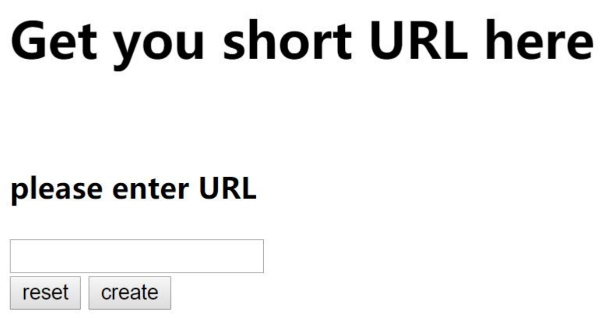
Test Task Report

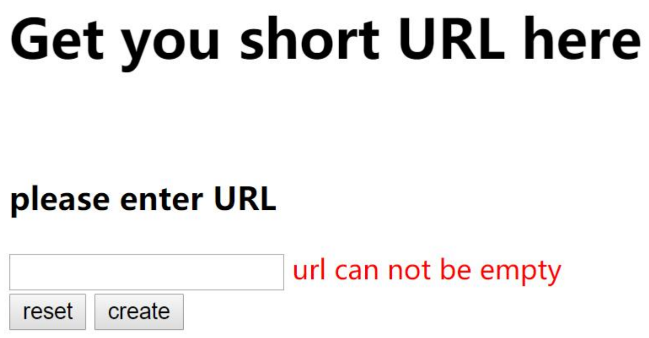
1. Result presentation

Original UI


Get you short URL here

please enter URL

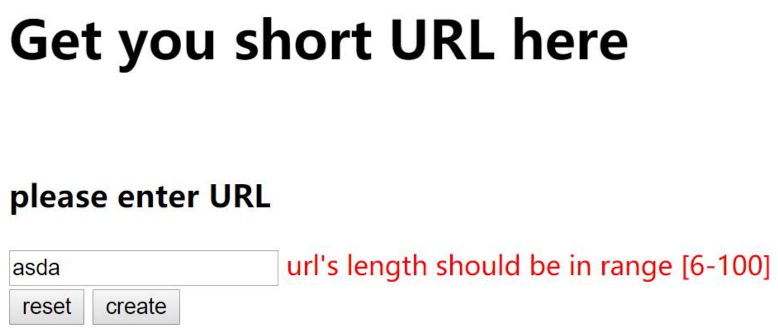
When the input is null or it only consists of spaces, the reminder comes out.


Get you short URL here

please enter URL

url can not be empty

When the length of url is not in range 6-100, the reminder comes out.

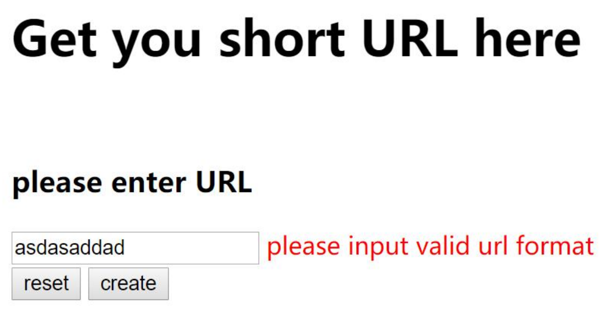

Get you short URL here

please enter URL

asda

url's length should be in range [6-100]

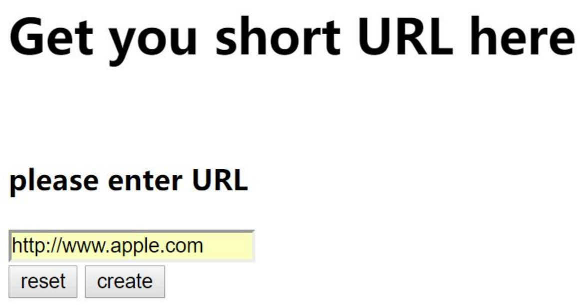
When the url does not match the regular expression, the reminder comes out.


