.annotation.web.configuration.WebSecurityCon
import org.springframework.security.web.authentication.UsernamePasswordAuthentication
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
@Bean
  public JwtAuthenticationFilter jwtAuthenticationFilter() {
    return new JwtAuthenticationFilter();
  }
@Override
  protected void configure(HttpSecurity http) throws Exception {
    http
      .csrf().disable()
      .authorizeRequests()
        .antMatchers("/public/**").permitAll()
        .anyRequest().authenticated()
      .and()
      .addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.cl
  }
}
```
In this example, JwtAuthenticationFilter is added before UsernamePasswordAuthenticationF
before handling other types of authentication.
Each filter serves a specific purpose and may need to be tailored to fit the particular require
security model.

4o mini

ng security configuration class.

rity;
figurerAdapter;
nFilter;

ass);

Filter, so it can process JWTs

ments of your application's