**Redis Cache**

Redis version used Redis-x64-3.2.100 .software avail in D:\software

Used in Spring Boot

# Redis Config

spring.cache.type=redis

spring.redis.host=localhost

spring.redis.port=6379

Dependency

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

Used in RoleServiceImpl findAll roles

@Cacheable(value = "roleList")

In create roll method clearing cache

@CacheEvict(value=" roleList ",allEntries=true)

Refer Download

Resource resource = fileService.getFile(id);

String contentType = null;

try {

contentType = request.getServletContext().getMimeType(resource.getFile().getAbsolutePath());

} catch (IOException ex) {

logger.info("Could not determine file type.");

}

// Fallback to the default content type if type could not be determined

if(contentType == null) {

contentType = "application/octet-stream";

}

JSONResponse jsonResponse = new JSONResponse();

jsonResponse.setStatus(HttpStatus.OK.value());

jsonResponse.setResult(resource);

//return new ResponseEntity<JSONResponse>(jsonResponse, HttpStatus.OK).ok().contentType(MediaType.parseMediaType(contentType)).header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=\"" + resource.getFilename() + "\""));

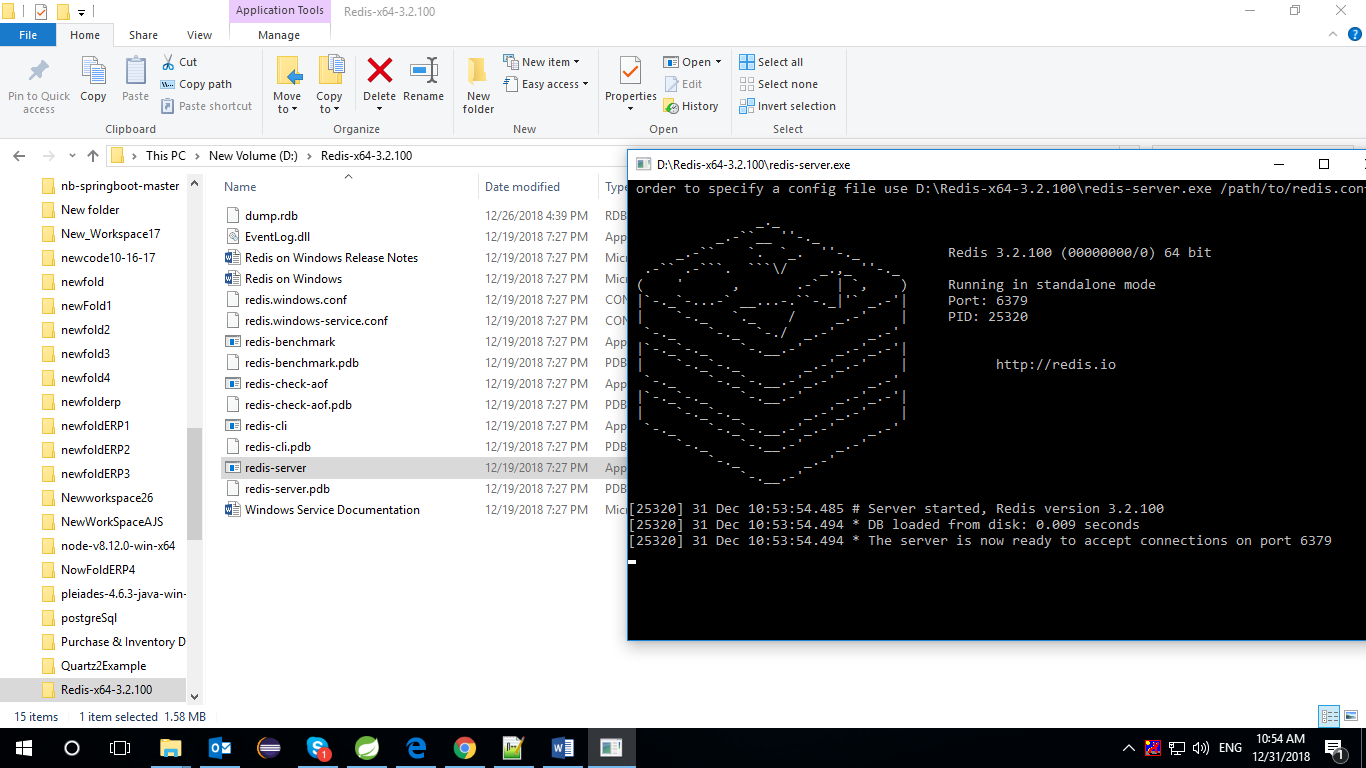
return ResponseEntity.ok()