Get you short URL here

please enter URL

asdasaddad please input valid url format

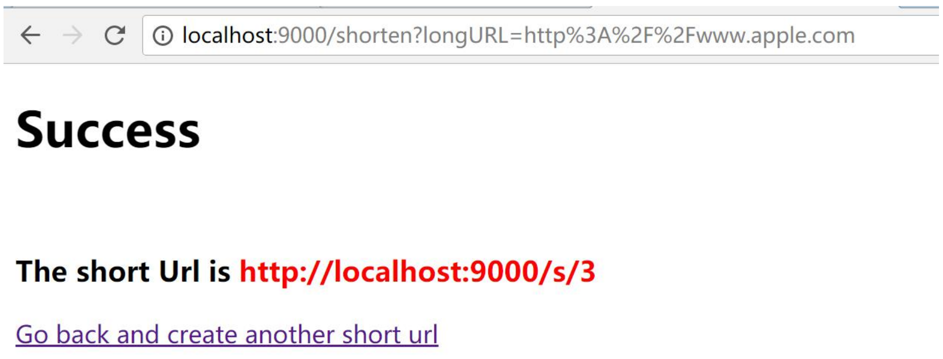
Only if input the correct url can press the create button.


Get you short URL here

please enter URL

http://www.apple.com

Second, when press the create button, server side will catch the request. And the server side give the result.

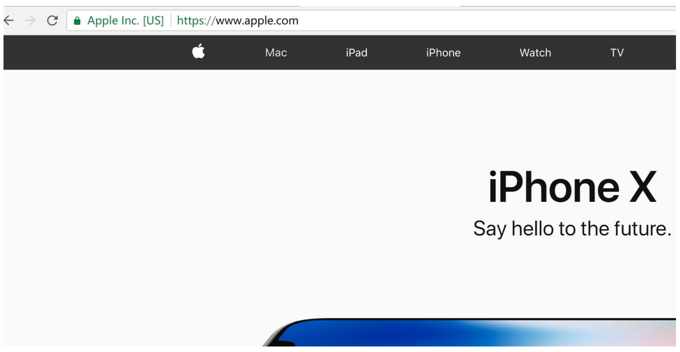

localhost:9000/shorten?longURL=http%3A%2F%2Fwww.apple.com

Success

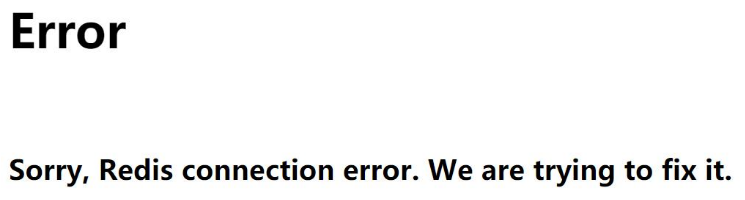
The short Url is http://localhost:9000/s/3

Go back and create another short url

Then, you can use the red short URL to access the original url (in this case is www.apple.com).



If there is something wrong in the system (Such as Redis server goes down), the error page will come out.



Error

Sorry, Redis connection error. We are trying to fix it.

