Grails开发之（Rest教程）

开发环境：

| Grails Version: 3.1.5

| Groovy Version: 2.4.6

| JVM Version: 1.8.0\_144

创建项目（略）

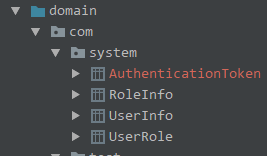
1. 配置build.gradle配置：

buildscript {  
 ext {  
 grailsVersion = project.grailsVersion  
 }  
 repositories {  
 mavenLocal()  
 maven { url "https://repo.grails.org/grails/core" }  
 }  
 dependencies {  
 classpath "org.grails:grails-gradle-plugin:$grailsVersion"  
 classpath "com.bertramlabs.plugins:asset-pipeline-gradle:2.8.2"  
 classpath "org.grails.plugins:hibernate4:5.0.5"  
 }  
}  
  
version "0.1"  
group "demo"  
  
apply plugin:"eclipse"  
apply plugin:"idea"  
apply plugin:"war"  
apply plugin:"org.grails.grails-web"  
apply plugin:"org.grails.grails-gsp"  
apply plugin:"asset-pipeline"  
  
ext {  
 grailsVersion = project.grailsVersion  
 gradleWrapperVersion = project.gradleWrapperVersion  
 springSecurityRestVersion = '2.0.0.M2'  
}  
  
repositories {  
 mavenLocal()  
 maven { url "https://repo.grails.org/grails/core" }  
}  
  
dependencyManagement {  
 imports {  
 mavenBom "org.grails:grails-bom:$grailsVersion"  
 }  
 applyMavenExclusions false  
}  
  
dependencies {  
 compile "org.springframework.boot:spring-boot-starter-logging"  
 compile "org.springframework.boot:spring-boot-autoconfigure"  
 compile "org.grails:grails-core"  
 compile "org.springframework.boot:spring-boot-starter-actuator"  
 compile "org.springframework.boot:spring-boot-starter-tomcat"  
 compile "org.grails:grails-dependencies"  
 compile "org.grails:grails-web-boot"  
 compile "org.grails.plugins:cache"  
 compile "org.grails.plugins:scaffolding"  
 compile "org.grails.plugins:hibernate4"  
 compile "org.hibernate:hibernate-ehcache"  
 console "org.grails:grails-console"  
 profile "org.grails.profiles:web:3.1.5"  
 runtime "com.bertramlabs.plugins:asset-pipeline-grails:2.8.2"  
 runtime "com.h2database:h2"  
 testCompile "org.grails:grails-plugin-testing"  
 testCompile "org.grails.plugins:geb"  
 testRuntime "org.seleniumhq.selenium:selenium-htmlunit-driver:2.47.1"  
 testRuntime "net.sourceforge.htmlunit:htmlunit:2.18"  
  
 //spring-security依赖  
 compile "org.grails.plugins:spring-security-core:3.0.0"  
 //rest api依赖  
 compile "org.grails.plugins:spring-security-rest:${springSecurityRestVersion}"  
 //rest api存储token的依赖  
 compile "org.grails.plugins:spring-security-rest-gorm:${springSecurityRestVersion}"  
  
 //mysql依赖  
 compile "mysql:mysql-connector-java:5.1.38"  
}  
  
task wrapper(type: Wrapper) {  
 gradleVersion = gradleWrapperVersion  
}  
  
assets {  
 minifyJs = true  
 minifyCss = true  
}

1. 在项目目录grails-app/conf下面创建脚本application.groovy，配置spring secruity和rest相关配置

// 添加spring-security插件核心配置  
  
grails.plugin.springsecurity.auth.loginFormUrl = '/'  
grails.plugin.springsecurity.auth.ajaxLoginFormUrl = '/'  
grails.plugin.springsecurity.failureHandler.defaultFailureUrl = '/'  
grails.plugin.springsecurity.failureHandler.ajaxAuthFailUrl = '/'  
//加密方式  
grails.plugin.springsecurity.password.algoritham = 'SHA-256'  
  
//用户类  
grails.plugin.springsecurity.userLookup.userDomainClassName = 'com.system.UserInfo'  
  
//用户角色类  
grails.plugin.springsecurity.userLookup.authorityJoinClassName = 'com.system.UserRole'  
  
//角色类  
grails.plugin.springsecurity.authority.className = 'com.system.RoleInfo'  
  
