

# Tomcat servlet container

Regular meeting for get knowledge and tech in  
EDUC

- контейнер сервлетов" - что и зачем
- Apache Tomcat - обзор и другие реализации
- Где взять
- Как запустить
- Как проверить
- Создание Приложения
- Деплой WAR
- Проверка

Контейнер сервлетов — программа, представляющая собой сервер, который занимается системной поддержкой сервлетов и обеспечивает их жизненный цикл в соответствии с правилами.

Servlet — в первую очередь это простой Java интерфейс, реализация которого расширяет функциональные возможности сервера.

контейнер сервлетов управляет четырьмя фазами жизненного цикла сервлета:

# ЖИЗНЕННЫЙ ЦИКЛ

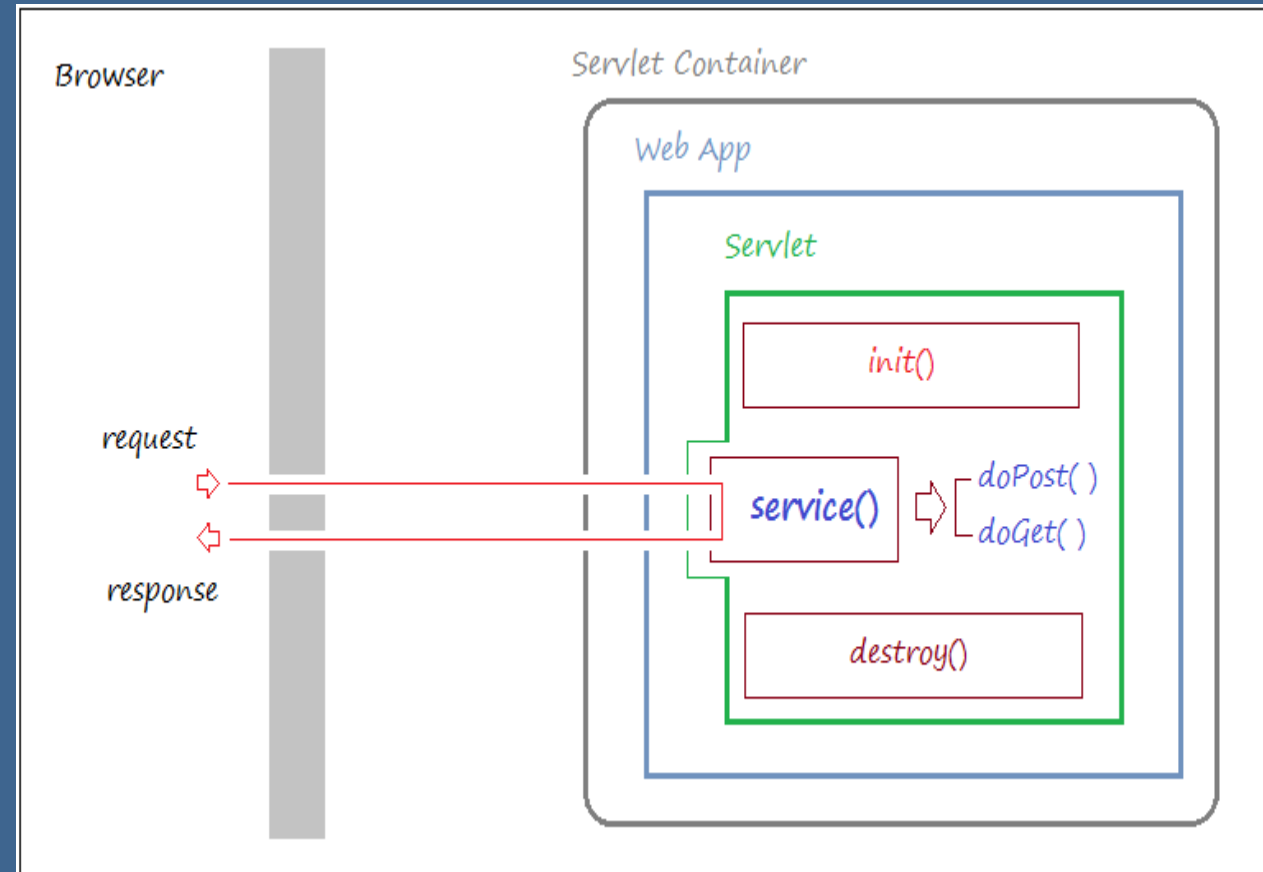
Загрузка класса сервлета — когда контейнер получает запрос для сервлета, то происходит загрузка класса сервлета в память и вызов конструктора без параметров.

Инициализация класса сервлета — это и есть момент когда код преобразуется из обычного класса в сервлет. путем вызова `init()` метода.

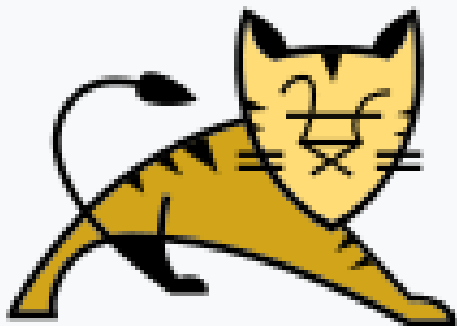
Обработка запросов — после инициализации сервлет готов к обработке запросов.

Удаление — когда контейнер останавливается или останавливается приложение, то контейнер сервлетов уничтожает классы сервлетов путем вызова `destroy()` метода.

- - пользователь отправляет запрос
- - Servlet создается в момент получения первого запроса, одновременно вызывается метод `init()`
- - Servlet вызывает метод `service()`
- - `service()` вызывает один из двух методов `doGet()` или `doPost()`, которые вам нужно переопределить и обработать эти методы.
- - Метод `destroy()` используется для разрушения servle

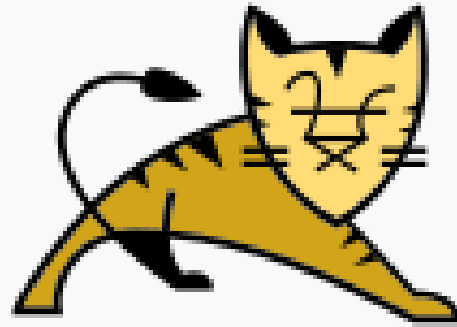


## Apache Tomcat



- Tomcat (в старых версиях — Catalina) — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation.
- Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java.
- Известные реализации: Apache Tomcat, Jetty, JBoss, GlassFish, IBM WebSphere, Oracle Weblogic.

