## 分布式Session一致性问题

### 什么是Session

Session 是客户端与服务器通讯会话技术， 比如浏览器登陆、记录整个浏览会话信息

### Session实现原理

客户对向服务器端发送请求后，Session 创建在服务器端，返回Sessionid给客户端浏览器保存在本地，当下次发送请求的时候，在请求头中传递sessionId获取对应的从服务器上获取对应的Sesison

### Session常见问题

#### Session 保证在那里？

答案：存放在服务器上

#### 关闭浏览器Session会失效吗

答案：不会消失

#### 相关代码

|  |
| --- |
| @SpringBootApplication  @RestController  **public** **class** TestSessionController {  // 创建session 会话  @RequestMapping("/createSession")  **public** String createSession(HttpServletRequest request, String nameValue) {  HttpSession session = request.getSession();  System.***out***.println("存入Session sessionid:信息" + session.getId() + ",nameValue:" + nameValue);  session.setAttribute("name", nameValue);  **return** "success";  }  // 获取session 会话  @RequestMapping("/getSession")  **public** Object getSession(HttpServletRequest request) {  HttpSession session = request.getSession();  System.***out***.println("获取Session sessionid:信息" + session.getId());  Object value = session.getAttribute("name");  **return** value;  }  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(TestSessionController.**class**, args);  }  } |

### 服务集群会产生那些问题

如果服务器产生了集群后，因为session是存放在服务器上，客户端会使用同一个Sessionid在多个不同的服务器上获取对应的Session，从而会导致Session不一致问题。

#### Nginx配置负载均衡

Nginx负载均衡提供上游服务器(真实业务逻辑访问的服务器),负载均衡、故障转移、失败重试、容错、健康检查等。

当上游服务器(真实业务逻辑访问的服务器)发生故障时，可以转移到其他上游服务器(真实业务逻辑访问的服务器)。

#### Upstream Server配置

upstream 主要配置如下:

IP地址和端口号：配置上游服务器的IP地址和端口

|  |
| --- |
| ###定义上游服务器(需要被nginx真实代理访问的服务器) 默认是轮训机制  upstream backServer{  server 127.0.0.1:8080;  server 127.0.0.1:8081;  }    server {  listen 80;  server\_name www.itmayiedu.com;  location / {  ### 指定上游服务器负载均衡服务器  proxy\_pass http://backServer;  index index.html index.htm;  }  } |

#### 负载均衡算法

1、轮询（默认）

每个请求按时间顺序逐一分配到不同的后端服务，如果后端某台服务器死机，自动剔除故障系统，使用户访问不受影响。

2、weight（轮询权值）

weight的值越大分配到的访问概率越高，主要用于后端每台服务器性能不均衡的情况下。或者仅仅为在主从的情况下设置不同的权值，达到合理有效的地利用主机资源。

3、ip\_hash

每个请求按访问IP的哈希结果分配，使来自同一个IP的访客固定访问一台后端服务器，并且可以有效解决动态网页存在的session共享问题。俗称IP绑定。

4、fair（第三方）

比 weight、ip\_hash更加智能的负载均衡算法，fair算法可以根据页面大小和加载时间长短智能地进行负载均衡，也就是根据后端服务器的响应时间 来分配请求，响应时间短的优先分配。Nginx本身不支持fair，如果需要这种调度算法，则必须安装upstream\_fair模块。

5、url\_hash(第三方)

按访问的URL的哈希结果来分配请求，使每个URL定向到一台后端服务器，可以进一步提高后端缓存服务器的效率。Nginx本身不支持url\_hash，如果需要这种调度算法，则必须安装Nginx的hash软件包。

|  |
| --- |
| @SpringBootApplication  @RestController  **public** **class** TestSessionController {  @Value("${server.port}")  **private** String serverPort;  @RequestMapping("/")  **public** String index() {  **return** serverPort;  }  // 创建session 会话  @RequestMapping("/createSession")  **public** String createSession(HttpServletRequest request, String nameValue) {  HttpSession session = request.getSession();  System.***out***.println(  "存入Session sessionid:信息" + session.getId() + ",nameValue:" + nameValue + ",serverPort:" + serverPort);  session.setAttribute("name", nameValue);  **return** "success-" + serverPort;  }  // 获取session 会话  @RequestMapping("/getSession")  **public** Object getSession(HttpServletRequest request) {  HttpSession session = request.getSession(**false**);  **if** (session == **null**) {  **return** serverPort + "-" + "没有找到对应的session值";  }  System.***out***.println("获取Session sessionid:信息" + session.getId() + "serverPort:" + serverPort);  Object value = session.getAttribute("name");  **return** serverPort + "-" + value;  }  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(TestSessionController.**class**, args);  }  } |

