

Ant style pattern

?	1 (matches single character)
*	0 (matches zero or more characters)
**	0 (matches all files / directories)

```
import static org.hamcrest.CoreMatchers.*;
import static org.junit.Assert.*;

import org.junit.Test;
import org.springframework.util.AntPathMatcher;

public class AntStylePatternMatcherUtilTest {

    @Test
    public void antStylePatternTest() {
        // double asterisks
        assertThat(true, is(checkAntPattern("/static/**",
"/static/a.jpg")));
        assertThat(true, is(checkAntPattern("/static/**",
"/static/css/a.css")));
        assertThat(true, is(checkAntPattern("/static/**",
"/static/js/a.js")));
        assertThat(true, is(checkAntPattern("/static/**",
"/static/img/a.jpg")));
        assertThat(true, is(checkAntPattern("/static/**",
"/static/a/b/c/d/e/f/g/a.jpg")));
        assertThat(true, is(checkAntPattern("/static/**", "/static")));
        assertThat(true, is(checkAntPattern("/static/**", "/static/")));

        // single asterisks
        assertThat(true, is(checkAntPattern("/static/*", "/static/a.jpg")));
        assertThat(true, is(checkAntPattern("/static/*",
"/static/namkyuProfilePicture.jpg")));

        assertThat(false, is(checkAntPattern("/static/*",
"/static/a/test.jpg")));
        assertThat(false, is(checkAntPattern("/static/*",
"/static/a/b/c/d/test.jpg")));

        assertThat(true, is(checkAntPattern("/static*/**",
"/static/test.jpg")));
        assertThat(true, is(checkAntPattern("/static*/**",
```

```

"/static1/test.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static123/test.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static-123/test.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static~!@#$$%^&*()_+}{|/test.jpg")));

    assertThat(false, is(checkAntPattern("/static*/**",
"/static12/a/test.jpg")));
    assertThat(false, is(checkAntPattern("/static*/**",
"/static12/a/b/test.jpg")));

    // double and single combine
    assertThat(true, is(checkAntPattern("/static*/**",
"/static/a.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static1/a.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static/a/a.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static/a/b/a.jpg")));
    assertThat(true, is(checkAntPattern("/static*/**",
"/static/a/b/c/a.jpg")));

    assertThat(true, is(checkAntPattern("**/static/**",
"a/static/a/b/c/a.jpg")));
    assertThat(true, is(checkAntPattern("**/static/**",
"a/b/static/a/b/c/a.jpg")));

    // question-mark
    assertThat(true, is(checkAntPattern("/static-?/**", "/static-
a/a.jpg")));
    assertThat(true, is(checkAntPattern("/static-?/**", "/static-
a/b/c/a.jpg")));
    assertThat(true, is(checkAntPattern("/static-?/*", "/static-
a/abcd.jpg")));
    assertThat(true, is(checkAntPattern("/static-?/???.jpg", "/static-
a/abc.jpg")));

}

private boolean checkAntPattern(String pattern, String inputStr) {
    return matches(pattern, inputStr);
}

public static boolean matches(String pattern, String inputStr) {
    AntPathMatcher antPathMatcher = new AntPathMatcher();
    return antPathMatcher.match(pattern, inputStr);
}

```

```
}
```

Ref

Ant style pattern

, antpattern,

From:

<http://rwiki.repia.com/> -

. - 2023.12

Permanent link:

<http://rwiki.repia.com/doku.php?id=wiki:programming:pattern&rev=1610533569>



Last update: **2022/03/10 19:52**