## Apache Tomcat

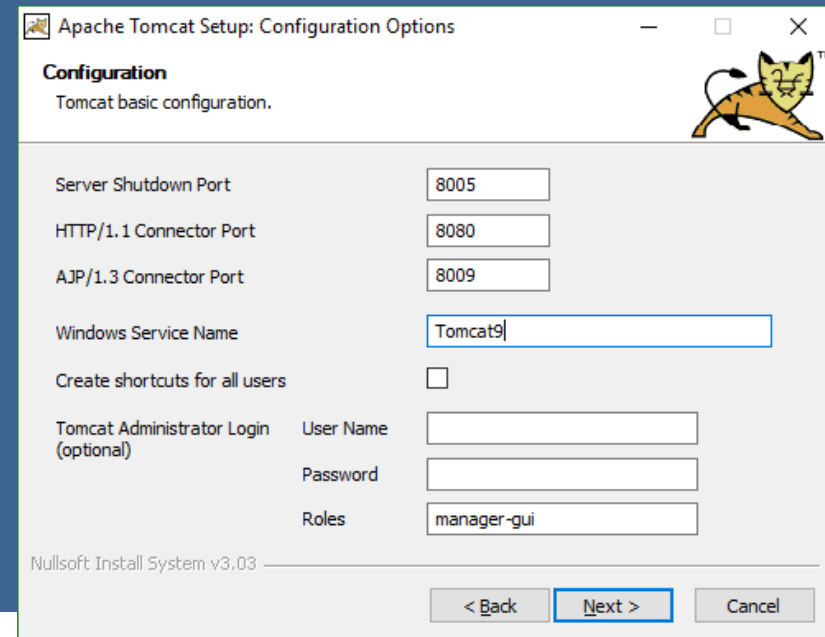
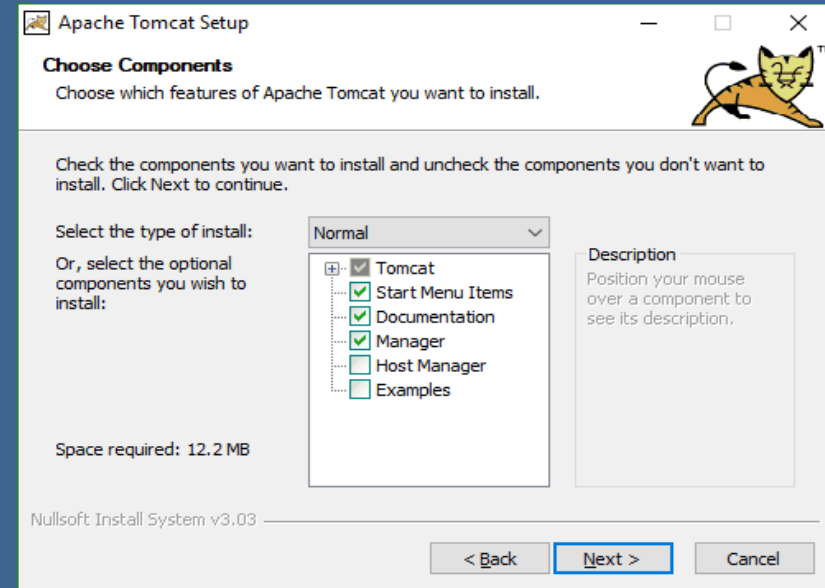
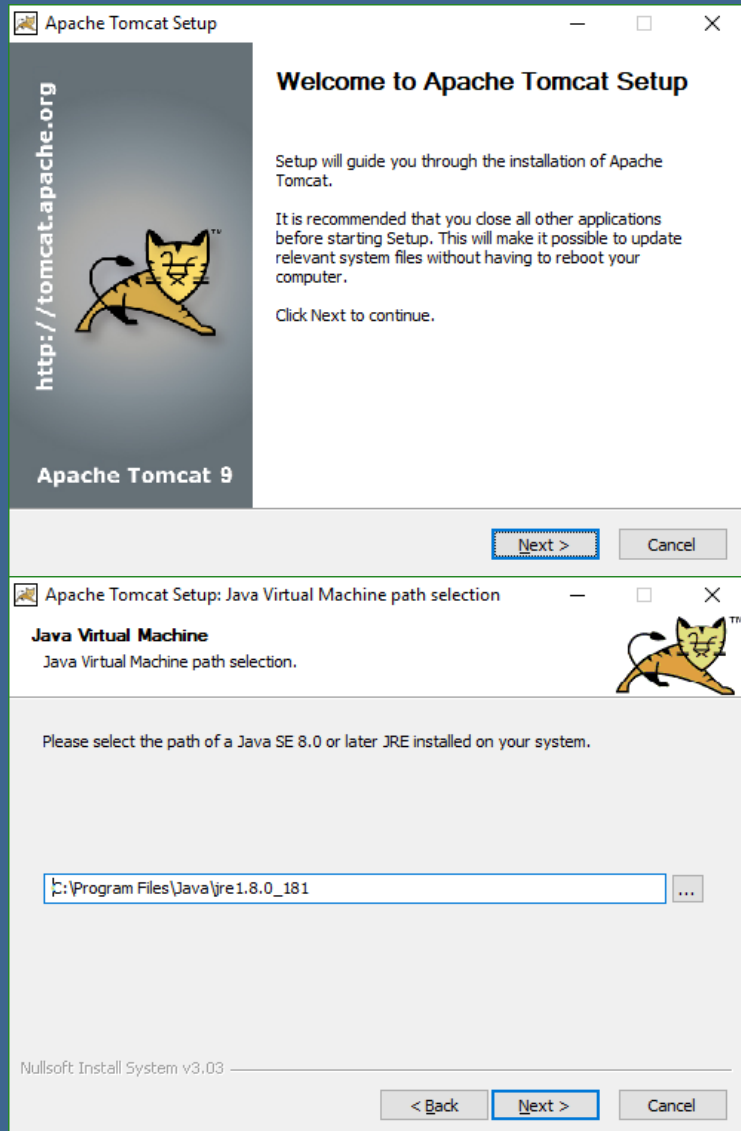


- 1 Переходим на сайт <https://tomcat.apache.org/>
- 2 Скачиваем установочный файл.



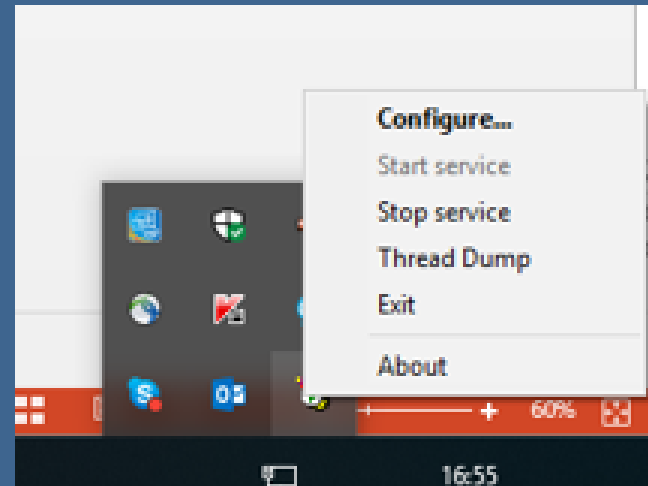
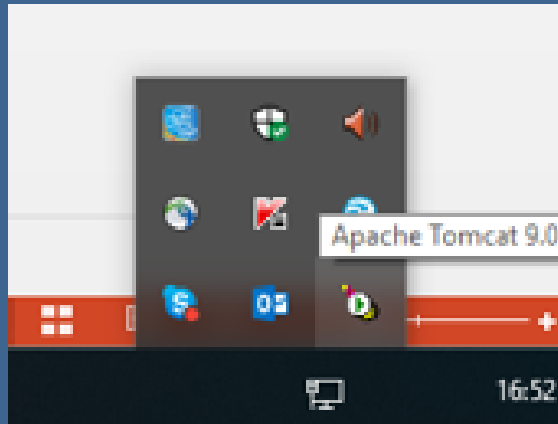
apache-tomcat-9.0.12.exe