//角色集合字段，是UserInfo的getAuthorities()  
grails.plugin.springsecurity.authority.groupAuthorityNameField = 'authorities'  
  
//是否使用组类进行管理，如果使用了true，则必须配置组的相关东西  
grails.plugin.springsecurity.useRoleGroups = false  
  
//请求的url属性字段  
grails.plugin.springsecurity.requestMap.urlField = 'url'  
  
grails.plugin.springsecurity.relationalAuthorities='allRoles'  
  
  
grails.plugin.springsecurity.filterChain.chainMap = [  
 [pattern: '/api/\*\*', filters:'JOINED\_FILTERS,-anonymousAuthenticationFilter,-exceptionTranslationFilter,-authenticationProcessingFilter,-securityContextPersistenceFilter'],  
 [pattern: '/\*\*', filters:'JOINED\_FILTERS,-restTokenValidationFilter,-restExceptionTranslationFilter']  
]  
  
grails.plugin.springsecurity.controllerAnnotations.staticRules = [  
 [pattern: '/', access: ['permitAll']],  
 [pattern: '/500', access: ['permitAll']],  
 [pattern: '/404', access: ['permitAll']],  
 [pattern: '/error', access: ['permitAll']],  
 [pattern: '/index', access: ['permitAll']],  
 [pattern: '/index.gsp', access: ['permitAll']],  
 [pattern: '/shutdown', access: ['permitAll']],  
 [pattern: '/assets/\*\*', access: ['permitAll']],  
 [pattern: '/\*\*/js/\*\*', access: ['permitAll']],  
 [pattern: '/\*\*/css/\*\*', access: ['permitAll']],  
 [pattern: '/\*\*/images/\*\*', access: ['permitAll']],  
 [pattern: '/\*\*/favicon.ico', access: ['permitAll']],  
// [pattern: '/j\_spring\_security\_check', access: ['permitAll']],  
 // block all other URL access  
 [pattern: '/\*\*', access: ['denyAll'], httpMethod: 'GET'],  
 [pattern: '/\*\*', access: ['denyAll'], httpMethod: 'POST'],  
 [pattern: '/\*\*', access: ['denyAll'], httpMethod: 'PUT'],  
 [pattern: '/\*\*', access: ['denyAll'], httpMethod: 'DELETE']  
]  
  
grails.plugin.springsecurity.filterChain.chainMap = [  
 [pattern: '/assets/\*\*', filters: 'none'],  
 [pattern: '/\*\*/js/\*\*', filters: 'none'],  
 [pattern: '/\*\*/css/\*\*', filters: 'none'],  
 [pattern: '/\*\*/images/\*\*', filters: 'none'],  
 [pattern: '/\*\*/favicon.ico', filters: 'none'],  
// [pattern: '/api/login', filters: 'securityCorsFilter,restAuthenticationFilter'],  
 //Stateless chain  
 [  
 pattern: '/api/\*\*',  
 filters: 'JOINED\_FILTERS,-securityCorsFilter,-anonymousAuthenticationFilter,-exceptionTranslationFilter,-authenticationProcessingFilter,-securityContextPersistenceFilter,-rememberMeAuthenticationFilter'  
 ],  
  
 //Traditional chain  
 [  
 pattern: '/\*\*',  
 filters: 'JOINED\_FILTERS,-securityCorsFilter,-restTokenValidationFilter,-restExceptionTranslationFilter'  
 ]  
]  
  
  
//rest configuration  
grails.plugin.springsecurity.rest.token.storage.useGorm = true // since using gorm for token storage  
grails.plugin.springsecurity.rest.token.generation.useSecureRandom = true  
grails.plugin.springsecurity.rest.login.active =true  
grails.plugin.springsecurity.rest.login.useJsonCredentials = true // can use json a request parameter  
grails.plugin.springsecurity.rest.login.usernamePropertyName = 'username' // field of username parameter  
grails.plugin.springsecurity.rest.login.passwordPropertyName = 'password' // field of pasword parameter  
grails.plugin.springsecurity.rest.login.useRequestParamsCredential = true  
  
grails.plugin.springsecurity.rest.token.storage.gorm.tokenDomainClassName ='com.system.AuthenticationToken' // token class name with package  
grails.plugin.springsecurity.rest.token.storage.gorm.tokenValuePropertyName = 'secretToken' // field name for token storage  
grails.plugin.springsecurity.rest.token.storage.gorm.usernamePropertyName = 'loginName'  
  