.contentType(MediaType.parseMediaType(contentType))

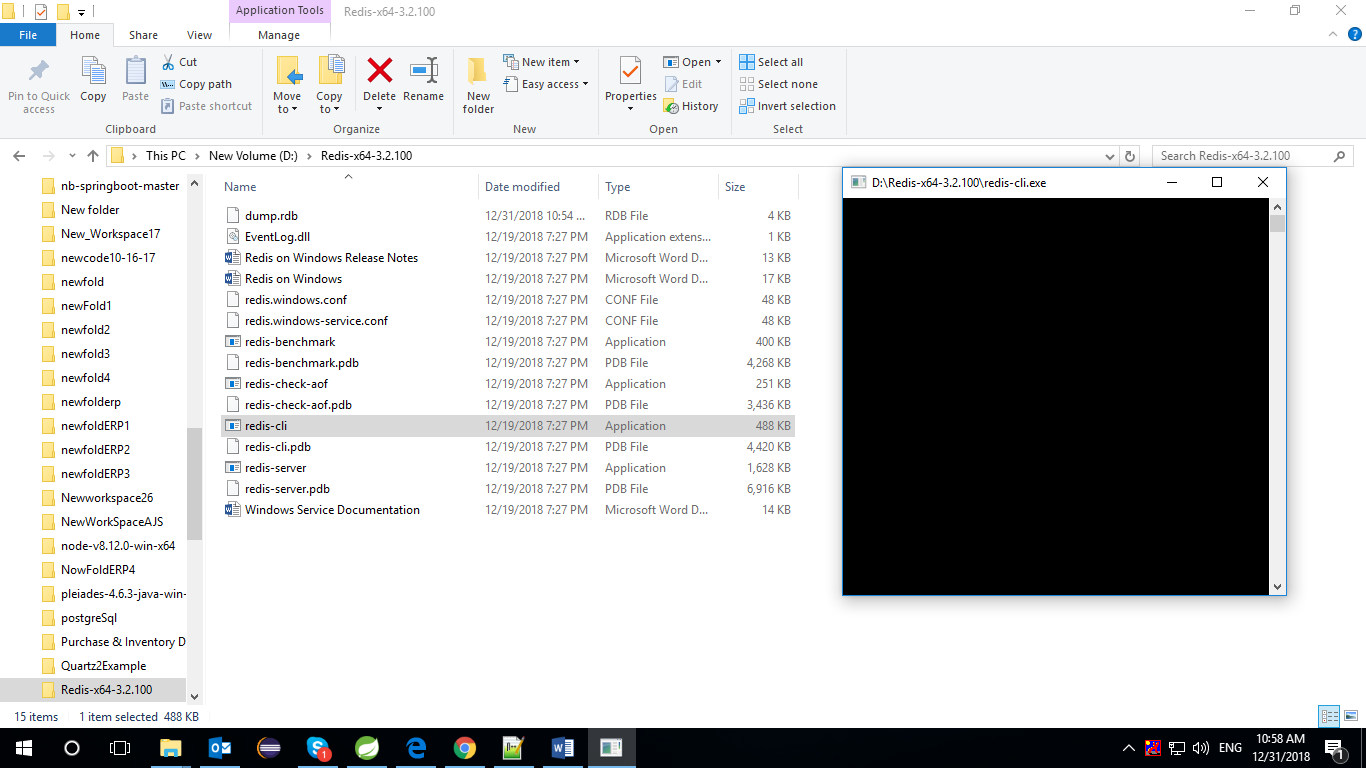
.header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=\"" + resource.getFilename() + "\"")