### • 3 Устанавливаем Tomcat.



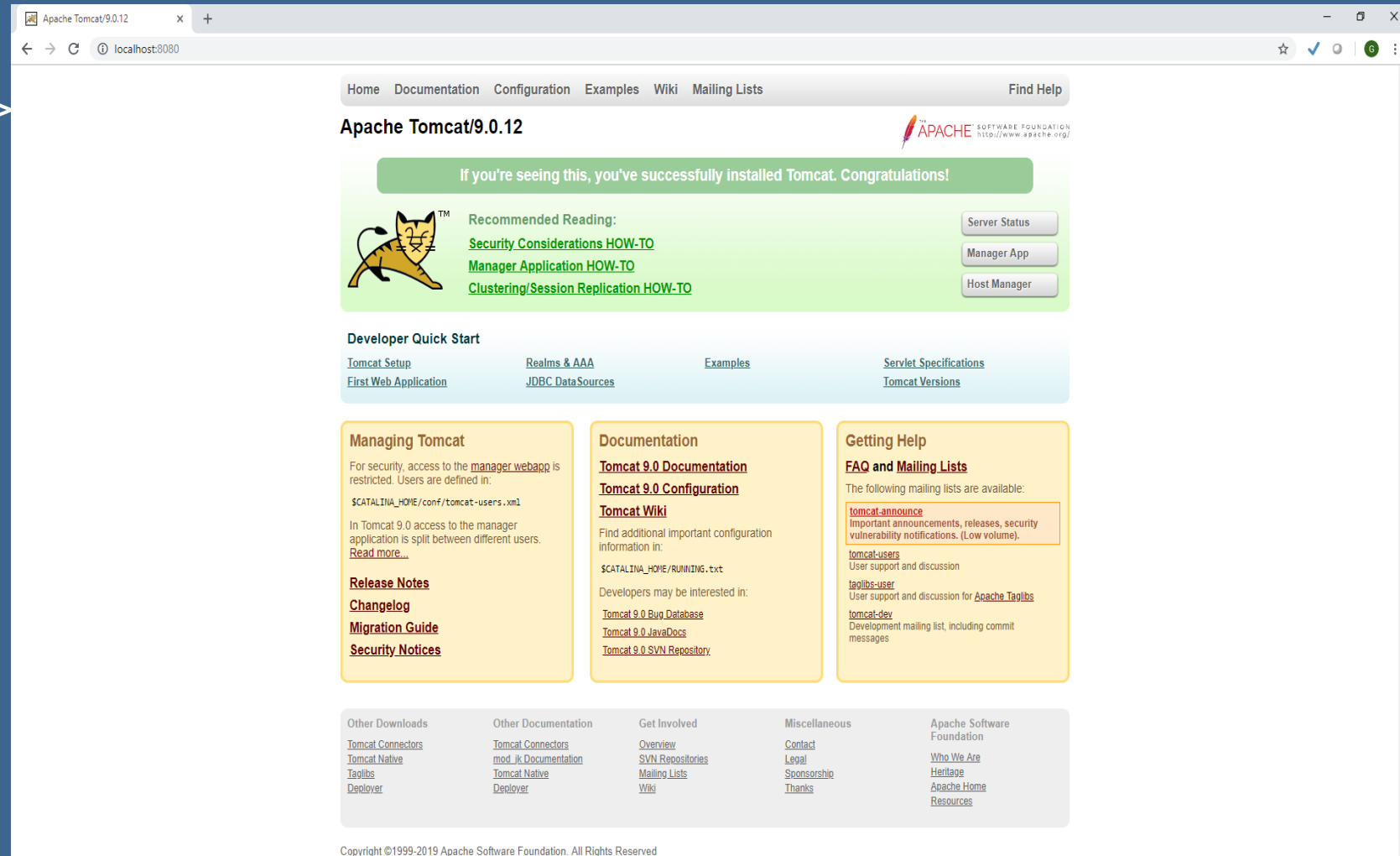


- 4 После этого в трее должен появиться значок запущенного сервиса.

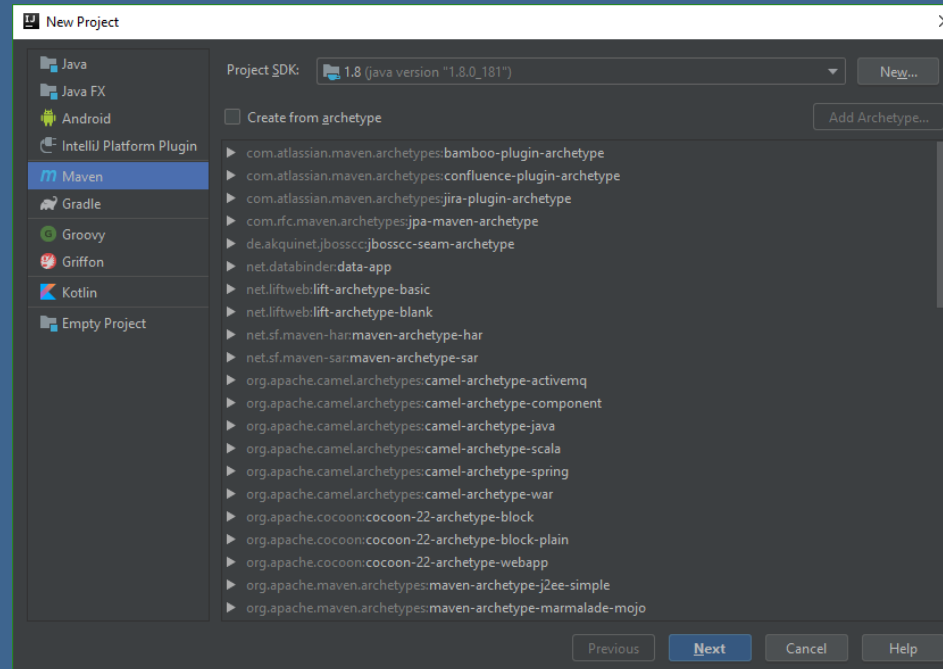


- Также через этот значок можно останавливать / запускать

- 5 Перейдите по адресу
- <http://localhost:8080/>.
- Если вы видите это =>
- то всё хорошо 😊

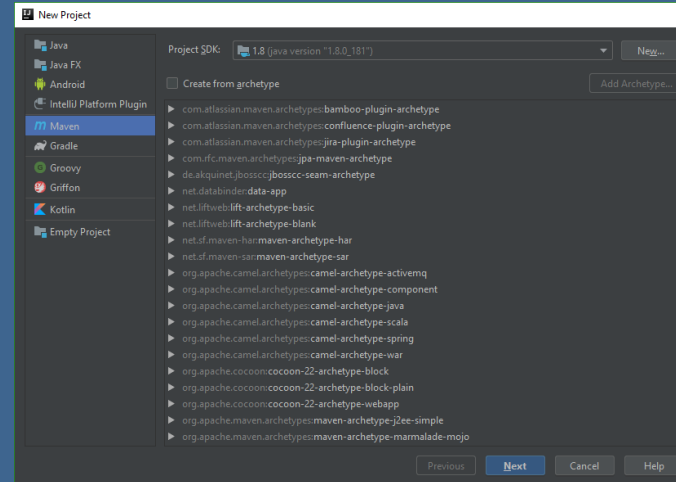
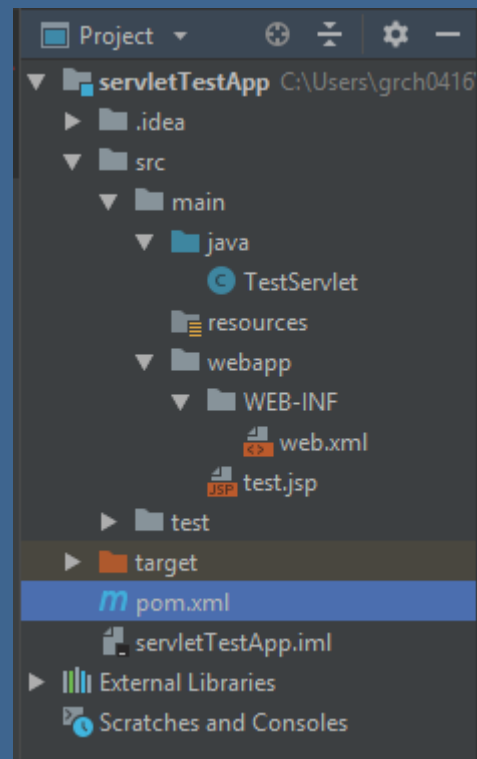


- Пример простого проекта
- Создать Maven Project



- Пример простого проекта

## Создать Maven Project



## Структура проекта

## POM.XML

POM (Project Object Model) является базовым модулем Maven. Всегда хранится в базовой директории проекта и называется pom.xml. Содержит информацию о проекте и различных деталях конфигурации, которые используются Maven для создания проекта.

- **Зависимости** — это те библиотеки, которые непосредственно используются в вашем проекте для компиляции кода или его тестирования. Например чтобы подключить Spring Framework необходимо определить следующую зависимость:

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring</artifactId>
    <version>2.5.5</version>
  </dependency>
</dependencies>
```

- **Плагины** - используются самим Maven'ом при сборке проекта или для каких-то других целей
- **профиль создания**
- **версия проекта**

# POM.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>servletTest</groupId>
  <artifactId>st</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <dependencies>

  <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
  </dependency>

  </dependencies>

</project>
```

```

TestServlet.java ×
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class TestServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType( "text/html" );
        PrintWriter out = response.getWriter();
        out.println( "<html><head>" );
        out.println( "<title>A Sample Servlet!</title>" );
        out.println( "</head>" );
        out.println( "<body>" );
        out.println( "<form method='get' action='/st/test'>" );
        out.println( "<input type='text' name='a' />" );
        out.println( "<input type='text' name='b' />" );
        out.println("<input type='submit' value='Show text' />");
        out.println("</form>");

        out.println( "<p>Param A " + request.getParameter( "a" ) + "</p>" );
        out.println( "<p>Param B " + request.getParameter( "b" ) + "</p>" );
        out.println( "</body></html>" );
        out.close();
    }
}

```

**GET**- лучше использовать в случаях:

- когда это безопасно
- дебаг

чтобы позволить человеку вызвать действие - например получить определенную страницу в определенном виде ( сортировка, строка поиска и т.п. ).