grails.plugin.springsecurity.rest.logout.endpointUrl = '/api/logout'  
grails.plugin.springsecurity.rest.login.endpointUrl = '/api/login'  
grails.plugin.springsecurity.rest.login.failureStatusCode = 401  
//token validate  
grails.plugin.springsecurity.rest.token.validation.active=true  
grails.plugin.springsecurity.rest.token.generation.useUUID=false  
grails.plugin.springsecurity.rest.token.validation.useBearerToken = true  
grails.plugin.springsecurity.rest.token.validation.headerName = 'X-Auth-Token'  
grails.plugin.springsecurity.rest.token.validation.endpointUrl='/api/validate'  
//end of rest configuration  
grails.plugin.springsecurity.logout.postOnly = false

1. 创建domain



3.1 AuthenticationToken

package com.system  
  
class AuthenticationToken {  
  
 String secretToken // field to store the token used for accessing api end point  
 String loginName // login name of the user  
  
 static *mapping* = {  
 version false  
 }  
}

3.2 UserInfo

package com.system  
  
import groovy.transform.EqualsAndHashCode  
import groovy.transform.ToString  
  
*/\*\*  
 \* 用户  
 \*/*@EqualsAndHashCode(includes='username')  
@ToString(includes='username', includeNames=true, includePackage=false)  
class UserInfo implements Serializable {  
  
 private static final long *serialVersionUID* = 1  
  
 transient springSecurityService  
  
 String username  
 String password  
 boolean enabled = true  
 boolean accountExpired  
 boolean accountLocked  
 boolean passwordExpired  
  
 UserInfo(String username, String password) {  
 this()  
 this.username = username  
 this.password = password  
 }  
  
 Set<RoleInfo> getAuthorities() {  
 UserRole.*findAllByUser*(this)\*.role as Set<RoleInfo>  
 }  
  
// Set<RoleGroup> getAuthorities() {  
// def authorities = (UserRoleGroup.findAllByUserInfo(this) as List<UserRoleGroup>)\*.roleGroup as Set<RoleGroup>  
// authorities  
// }  
  
 def beforeInsert() {  
 encodePassword()  
 }  
  
 def beforeUpdate() {  
 if (isDirty('password')) {  
 encodePassword()  
 }  
 }  
  
 protected void encodePassword() {  
 password = springSecurityService?.passwordEncoder ? springSecurityService.encodePassword(password) : password  
 }  
  
 static *transients* = ['springSecurityService']  
  
 static *constraints* = {  
 password blank: false, password: true  
 username blank: false, unique: true  
 }  
  
 static *mapping* = {  
 password column: '`password`'  
 }  
}

3.3 RoleInfo

package com.system  
  
import groovy.transform.EqualsAndHashCode  
import groovy.transform.ToString  
  
@EqualsAndHashCode(includes='authority')  
@ToString(includes='authority', includeNames=true, includePackage=false)  
class RoleInfo implements Serializable {  
  
 private static final long *serialVersionUID* = 1  
  
 RoleInfo(String authority, String authorityName) {  
 this.authority = authority  
 this.authorityName = authorityName  
 }  
  
 String authority //权限标识  
 String authorityName //权限名称  
  
 static *constraints* = {  
 authority blank: false, unique: true  
 }  
  
 static *mapping* = {  
 cache true  
 }  
  
  
 @Override  
 public String toString() {  
 "${authorityName}(${authority})"  
 }  
}

3.4 UserRole

*/\*\*  
 \* 用户-角色关联表  
 \* \*/*package com.system  
  
import grails.gorm.DetachedCriteria  
import groovy.transform.ToString  
import org.apache.commons.lang.builder.HashCodeBuilder  
  