### 分布式Session一致性解决方案

#### nginx或者haproxy实现IP绑定

用Nginx 做的负载均衡可以添加ip\_hash这个配置，

用haproxy做的负载均衡可以用 balance source这个配置。

从而使同一个ip的请求发到同一台服务器。

#### 利用数据库同步session

#### 使用Session集群存放Redis

使用spring-session框架，底层实现原理是重写httpsession

##### 引入maven依赖

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <properties>  <weixin-java-mp.version>2.8.0</weixin-java-mp.version>  <maven.compiler.source>1.8</maven.compiler.source>  <maven.compiler.target>1.8</maven.compiler.target>  <maven.compiler.encoding>UTF-8</maven.compiler.encoding>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.build.locales>zh\_CN</project.build.locales>  </properties>  <dependencies>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  <!-- <exclusions> <exclusion> <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId> </exclusion> </exclusions> -->  </dependency>  <!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.47</version>  </dependency>  <!-- Testing Dependencies -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-redis</artifactId>  </dependency>  <!--spring session 与redis应用基本环境配置,需要开启redis后才可以使用，不然启动Spring boot会报错 -->  <dependency>  <groupId>org.springframework.session</groupId>  <artifactId>spring-session-data-redis</artifactId>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-pool2</artifactId>  </dependency>  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <configuration>  <maimClass>com.meiteedu.WxMpApplication</maimClass>  </configuration>  <executions>  <execution>  <goals>  <goal>repackage</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build> |

##### YML配置信息

|  |
| --- |
| server:  port: 8080  redis:  hostname: 192.168.212.151  port: 6379  password: 123456 |

启动redis /usr/local/redis/bin/redis-server /usr/local/redis/etc/redis.conf

##### 创建SessionConfig

|  |
| --- |
| **import org.springframework.beans.factory.annotation.Value;**  **import org.springframework.context.annotation.Bean;**  **import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;**  **import org.springframework.session.data.redis.config.annotation.web.http.EnableRedisHttpSession;**  **//这个类用配置redis服务器的连接**  **//maxInactiveIntervalInSeconds为SpringSession的过期时间（单位：秒）**  **@EnableRedisHttpSession(maxInactiveIntervalInSeconds = 1800)**  **public class SessionConfig {**  **// 冒号后的值为没有配置文件时，制动装载的默认值**  **@Value("${redis.hostname:localhost}")**  **String HostName;**  **@Value("${redis.port:6379}")**  **int Port;**  **@Bean**  **public JedisConnectionFactory connectionFactory() {**  **JedisConnectionFactory connection = new JedisConnectionFactory();**  **connection.setPort(Port);**  **connection.setHostName(HostName);**  **return connection;**  **}**  **}** |

##### 初始化Session

|  |
| --- |
| **//初始化Session配置**  **public class SessionInitializer extends AbstractHttpSessionApplicationInitializer{**  **public SessionInitializer() {**  **super(SessionConfig.class);**  **}**  **}** |

#### 最靠谱的分布式Session解决方案

基于令牌（Token）方式实现Session解决方案，因为Session本身就是分布式共享连接。

|  |
| --- |
| @Service  **public** **class** TokenService {  @Autowired  **private** RedisService redisService;  // 新增 返回token  **public** String put(Object object) {  String token = getToken();  redisService.setString(token, object);  **return** token;  }  // 获取信息  **public** String get(String token) {  String reuslt = redisService.getString(token);  **return** reuslt;  }  **public** String getToken() {  **return** UUID.*randomUUID*().toString();  }  } |

TokenController

|  |
| --- |
| @RestController  **public** **class** TokenController {  @Autowired  **private** TokenService tokenService;  @Value("${server.port}")  **private** String serverPort;  @RequestMapping("/put")  **public** String put(String nameValue) {  String token = tokenService.put(nameValue);  **return** token + "-" + serverPort;  }  @RequestMapping("/get")  **public** String get(String token) {  String value = tokenService.get(token);  **return** value + "-" + serverPort;  }  } |

## 网站跨域解决方案

### 什么是网站跨域

跨域原因产生：在当前域名请求网站中，默认不允许通过ajax请求发送其他域名。

### 网站跨域报错案例

jquery-1.7.2.min.js?t=2017-07-27:4 Failed to load http://b.itmayiedu.com:8081/ajaxB: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://a.itmayiedu.com:8080' is therefore not allowed access.