2. Technique Stack and Approach Description

Web Dev: SpringMVC 4.2.4 , JSP 2.0, Servlet 2.5 , Spring 4.2.4

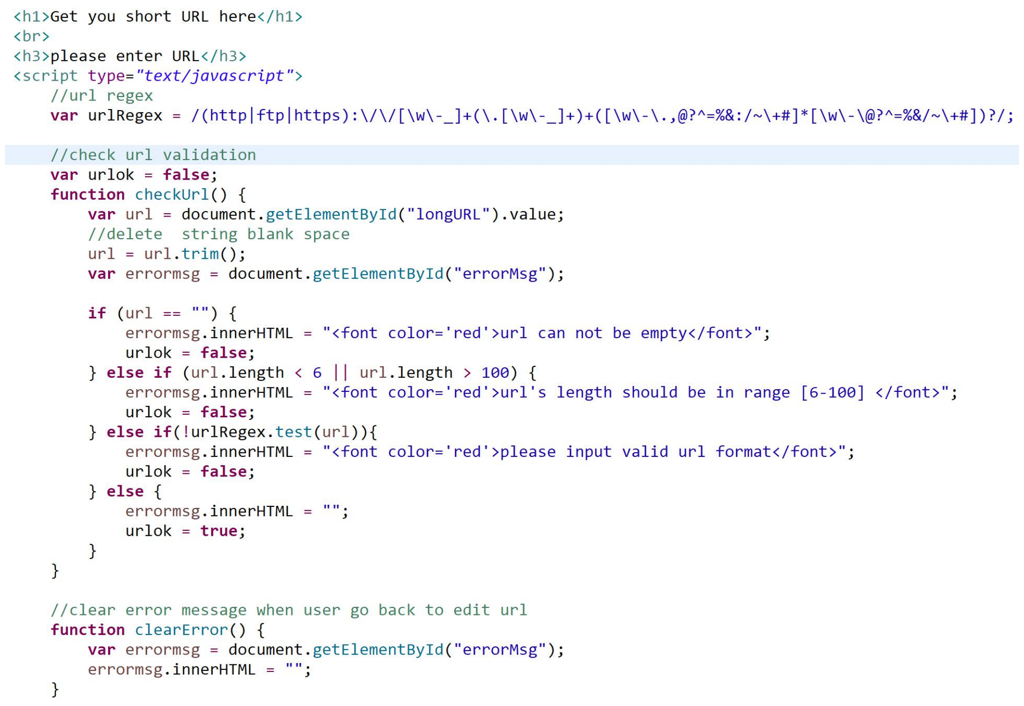
Database : Redis Jedis 2.7.2

Test: Junit 4.12

Workflow:

1. Designed several Javascript functions to provide hints on UI JSP

   until user enter the valid URL.



```
<h1>Get you short URL here</h1>
<br>
<h3>please enter URL</h3>
<script type="text/javascript">
    //url regex
    var urlRegex = /(http|ftp|https):\/\/[\w\-_]+(\.[\w\-_]+)+([\w\-\.,@?^=%&:/~\+#]*[\w\-\@?^=%&/~\+#])?/;

    //check url validation
    var urlok = false;
    function checkUrl() {
        var url = document.getElementById("longURL").value;
        //delete  string blank space
        url = url.trim();
        var errormsg = document.getElementById("errorMsg");

        if (url == "") {
            errormsg.innerHTML = "<font color='red'>url can not be empty</font>";
            urlok = false;
        } else if (url.length < 6 || url.length > 100) {
            errormsg.innerHTML = "<font color='red'>url's length should be in range [6-100] </font>";
            urlok = false;
        } else if(!urlRegex.test(url)){
            errormsg.innerHTML = "<font color='red'>please input valid url format</font>";
            urlok = false;
        } else {
            errormsg.innerHTML = "";
            urlok = true;
        }
    }

    //clear error message when user go back to edit url
    function clearError() {
        var errormsg = document.getElementById("errorMsg");
        errormsg.innerHTML = "";
    }
```

```
//clear error message when user go back to edit url
function clearError() {
    var errormsg = document.getElementById("errorMsg");
    errormsg.innerHTML = "";
}

//only if user input valid url can summit form
function checkAll() {
    return urlok;
}
```

Form:

```
<form name="longURLform" action="http://localhost:9000/shorten"
    method="get" onsubmit="return checkAll()">
    <input type="text" name="longURL" id="longURL" onblur="checkUrl();"
        onfocus="clearError();" />
    <span id="errorMsg"></span><br>
    <input type="reset" value="reset" />
    <input type="submit" value="create" />
</form>
```