@ToString(cache=true, includeNames=true, includePackage=false)  
class UserRole implements Serializable {  
  
 private static final long *serialVersionUID* = 1  
  
 UserInfo user  
 RoleInfo role  
  
 UserRole(UserInfo u, RoleInfo r) {  
 this()  
 user = u  
 role = r  
 }  
  
 @Override  
 boolean equals(other) {  
 if (!(other instanceof UserRole)) {  
 return false  
 }  
  
 other.user?.id == user?.id && other.role?.id == role?.id  
 }  
  
 @Override  
 int hashCode() {  
 def builder = new HashCodeBuilder()  
 if (user) builder.append(user.id)  
 if (role) builder.append(role.id)  
 builder.toHashCode()  
 }  
  
 static UserRole get(long userId, long roleId) {  
 *criteriaFor*(userId, roleId).get()  
 }  
  
 static boolean exists(long userId, long roleId) {  
 *criteriaFor*(userId, roleId).count()  
 }  
  
 private static DetachedCriteria criteriaFor(long userId, long roleId) {  
 UserRole.*where* {  
 user == UserInfo.*load*(userId) &&  
 role == RoleInfo.*load*(roleId)  
 }  
 }  
  
 static UserRole create(UserInfo user, RoleInfo role, boolean flush = false) {  
 def instance = new UserRole(user: user, role: role)  
 instance.save(flush: flush, insert: true)  
 instance  
 }  
  
 static boolean remove(UserInfo u, RoleInfo r, boolean flush = false) {  
 if (u == null || r == null) return false  
  
 int rowCount = UserRole.*where* { user == u && role == r }.deleteAll()  
  
 if (flush) { UserRole.*withSession* { it.flush() } }  
  
 rowCount  
 }  
  
 static boolean removeAll(UserInfo u, boolean flush = false) {  
 if (u == null) return false  
  
 UserRole.*where* { user == u }.deleteAll()  
  
 if (flush) { UserRole.*withSession* { it.flush() } }  
 return true  
 }  
  
 static void removeAll(RoleInfo r, boolean flush = false) {  
 if (r == null) return  
  
 UserRole.*where* { role == r }.deleteAll()  
  
 if (flush) { UserRole.*withSession* { it.flush() } }  
 }  
  
 static *constraints* = {  
 role validator: { RoleInfo r, UserRole ur ->  
 if (ur.user == null || ur.user.id == null) return  
 boolean existing = false  
 UserRole.*withNewSession* {  
 existing = UserRole.*exists*(ur.user.id, r.id)  
 }  
 if (existing) {  
 return 'userRole.exists'  
 }  
 }  
 }  
  
 static *mapping* = {  
 id composite: ['user', 'role']  
 version false  
 }  
}

1. 添加一个测试domain类Product

package com.test  
  
import com.system.RoleInfo  
import com.system.UserInfo  
import grails.rest.Resource  
  
@Resource(readOnly = false, formats = ['json', 'xml'])  
class Product {  
  
 String prodName  
 String prodDesc  
 Double prodPrice  
 Date createDate = new Date()  
  
 static *belongsTo* = [user:UserInfo,role:RoleInfo]  
  