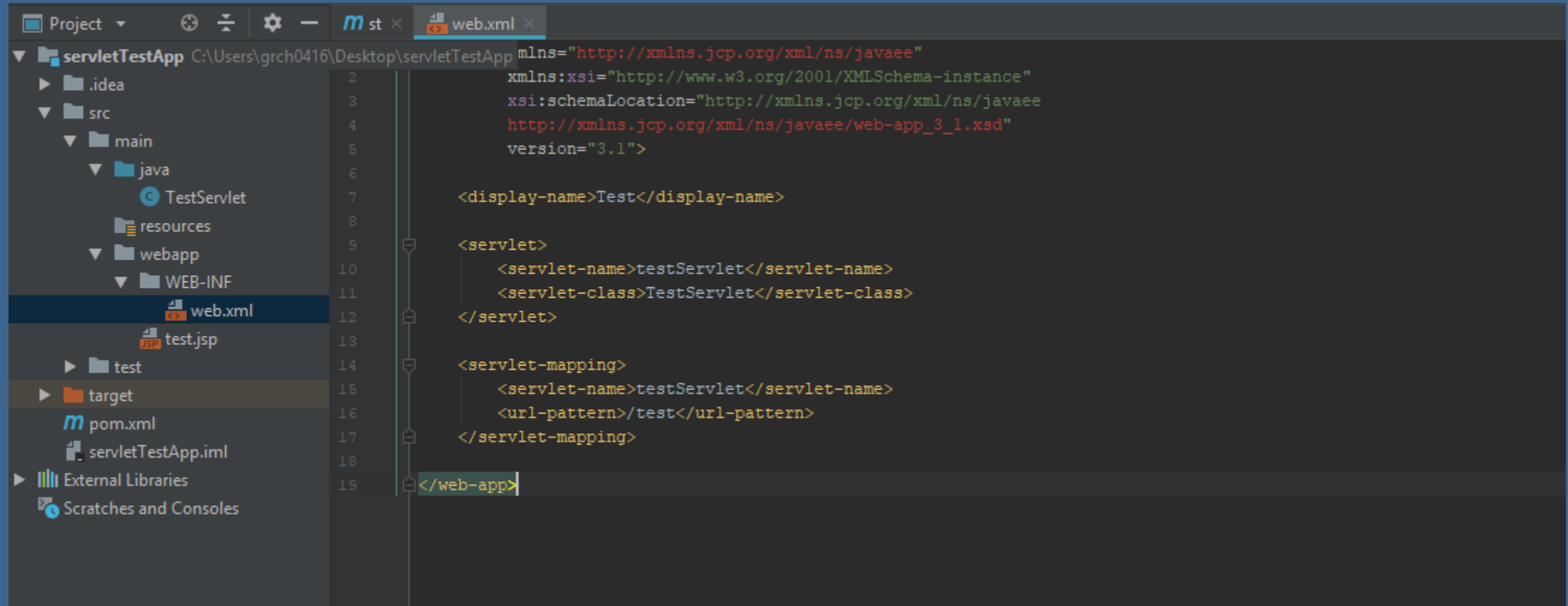
**POST** - не имеет ограничение по размеру и более безопасный, использовать лучше для отправки данных:

- которые не влияют на отображение страницы, в том плане что эти данные влияют только на результат выполнения скрипта ( логины, пароли, номера кредиток, сообщения и т.п. ).
- выполнения таких действий как создание, редактирование и удаление, потому что злоумышленник не может видеть действие POST в адресной строке браузера.

Также у HTTP существуют еще методы запросов  
HEAD, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH



# WEB.XML



```
1<?xml version="1.0" encoding="UTF-8"?>
2xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6version="3.1">
7
8  <display-name>Test</display-name>
9
10  <servlet>
11    <servlet-name>testServlet</servlet-name>
12    <servlet-class>TestServlet</servlet-class>
13  </servlet>
14
15  <servlet-mapping>
16    <servlet-name>testServlet</servlet-name>
17    <url-pattern>/test</url-pattern>
18  </servlet-mapping>
19</web-app>
```

# Создание WAR - Web Archive или Web Application Resource

The screenshot shows an IDE interface with the following components:

- Project Explorer (Left):** Displays the project structure for `servletTestApp`. The `target` directory is highlighted, indicating the location where the WAR file will be generated.
- Main Editor:** Shows the `TestServlet.java` file. The code is as follows:

```
1 import javax.servlet.ServletException;
2 import javax.servlet.http.HttpServlet;
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5 import java.io.IOException;
6 import java.io.PrintWriter;
7
8 public class TestServlet extends HttpServlet {
9
10     @Override
11     protected void doGet(HttpServletRequest request, HttpServletResponse response)
12         throws ServletException, IOException {
13         response.setContentType("text/html");
14         PrintWriter out = response.getWriter();
15         out.println("<html><head>");
16         out.println("<title>A Sample Servlet!</title>");
17         out.println("</head>");
18         out.println("<body>");
19         out.println("<form method='get' action='/st/test'>");
20         out.println("<input type='text' name='a' />");
21         out.println("<input type='text' name='b' />");
22         out.println("<input type='submit' value='Show text' />");
23         out.println("</form>");
24
25         out.println("<p>Param A " + request.getParameter("a") + "</p>");
26         out.println("<p>Param B " + request.getParameter("b") + "</p>");
27         out.println("</body></html>");
28         out.close();
29     }
30 }
31
```
- Maven Projects (Right):** Shows the Maven lifecycle. The `install` goal is highlighted, which is the step that generates the WAR file.

# Создание WAR - Web Archive или Web Application Resource

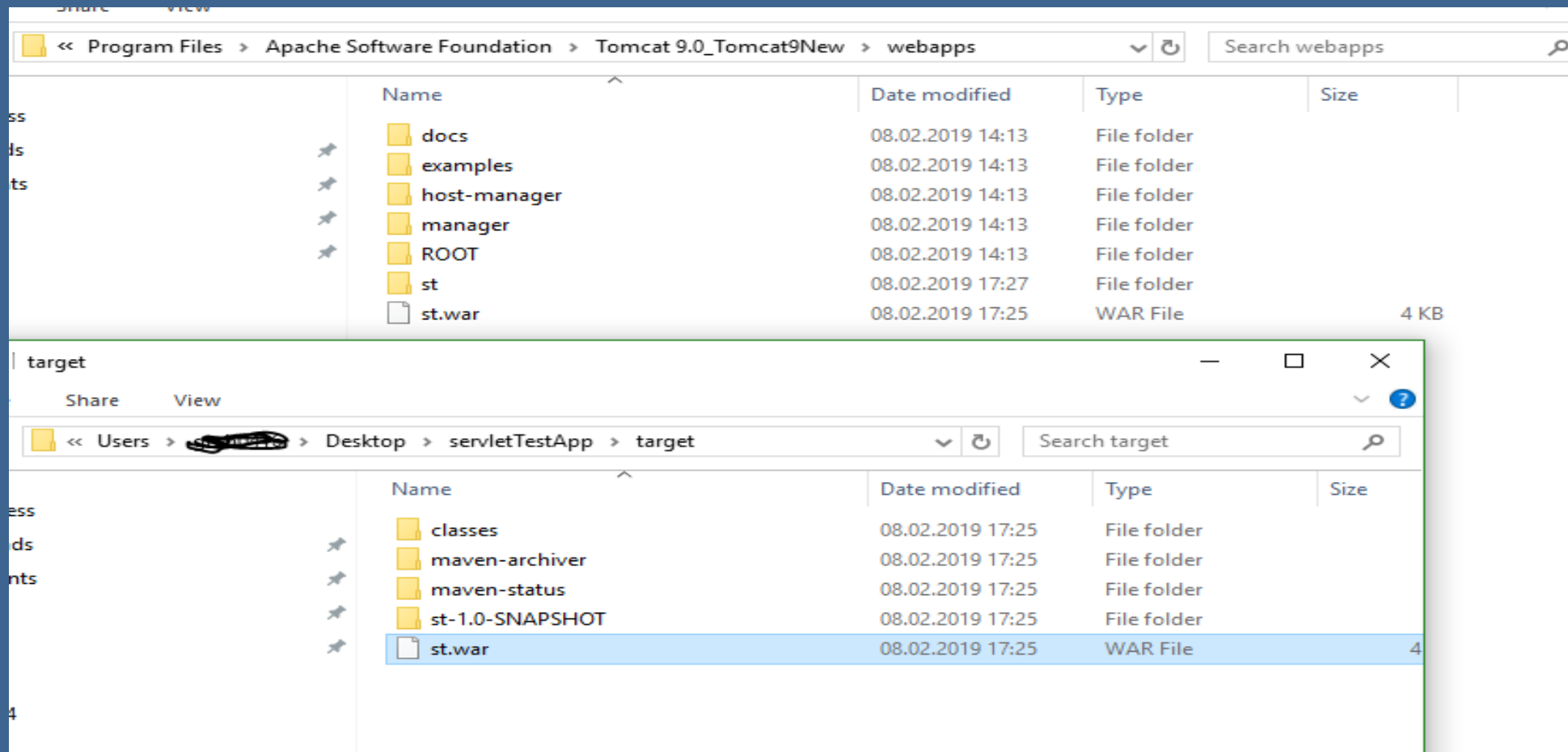
The screenshot displays the IntelliJ IDEA IDE interface for a project named `servletTestApp`. The main editor shows the `TestServlet.java` file, which implements the `HttpServlet` interface. The code includes imports for `javax.servlet.ServletException`, `javax.servlet.http.HttpServlet`, `javax.servlet.http.HttpServletRequest`, `javax.servlet.http.HttpServletResponse`, `java.io.IOException`, and `java.io.PrintWriter`. The `doGet` method is overridden to send an HTML response with a form.