2. Configure Spring container, set up Redis server and initialize Redis

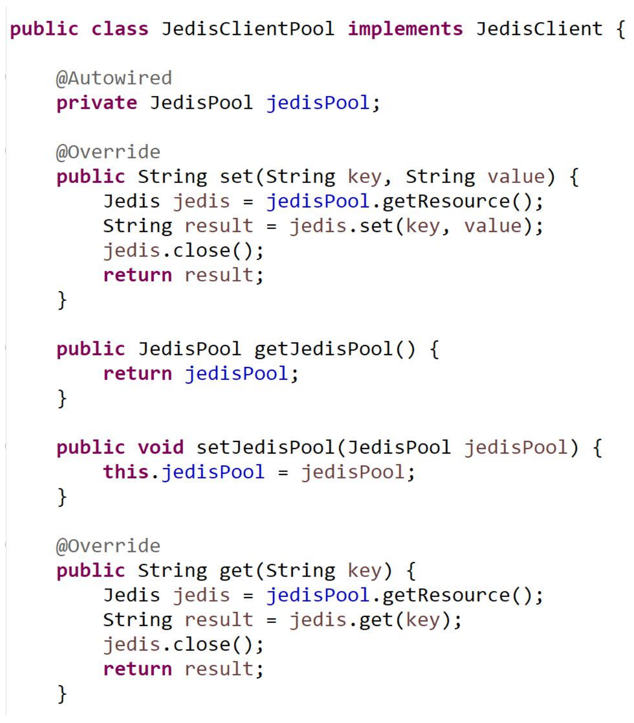
   connection pool instance in containers.

```
<!-- load configuration if need-->
<context:property-placeholder location="classpath:conf/resource.properties" />

<!-- redis single node -->
<bean id="jedisClientPool" class="com.appletest.jedis.JedisClientPool">
    <property name="jedisPool" ref="jedisPool"></property>
</bean>

<!-- Redis connection pool and server address -->
<bean id="jedisPool" class="redis.clients.jedis.JedisPool">
    <constructor-arg name="host" value="${REDIS_SERVER_IP}"/>
    <constructor-arg name="port" value="${REDIS_SERVER_PORT}" />
</bean>
```

3. Design Java Redis client interfaces and implementations.

```
public interface JedisClient {

    String set(String key, String value);
    String get(String key);
    Boolean exists(String key);
    Long expire(String key, int seconds);
    Long ttl(String key);
    Long incr(String key);
    Long hset(String key, String field, String value);
    String hget(String key, String field);
    Long hdel(String key, String field);
    Boolean hexists(String key, String field);
    List<String> hvals(String key);
    Long del(String key);
}
```

Part of implementations.

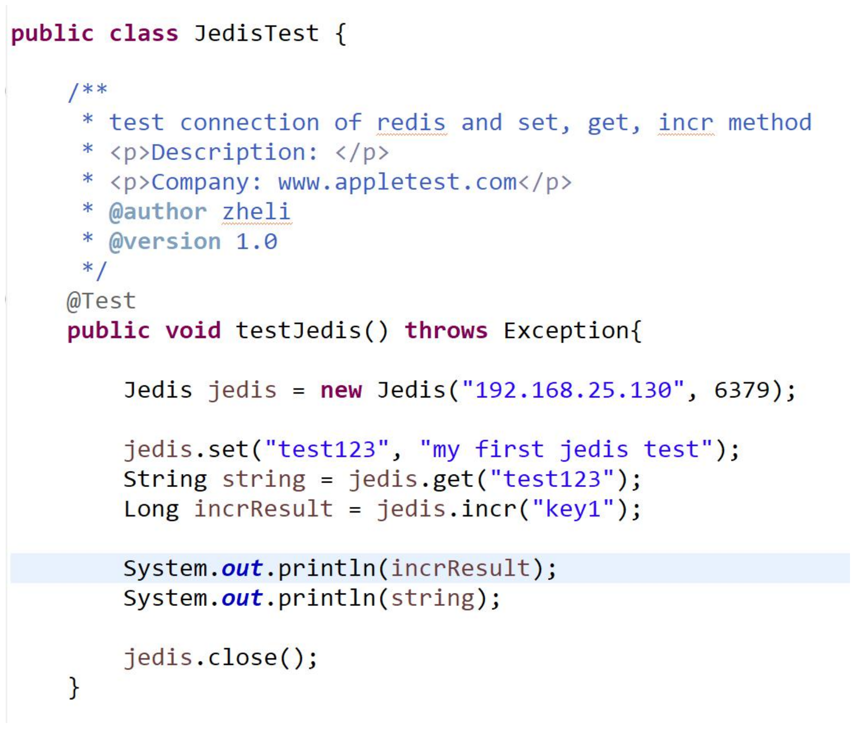


```java
public class JedisClientPool implements JedisClient {

    @Autowired
    private JedisPool jedisPool;

    @Override
    public String set(String key, String value) {
        Jedis jedis = jedisPool.getResource();
        String result = jedis.set(key, value);
        jedis.close();
        return result;
    }

    public JedisPool getJedisPool() {
        return jedisPool;
    }

    public void setJedisPool(JedisPool jedisPool) {
        this.jedisPool = jedisPool;
    }

    @Override
    public String get(String key) {
        Jedis jedis = jedisPool.getResource();
        String result = jedis.get(key);
        jedis.close();
        return result;
    }
```

4. Write several test cases to check the database connection

   methods needed in this project.