 static *constraints* = {  
 user nullabled:true  
 role nullabled:true  
 }  
}

1. 创建controller，用于测试

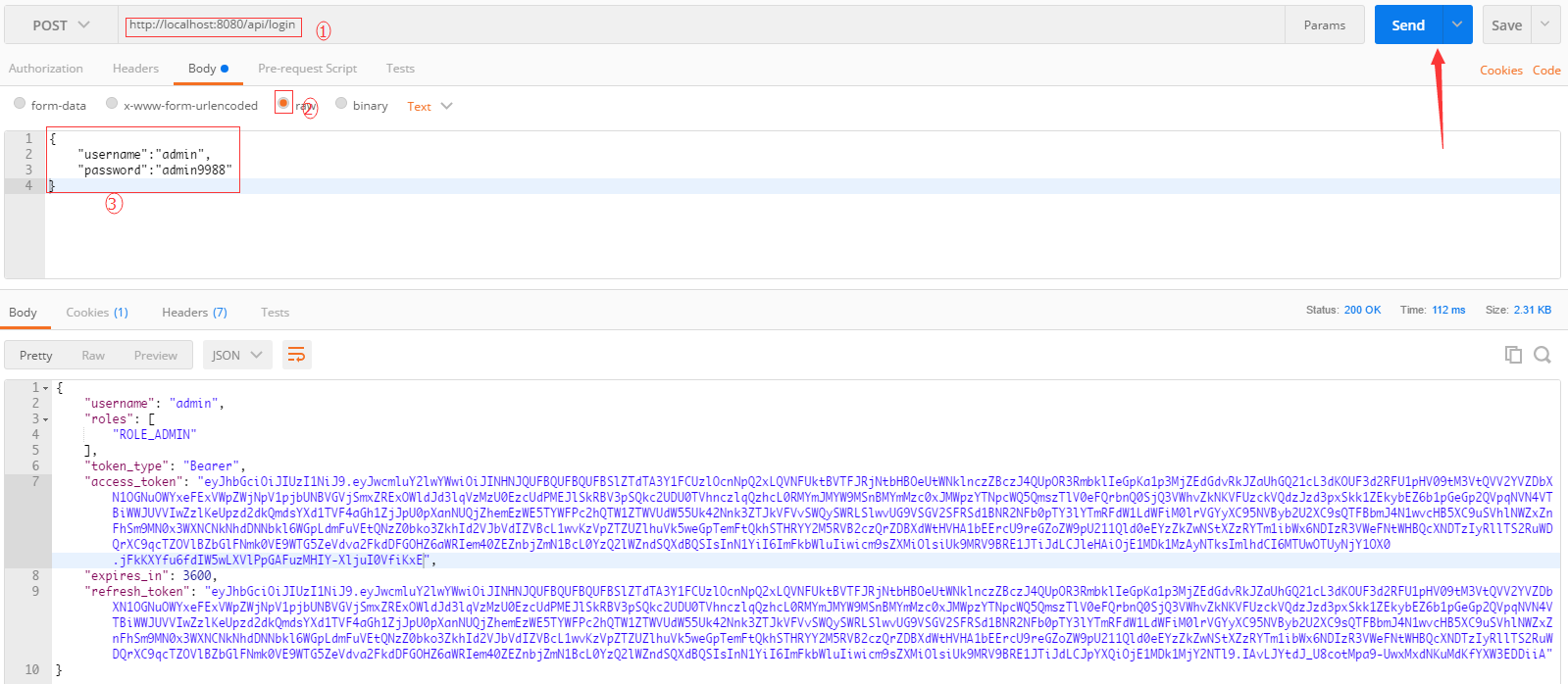
package com.test  
  
import grails.converters.JSON  
import grails.rest.RestfulController  
import org.springframework.security.access.annotation.Secured  
  
@Secured("ROLE\_ADMIN")  
class ProductController extends RestfulController<Product> {  
  
 def tokenStorageService  
 static *responseFormats* = ['json', 'xml']  
  
 ProductController() {  
 super(Product)  
 }  
  
 def list() {  
 render ([result:true, message: "操作成功"]) as JSON  
 }  
}

1. 在grails-app/init/bootstrap.groovy中添加启动任务

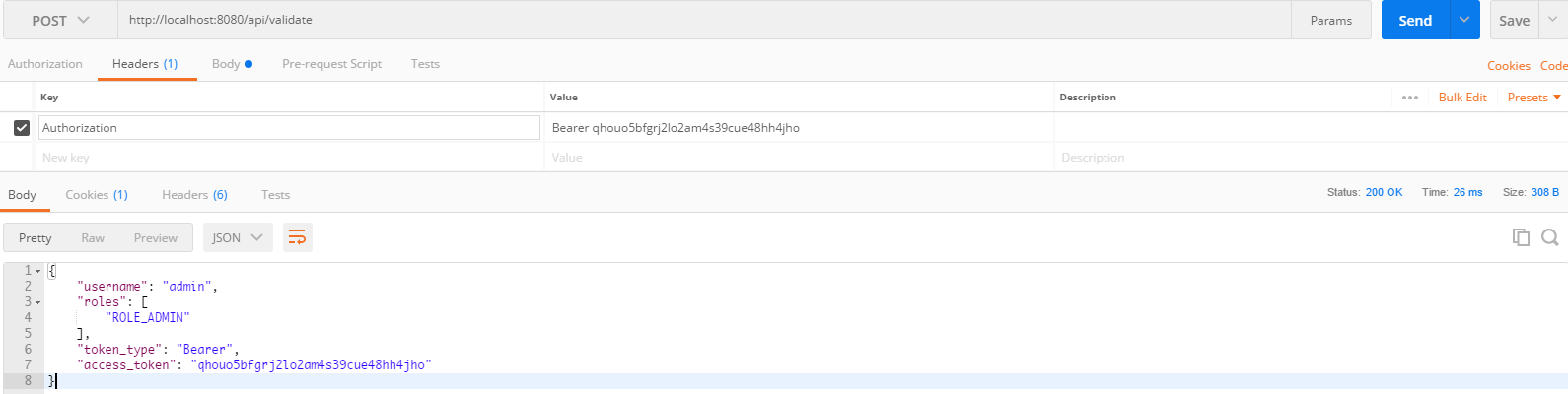
import com.system.RoleInfo  
import com.system.UserInfo  
import com.system.UserRole  
import com.test.Product  
import grails.plugin.springsecurity.SecurityFilterPosition  
import grails.plugin.springsecurity.SpringSecurityUtils  
  