The left sidebar shows the project structure, with the `target` directory expanded to reveal the `st-1.0-SNAPSHOT.war` file. The right sidebar shows the Maven Projects view, with the `install` goal selected under the `Lifecycle` tab.

The bottom panel shows the Run configuration and the output log. The Run configuration is set to `st [clean,install]`. The output log shows the following messages:

```
[INFO] --- maven-install-plugin:2.4:install (default-install) @ st ---
[INFO] Installing C:\Users\grch0416\Desktop\servletTestApp\target\st-1.0-SNAPSHOT.war to C:\Users\grch0416\.m2\repository\servletTest\st\1.0-SNAPSHOT\st-1.0-SNAPSHOT.war
[INFO] Installing C:\Users\grch0416\Desktop\servletTestApp\pom.xml to C:\Users\grch0416\.m2\repository\servletTest\st\1.0-SNAPSHOT\st-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.603 s
[INFO] Finished at: 2019-02-08T17:25:22+04:00
[INFO] Final Memory: 13M/31M
[INFO] -----
Process finished with exit code 0
```

## Деплой WAR Web Archive или Web Application Resource



- 1 скопировать WSR архив
- 2 перезапустить сервер



## Tomcat Web Application Manager

Message: OK

### Manager

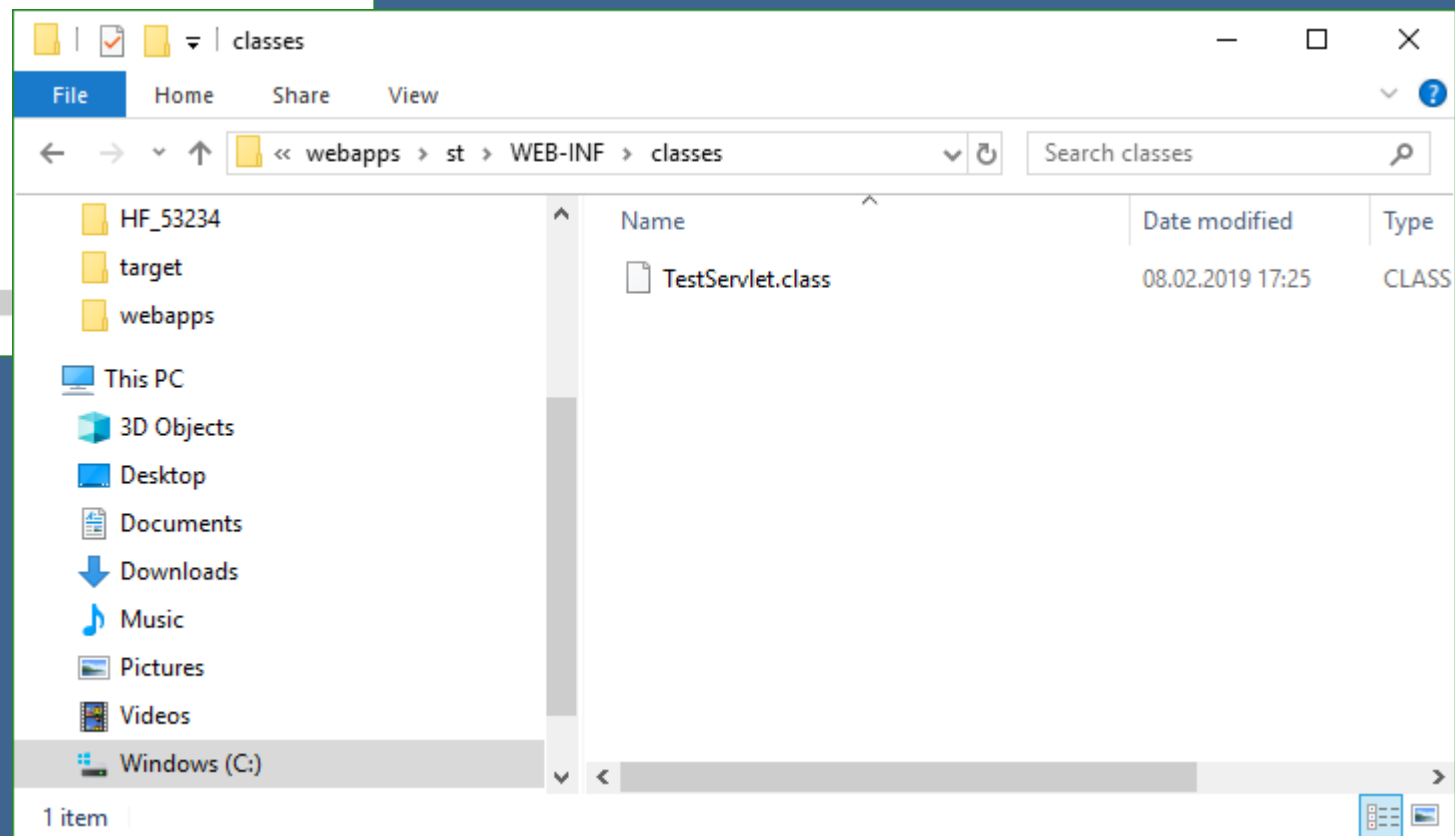
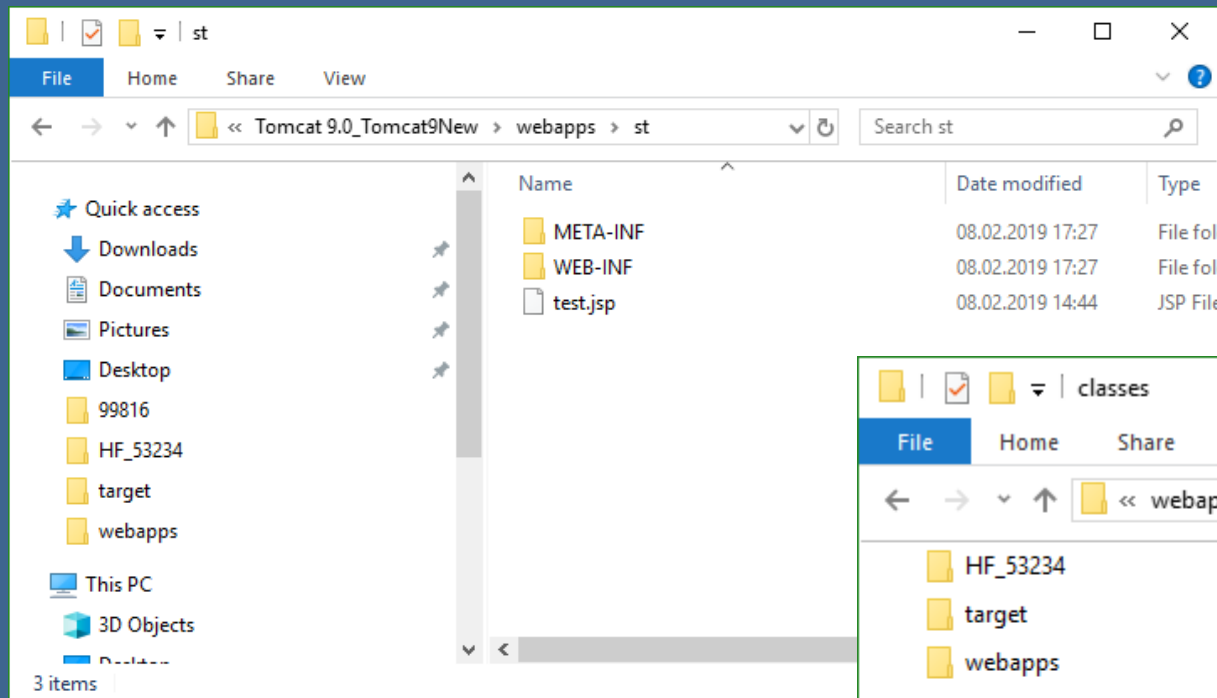
[List Applications](#) [HTML Manager Help](#) [Manager Help](#) [Server Status](#)

