Title: WEB-INF not included in WebApp using SpringBoot, Spring-MVC and Maven

Post Body:

I have a webapp using Spring-MVC built with Maven. When I generate the JAR file the app start just fine. The controller is execute but when I reach this part:

```
@RequestMapping(value = '/test-block', method = RequestMethod.GET) public ModelAndView block(Model model) {     return new Mod
```

I get this error:

```
There was an unexpected error (type=Not Found, status=404). /WEB-INF/templates/test-block.jsp
```

Note that debugging or running in the IDE works fine.

My folder:

```
src |--main    |--java       |--com.xxx // sources       webapp       |--WEB-INF          |--templates             |--*.jsp //
```

My application properties:

```
spring.mvc.view.prefix=/WEB-INF/ spring.mvc.view.suffix=.jsp
```

I checked the generated JAR file and I can't see WEB-INF anywhere.

edit:

pom file:

```
    <packaging>war</packaging>      <build>         <plugins>             <plugin>                 <groupId>org.springframewor
```

Accepted Answer:

You should create a `.war` rather than a `.jar` for a web application and you will see the `WEB-INF` folder.

Also change

`spring.mvc.view.prefix=/WEB-INF/` to `spring.mvc.view.prefix=/WEB-INF/templates` and

`ModelAndView('templates/test-block')` to `ModelAndView('test-block')` to address the 404 error.

Highest Rated Answer:

Reference the Spring Boot documentation on serving static content.

Of note:

> By default, Spring Boot serves static content from a directory called /static (or /public or /resources or /META-INF/resources) in the classpath or from the root of the ServletContext. It uses the ResourceHttpRequestHandler from Spring MVC so that you can modify that behavior by adding your own WebMvcConfigurer and overriding the addResourceHandlers method.

...

> You can also customize the static resource locations by using the spring.resources.static-locations property (replacing the default values with a list of directory locations).

...

> Do not use the src/main/webapp directory if your application is packaged as a jar. Although this directory is a common standard, it works only with war packaging, and it is silently ignored by most build tools if you generate a jar.

So, what you're seeing is the expected behavior. You can either move your templates to one of the expected locations, customize the default locations, or use war packaging.