class BootStrap {  
  
 def init = { servletContext ->  
  
 //注册注销过滤器，如果不添加则注销不起作用  
 SpringSecurityUtils.*clientRegisterFilter*('restLogoutFilter', SecurityFilterPosition.*LOGOUT\_FILTER*.order - 1)  
  
 //建立默认用户权限  
 def superadminRoleInfo = new RoleInfo('ROLE\_SUPERADMIN', '超级管理员').save()  
 def adminRoleInfo = new RoleInfo('ROLE\_ADMIN', '管理员').save()  
 def userRoleInfo = new RoleInfo('ROLE\_USER', '用户').save()  
  
  
 def superUser = new UserInfo('sys', 'sys9988').save()  
 UserRole.*create* superUser, superadminRoleInfo, true  
  
 def user = new UserInfo('admin', 'admin9988').save()  
 UserRole.*create* user, adminRoleInfo, true  
  
 def testUser = new UserInfo('tom', 'tom9988').save()  
 UserRole.*create* testUser, userRoleInfo, true  
  
  
 def prod1 = new Product(prodName:"iPhone 7",prodDesc:"New iPhone 7 32GB",prodPrice:780).save flush:true  
 def prod2 = new Product(prodName:"iPhone 7 Plus",prodDesc:"New iPhone 7 Plus 128GB",prodPrice:990).save flush:true  
 def prod3 = new Product(prodName:"iPhone 7 SE",prodDesc:"New iPhone 7 SE 64GB",prodPrice:520).save flush:true  
 }  
  
 def destroy = {  
  
 }  
}

1. 通过PostMan接口测试工具进行测试

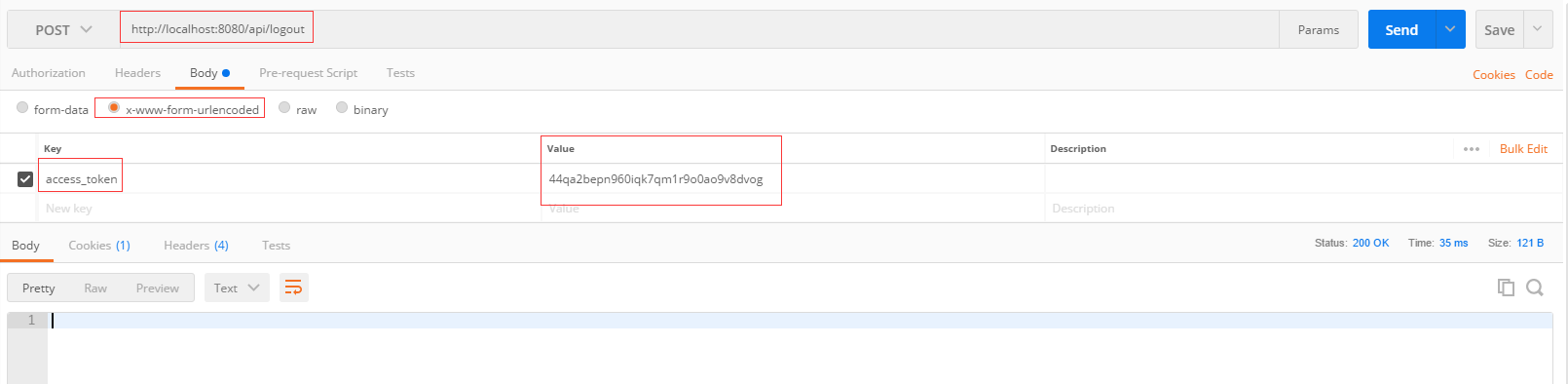
7.1

首先测试登录，url:<http://localhost:8080/api/login>

7.2 验证token是否有效，url为：<http://localhost:8080/api/validate>，参数key为Authorization，value为：Bearer+空格+access\_token，空格一定不能丢



7.3 测试注销登录，url为：<http://localhost:8080/api/logout>



7.4 测试接口数据返回，url: <http://localhost:8080/product/list>