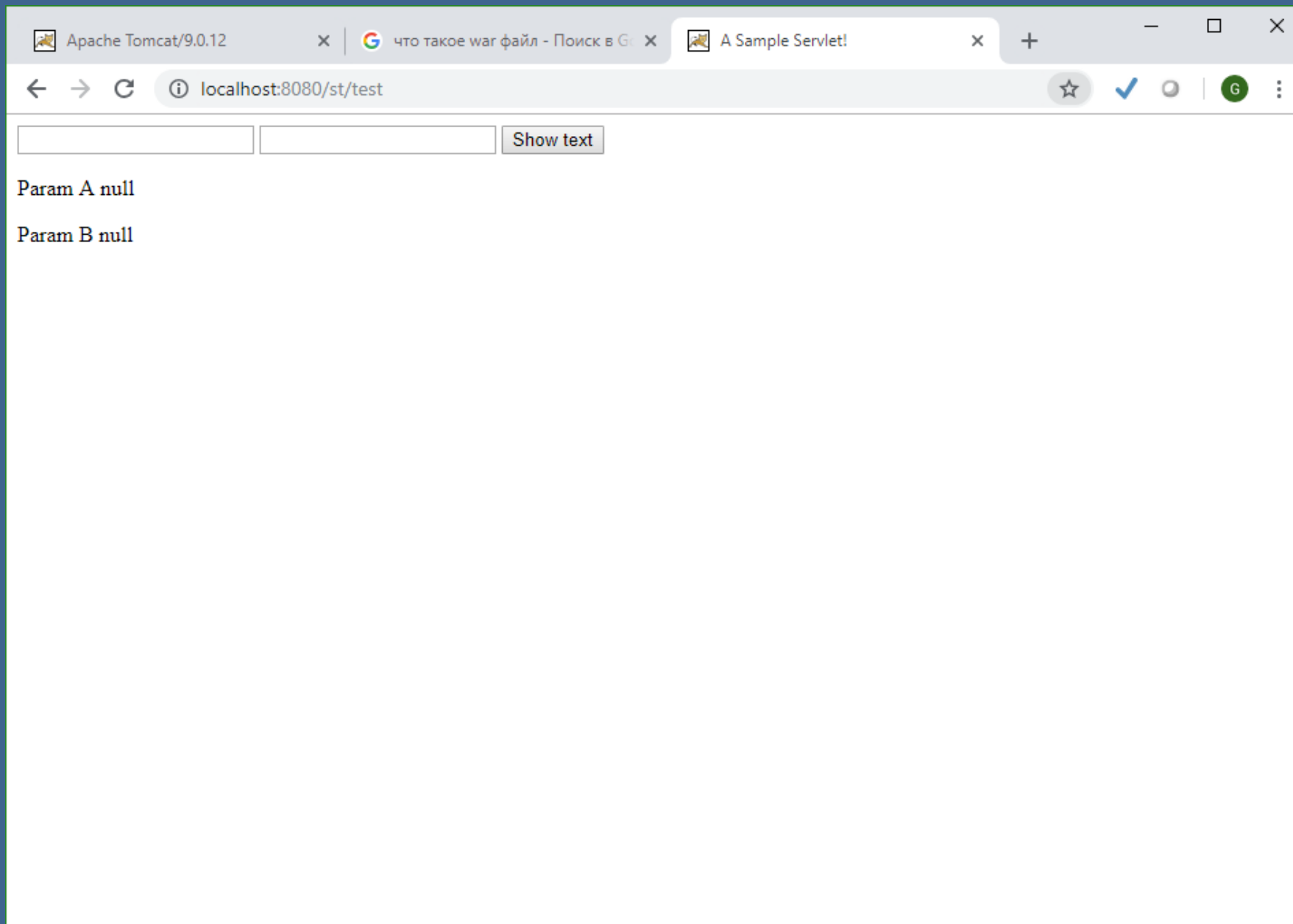
### Applications

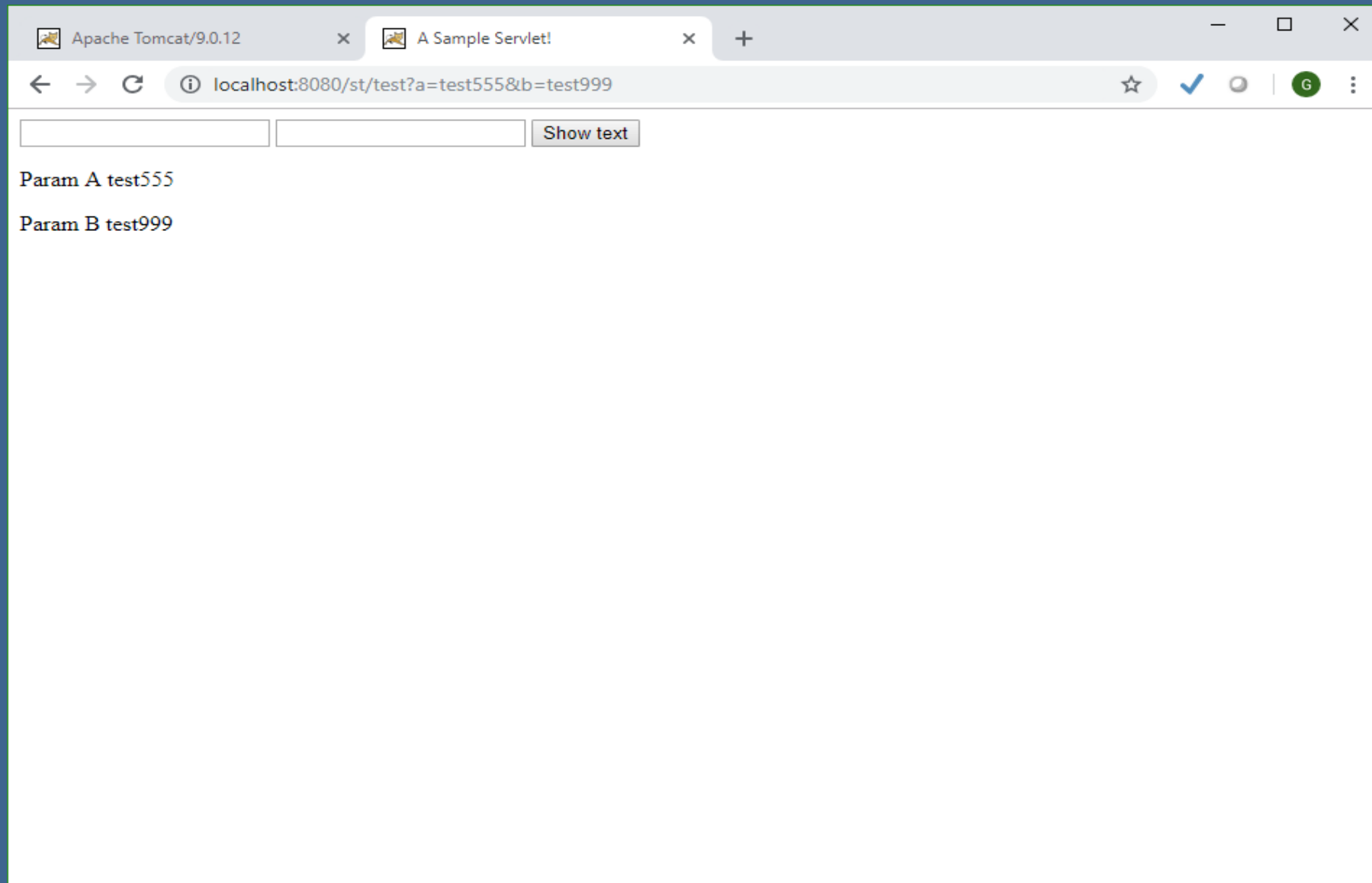
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/test	None specified	Test	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