.body(resource);

To start redis server



To check cache data



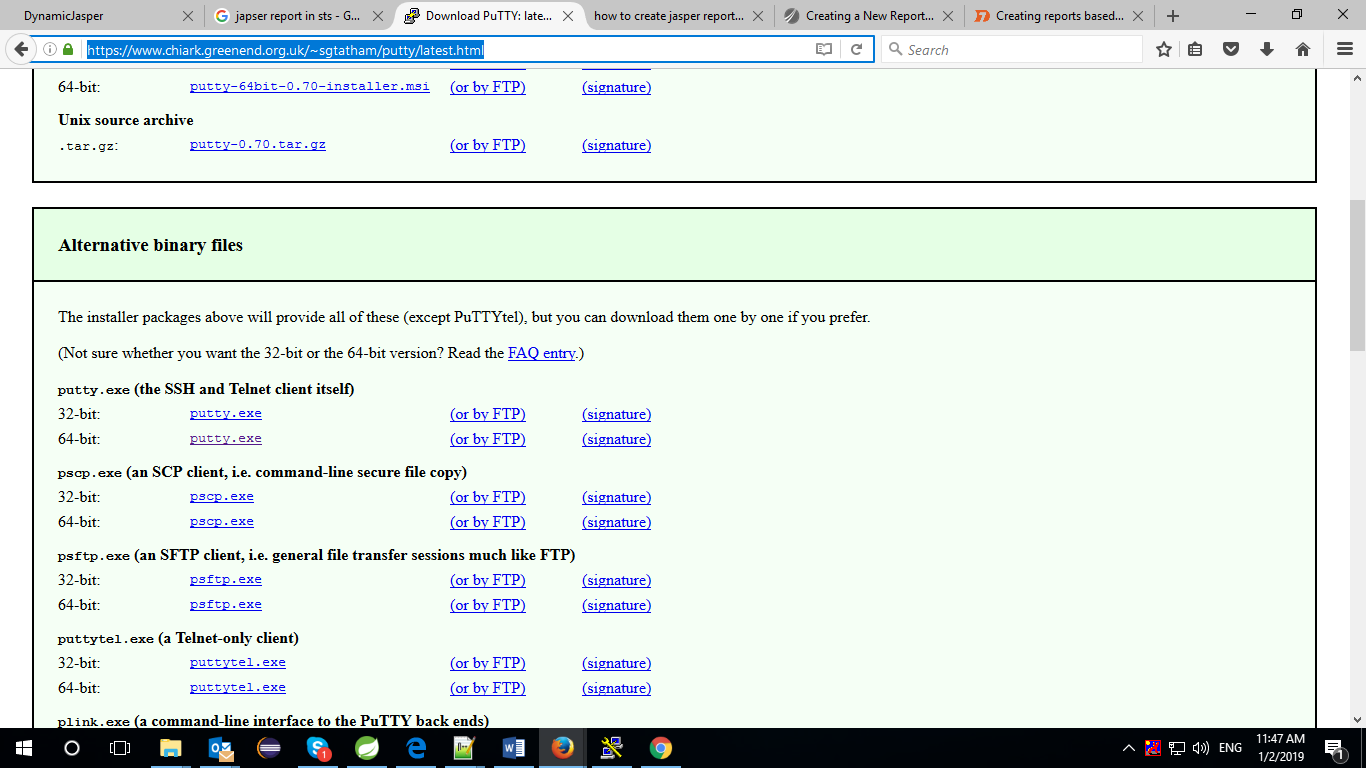
Keys \*

flushAll

putty

======

Installation <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>



**Jasper Report**

================

Interceptor

package com.srmuniv..security.client.config;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Map;

import java.util.Set;

import java.util.StringTokenizer;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.security.authentication.AnonymousAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Component;

import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

@Component

public class UserServiceCheckInterceptor extends HandlerInterceptorAdapter {

private static final Logger LOG = LoggerFactory.getLogger(UserServiceCheckInterceptor.class);

@Autowired

private MessageSource MessageSource;