   Part of tests.



```java
public class JedisTest {

    /**
     * test connection of redis and set, get, incr method
     * <p>Description: </p>
     * <p>Company: www.appletest.com</p>
     * @author zheli
     * @version 1.0
     */
    @Test
    public void testJedis() throws Exception{

        Jedis jedis = new Jedis("192.168.25.130", 6379);

        jedis.set("test123", "my first jedis test");
        String string = jedis.get("test123");
        Long incrResult = jedis.incr("key1");

        System.out.println(incrResult);
        System.out.println(string);

        jedis.close();
    }
```

5. Design web controller to handle request with longURL parameter

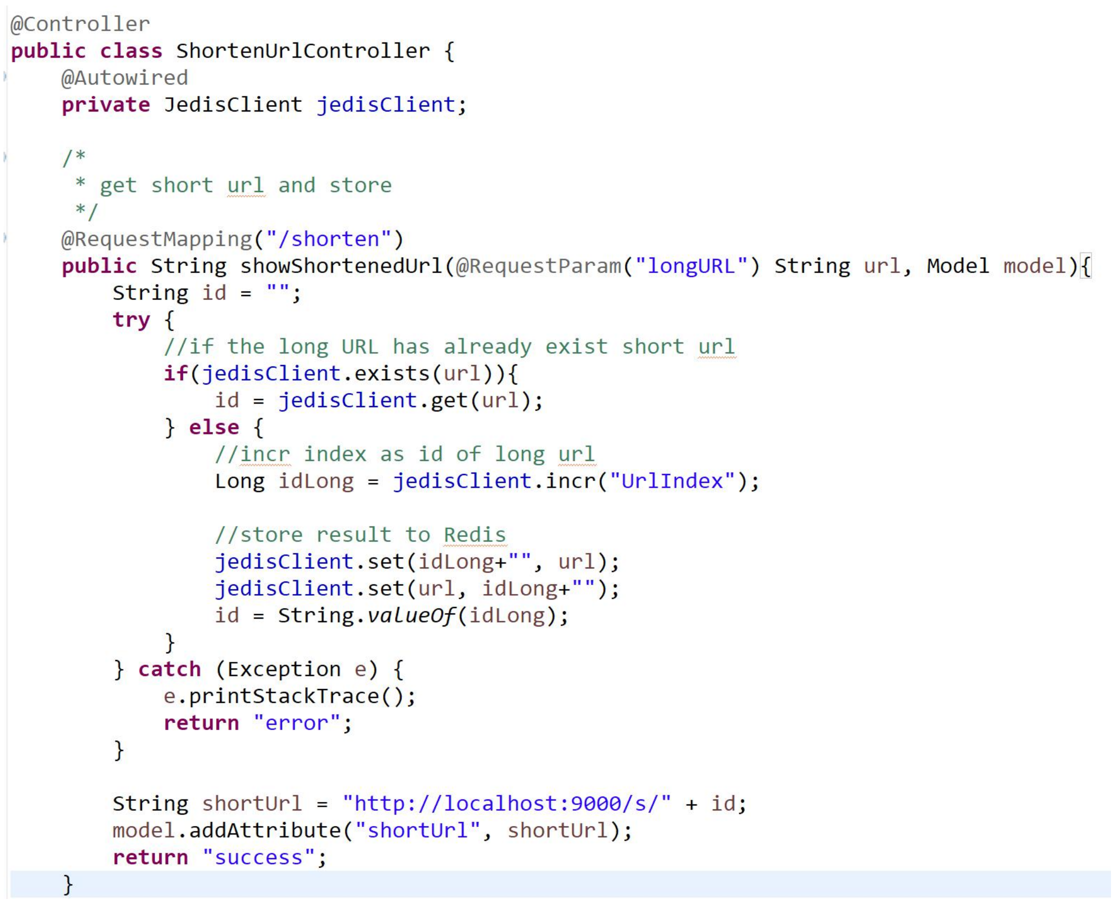
   using SpringMVC. Store the mapping relationship in Redis.