### Deploy

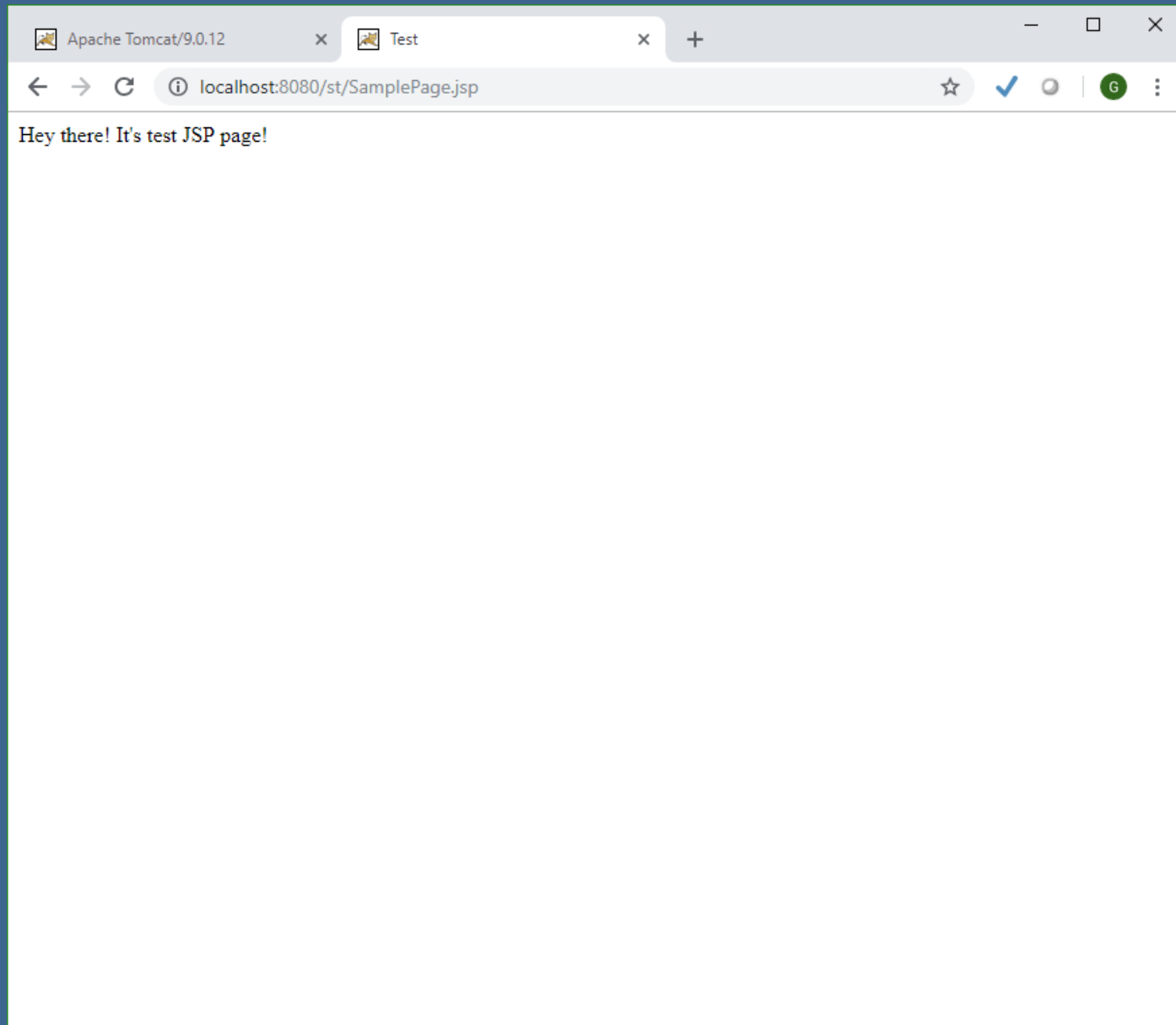
Deploy directory or WAR file located on server











# Реализация сервлетов на Spring Boot

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>springBootTest</groupId>
  <artifactId>testBoot</artifactId>
  <version>01</version>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.9.RELEASE</version>
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>

  <!--<build>-->
  <!--<plugins>-->
  <!--<plugin>-->
  <!--<groupId>org.springframework.boot</groupId>-->
  <!--<artifactId>spring-boot-maven-plugin</artifactId>-->
  <!--</plugin>-->
  <!--</plugins>-->
  <!--</build>-->
</project>
```

```
MyServlet.java × m testBoot × MainController.java × SpringBootApplication.java ×
1 package com;
2
3 import com.webapp.MyServlet;
4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6 import org.springframework.boot.web.servlet.ServletRegistrationBean;
7 import org.springframework.context.annotation.Bean;
8
9
10 @SpringBootApplication
11 public class SpringBootApplication {
12
13     // Register Servlet
14     @Bean
15     public ServletRegistrationBean servletRegistrationBean() {
16         ServletRegistrationBean bean = new ServletRegistrationBean(
17             new MyServlet(), ...urlMappings: "/myServlet");
18         return bean;
19     }
20
21     public static void main(String[] args) { SpringApplication.run(SpringBootApplication.class, args); }
22
23 }
```

springBootTest [testBoot] C:\students\springBoot

Project

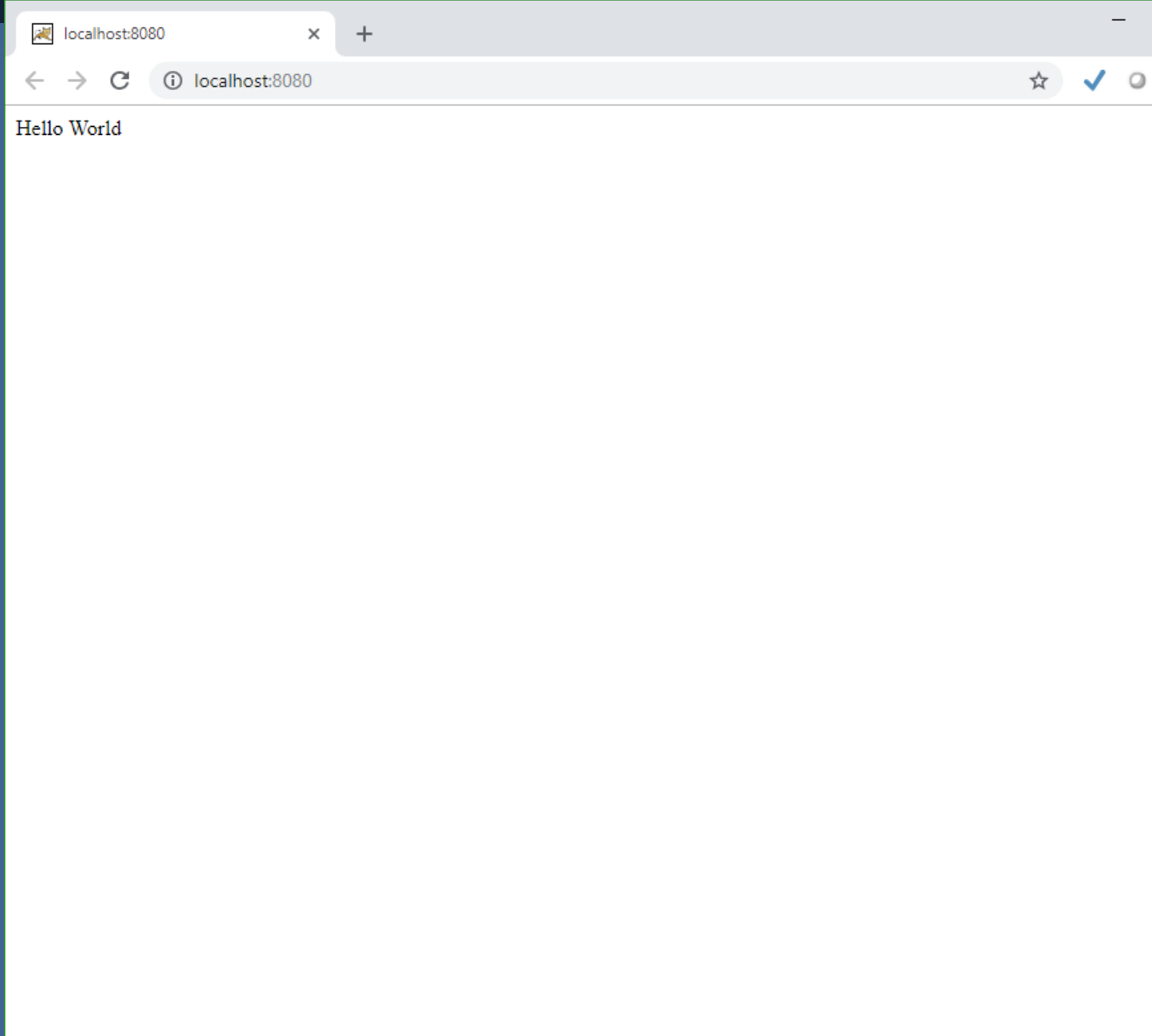
- springBootTest [testBoot] C:\students\springBoot
  - .idea
  - src
    - main
      - java
        - com
          - webapp
            - MainController
            - SpringBootApplication
          - resources
        - test
      - target
      - pom.xml
      - springBootTest.iml
      - testBoot.iml
    - External Libraries
    - Scratches and Consoles

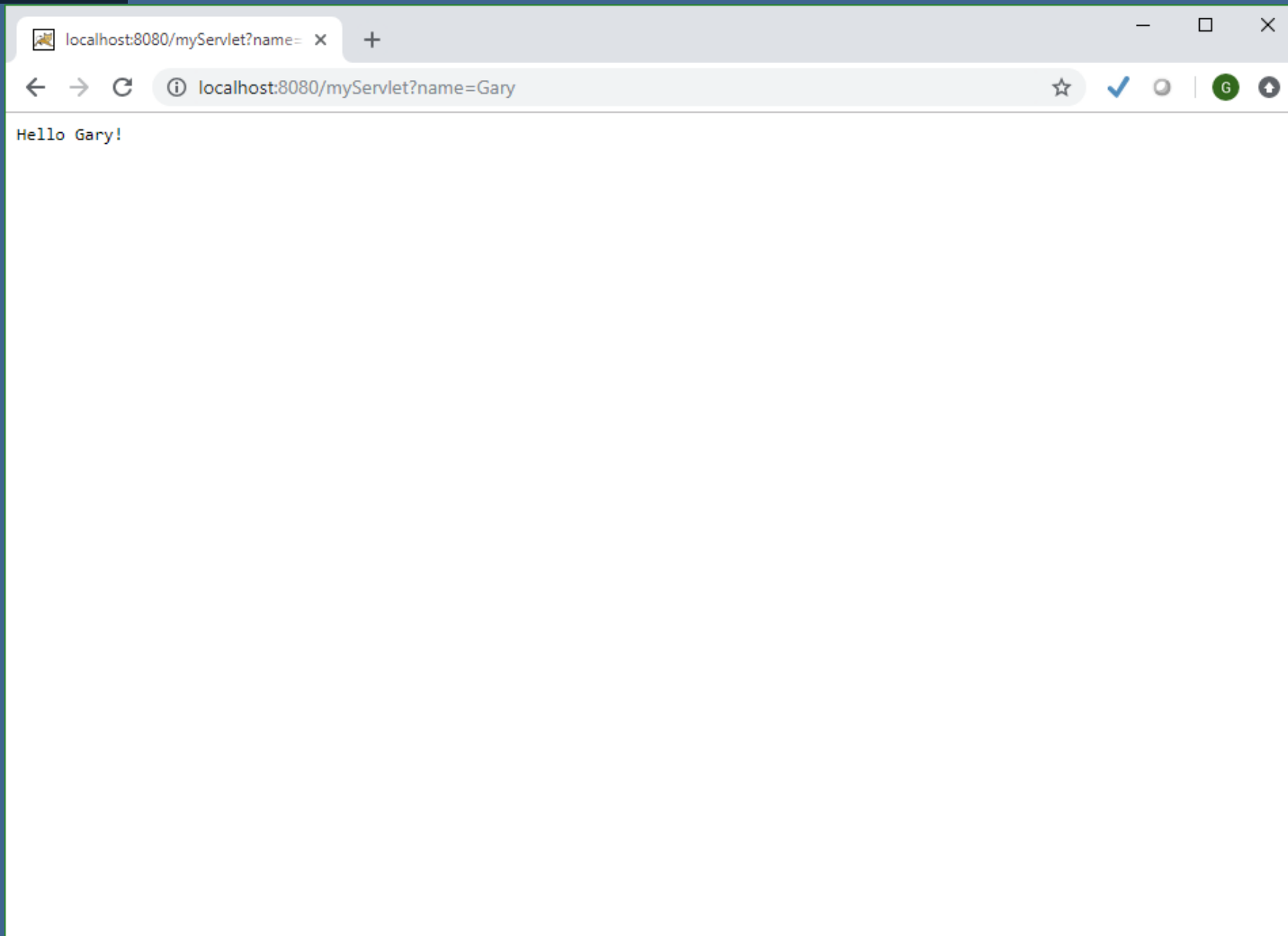
MyServlet.java x testBoot x MainController.java x SpringBootApplication.java x