@Autowired

private RoleService roleService;

private List<Role> roleList = null;

private Map<Long, Role> roleMap = null;

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

throws Exception {

CustomUserDetails customUserDetails = null;

boolean access = true;

List<String> authorityList = null;

String roleAccessControlSet = null;

Set<String> roleAccessControlList = new HashSet<>();

String url = request.getRequestURI();

try {

if ((!(SecurityContextHolder.getContext().getAuthentication() instanceof AnonymousAuthenticationToken))) {

OAuth2AuthenticationUser authentication = (OAuth2AuthenticationUser) SecurityContextHolder.getContext()

.getAuthentication();

if (authentication != null) {

customUserDetails = authentication.getCustomUserDetails();

authorityList = new ArrayList<String>();

StringTokenizer st = new StringTokenizer(customUserDetails.getAuthorities().toString(), ",=[]{}");

while (st != null && st.hasMoreTokens()) {

String s2 = st.nextToken();

if (s2 != null && !CommonConstant.AUTHORITY.equalsIgnoreCase(s2)

&& !" ".equalsIgnoreCase(s2)) {

authorityList.add(s2.trim());

}

}

String screenIdStr = request.getHeader(CommonConstant.SCREEN\_ID);

// LookUp service Start

if (url != null && url.startsWith(CommonConstant.LOOK\_UP\_END\_POINT)) {// Only for lookup

access = true;

screenIdStr = CommonConstant.DUMMY;

roleAccessControlList.add(CommonConstant.DUMMY);

}

// LookUp service End

if (screenIdStr == null || "".equalsIgnoreCase(screenIdStr)) {

access = false;

response.sendError(HttpStatus.BAD\_REQUEST.value(),

MessageSource.getMessage(ErrorMessageCode.INVALID\_REQUEST));

}

roleMap = new HashMap<Long, Role>();

roleList = roleService.findAllRoles();

for (Role role1 : roleList) {

roleMap.put(role1.getRoleId(), role1);

}

roleAccessControlSet = validateAuthority(authorityList, roleAccessControlSet, roleAccessControlList,

screenIdStr);

if (roleAccessControlList != null && roleAccessControlList.isEmpty()) {

access = false;

response.sendError(HttpStatus.BAD\_REQUEST.value(),

MessageSource.getMessage("framework.error-message.invalid-403-exception"));

}

LOG.info(" Screen Access List " + roleAccessControlSet);

}

} else {

access = true;

}

} catch (Exception exception) {

LOG.debug(exception.getMessage());

}

return access;

}

/\*\*

\*

\* @param authorityList

\* @param roleAccessControlSet

\* @param roleAccessControlList

\* @param screenIdStr

\* @return

\*/

private String validateAuthority(List<String> authorityList, String roleAccessControlSet,

Set<String> roleAccessControlList, String screenIdStr) {

for (String roleId : authorityList) {

Role role = roleMap.get(Long.valueOf(roleId));

for (ModuleRoleAccess mra : role.getModuleRoleAccesses()) {

List<ScreenAccess> screenList = mra.getScreenAccesses();

if (screenList != null) {

for (ScreenAccess ScreenAccess : screenList) {

if (ScreenAccess.getScreen() != null) {

Long dbScreenId = ScreenAccess.getScreen().getScreenId();

if (dbScreenId != null

&& screenIdStr.trim().equalsIgnoreCase(String.valueOf(dbScreenId).trim())) {

LOG.info(" Screen ID matched=========>");

roleAccessControlSet = ScreenAccess.getAccessControlSet();

roleAccessControlList.add(roleAccessControlSet);

}

}

}

}

}

}

return roleAccessControlSet;

}

}

Interceptor Impl

package com.srmuniv..security;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

import com.security.client.config.UserServiceCheckInterceptor;

@SpringBootApplication

public class SecurityBootApplication extends WebMvcConfigurerAdapter {

@Autowired

private UserServiceCheckInterceptor userServiceIn;

public static void main(String[] args) {

SpringApplication.run(SecurityBootApplication.class, args);

}

/\*

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(userServiceIn);

}

\*/

}