   Using the Redis incr function to generate shortened ID.

Configuration:

```xml
<!-- load configuration if need-->
<context:property-placeholder location="classpath:conf/resource.properties" />

<!-- scan controller package -->
<context:component-scan base-package="com.appletest.controller"/>
<mvc:annotation-driven />

<!-- configure view resolver -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>
```

Get short url and store mapping relationship handler.

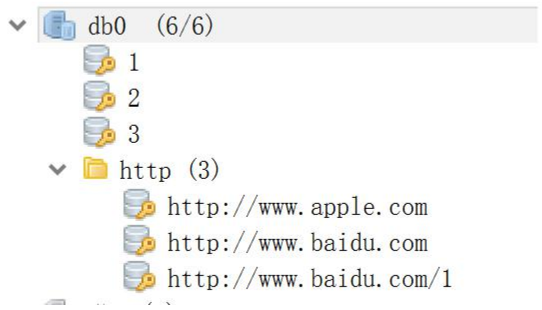


```java
@Controller
public class ShortenUrlController {
    @Autowired
    private JedisClient jedisClient;

    /*
     * get short url and store
     */
    @RequestMapping("/shorten")
    public String showShortenedUrl(@RequestParam("longURL") String url, Model model){
        String id = "";
        try {
            //if the long URL has already exist short url
            if(jedisClient.exists(url)){
                id = jedisClient.get(url);
            } else {
                //incr index as id of long url
                Long idLong = jedisClient.incr("UrlIndex");

                //store result to Redis
                jedisClient.set(idLong+"", url);
                jedisClient.set(url, idLong+"");
                id = String.valueOf(idLong);
            }
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }

        String shortUrl = "http://localhost:9000/s/" + id;
        model.addAttribute("shortUrl", shortUrl);
        return "success";
    }
```

Redirected handler. Get the short URL and query database to get and redirect request to original URL.

```java
/*
 * redirect
 */
@RequestMapping("/s/{id}")
public void shortUrlRedirect(@PathVariable String id, HttpServletResponse response){
    try {
        String url = jedisClient.get(id);
        //redirect
        response.sendRedirect(url);

    } catch (Exception e) {
        e.printStackTrace();
        try {
            response.sendRedirect("http://localhost:9000/error.jsp");
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
}
```

Redis key details. Each incr number correspond one URL.



```
db0  (6/6)
    1
    2
    3
  http (3)
      http://www.apple.com
      http://www.baidu.com
      http://www.baidu.com/1
```

(I do not set up the key expire time. I assume short url does not expire.)

3. How to Launch Apps

Method 1:

Environment Requirement:

1. Redis installed and service on port 6379.

   (You Redis server IP and port can be changed on the path
/shortenurl-web/src/main/resources/conf/resource.properties)

2. Java 8 and Maven installed
3. Tomcat server installed

Process:

1. Start Redis service. (My default is 192.168.25.130:6379).
2. Deploy the war package on tomcat and set port 9000.
3. Access the index page (localhost:9000) on browser.
4. Enter the long URL that you want to be shortened.
5. Copy the shorten url to the browser and you can be redirected to the original url.

Method 1:

Environment Requirement:

1. Redis installed and service on port 6379.
2. Java 8 and Maven installed

Process:

1. Start Redis service. (My default is 192.168.25.130:6379).
2. Open this project on your own IDE. Run maven build with "clean tomcat7:run" (I installed a tomcat plug-in and assign on port 9000. So you don't have to have a tomcat server).
3. Access the index page (localhost:9000) on browser.
4. Enter the long URL that you want to be shortened.
5. Copy the shorten url to the browser and you can be redirected to the original url.