```
1 package com;
2
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6
7 import java.io.*;
8
9 @RestController
10 public class MainController {
11     @RequestMapping({"/", "/index.html"})
12     public String index() throws IOException {
13         return "Hello World";
14     }
15 }
```

```
MyServlet.java × testBoot × MainController.java × SpringBootApplication.java ×
1 package com.webapp;
2
3 import javax.servlet.ServletException;
4 import javax.servlet.annotation.WebServlet;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 import java.io.IOException;
9
10 @WebServlet(urlPatterns = "/mytest")
11 public class MyServlet extends HttpServlet {
12
13     @Override
14     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
15         throws ServletException, IOException {
16         System.out.println("MyServlet's doGet() method is invoked.");
17         doAction(req, resp);
18     }
19
20     @Override
21     protected void doPost(HttpServletRequest req, HttpServletResponse resp)
22         throws ServletException, IOException {
23         System.out.println("MyServlet's doPost() method is invoked.");
24         doAction(req, resp);
25     }
26
27     private void doAction(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
28         String name = req.getParameter( "name" );
29         resp.setContentType("text/plain");
30         resp.getWriter().write( "Hello " + name + "!" );
31     }
32 }
```









springBootTest [C:\students\springBootTest] - ...src\main\java\com\webapp\MyServlet.java [testBoot] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

springBootTest \src\main\java\com\webapp\MyServlet

Project ▾  
 springBootTest [testBoot] C:\students\springBoo  
 .idea  
 src  
 main  
 java  
 com  
 webapp  
 MainController  
 SpringBootApplication  
 resources  
 test  
 target  
 pom.xml  
 springBootTest.iml  
 testBoot.iml  
 External Libraries  
 Scratches and Consoles

MyServlet.java x testBoot x MainController.java x SpringBootApplication x

```

1 package com.webapp;
2
3 import javax.servlet.ServletException;
4 import javax.servlet.annotation.WebServlet;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 import java.io.IOException;
9
10 @WebServlet(urlPatterns = "/mytest")
11 public class MyServlet extends HttpServlet {
12
13     @Override
14     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
15         throws ServletException, IOException {
16         System.out.println("MyServlet's doGet() method is invoked.");
17         doAction(req, resp);
18     }
19
20     @Override
21     protected void doPost(HttpServletRequest req, HttpServletResponse resp)
22         throws ServletException, IOException {
23         System.out.println("MyServlet's doPost() method is invoked.");
24         doAction(req, resp);
25     }
26
27     @Override
28     private void doAction(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
29         String name = req.getParameter("name");
30         resp.setContentType("text/plain");
31         resp.getWriter().write("Hello " + name + "!");
32     }
33 }

```

Maven Projects  
 testBoot  
 Lifecycle  
 Plugins  
 Dependencies

Run: SpringBootApplication x

2019-02-15 13:42:12.026 INFO 19664 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "[[/ | /index.html]]" onto public java.lang.String com.MainController.index() throws java.io.IOException  
 2019-02-15 13:42:12.030 INFO 19664 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "[[/error]]" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.BasicErrorController.  
 2019-02-15 13:42:12.030 INFO 19664 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "[[/error],produces=[text/html]]" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.  
 2019-02-15 13:42:12.103 INFO 19664 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  
 2019-02-15 13:42:12.103 INFO 19664 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  
 2019-02-15 13:42:12.191 INFO 19664 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  
 2019-02-15 13:42:12.633 INFO 19664 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup  
 2019-02-15 13:42:12.776 INFO 19664 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)  
 2019-02-15 13:42:12.781 INFO 19664 --- [main] com.SpringBootApplication : Started SpringBootApplication in 21.535 seconds (JVM running for 26.308)  
 2019-02-15 13:46:04.614 INFO 19664 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'  
 2019-02-15 13:46:04.614 INFO 19664 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started  
 2019-02-15 13:46:04.638 INFO 19664 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 24 ms  
 MyServlet's doGet() method is invoked.  
 MyServlet's doGet() method is invoked.

Messages Terminal Run 4: Run TODO

Compilation completed successfully in 1 m 24 s 17 ms (7 minutes ago)

8 chars 26:26 CRLF UTF-8 Git: master