### 五种网站跨域解决方案

1. 使用jsonp解决网站跨域   
    2.使用HttpClient内部转发  
    3.使用设置响应头允许跨域  
    4.基于Nginx搭建企业级API接口网关  
    5.使用Zuul搭建微服务API接口网关

### 跨域项目环境搭建

#### 使用JSONP解决网站跨域

##### 前端代码

|  |
| --- |
| <script type=*"text/javascript"*  src=*"http://www.itmayiedu.com/static/common/jquery-1.7.2.min.js?t=2017-07-27"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $.ajax({  type : "GET",  async : **false**,  url : "http://b.itmayiedu.com:8081/ajaxB",  dataType : "jsonp",  jsonp : "jsonpCallback",//服务端用于接收callback调用的function名的参数  success : **function**(data) {  alert(data["errorCode"]);  },  error : **function**() {  alert('fail');  }  });  });  </script> |

##### 后端代码

|  |
| --- |
| @RequestMapping(value = "/ajaxB", method = RequestMethod.***GET***)  **public** **void** ajaxB(HttpServletResponse response, String jsonpCallback) **throws** IOException {  JSONObject root = **new** JSONObject();  root.put("errorCode", "200");  root.put("errorMsg", "登陆成功");  response.setHeader("Content-type", "text/html;charset=UTF-8");  PrintWriter writer = response.getWriter();  writer.print(jsonpCallback + "(" + root.toString() + ")");  writer.close();  } |

缺点：不支持post请求，代码书写比较复杂

#### 使用设置响应头允许跨域

##### 前端代码

|  |
| --- |
| <script type=*"text/javascript"*  src=*"http://www.itmayiedu.com/static/common/jquery-1.7.2.min.js?t=2017-07-27"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $.ajax({  type : "GET",  async : **false**,  url : "http://b.itmayiedu.com:8081/ajaxB",  dataType : "json",  success : **function**(data) {  alert(data["errorCode"]);  },  error : **function**() {  alert('fail');  }  });  });  </script> |

##### 后端代码

|  |
| --- |
| @RequestMapping("/ajaxB")  **public** Map<String, Object> ajaxB(HttpServletResponse response) {  response.setHeader("Access-Control-Allow-Origin", "\*");  Map<String, Object> result = **new** HashMap<String, Object>();  result.put("errorCode", "200");  result.put("errorMsg", "登陆成功");  **return** result;  } |

response.setHeader("Access-Control-Allow-Origin", "\*"); 设置响应头允许跨域

如果在实际项目中，该代码建议放在过滤器中。

#### 使用HttpClient进行内部转发

##### 前端代码

|  |
| --- |
| <script type=*"text/javascript"*  src=*"http://www.itmayiedu.com/static/common/jquery-1.7.2.min.js?t=2017-07-27"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $.ajax({  type : "POST",  async : **false**,  url : "http://a.itmayiedu.com:8080/forwardB",  dataType : "json",  success : **function**(data) {  alert(data["errorCode"]);  },  error : **function**() {  alert('fail');  }  });  });  </script> |

##### 后端代码

A项目进行转发到B项目

|  |
| --- |
| @RequestMapping("/forwardB")  @ResponseBody  **public** JSONObject forwardB() {  JSONObject result = HttpClientUtils.*httpGet*("http://b.itmayiedu.com:8081/ajaxB");  System.***out***.println("result:" + result);  **return** result;  } |

B项目代码

|  |
| --- |
| @RequestMapping("/ajaxB")  **public** Map<String, Object> ajaxB(HttpServletResponse response) {  response.setHeader("Access-Control-Allow-Origin", "\*");  Map<String, Object> result = **new** HashMap<String, Object>();  result.put("errorCode", "200");  result.put("errorMsg", "登陆成功");  **return** result;  } |

#### 搭建企业级API接口网关

##### 使用Nginx搭建API接口网关

###### Nginx相关配置

|  |
| --- |
| server {  listen 80;  server\_name www.itmayiedu.com;  ###A项目  location /a {  proxy\_pass http://a.itmayiedu.com:8080/;  index index.html index.htm;  }  ###B项目  location /b {  proxy\_pass http://b.itmayiedu.com:8081/;  index index.html index.htm;  }  } |

###### 前端代码

|  |
| --- |
| <script type=*"text/javascript"*  src=*"http://code.jquery.com/jquery-1.8.0.min.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $.ajax({  type : "POST",  async : **false**,  url : "http://www.itmayiedu.com/b/ajaxB",  dataType : "json",  success : **function**(data) {  alert(data["errorCode"]);  },  error : **function**() {  alert('fail');  }  });  });  </script> |

###### 后端代码

|  |
| --- |
| @RequestMapping("/ajaxB")  **public** Map<String, Object> ajaxB(HttpServletResponse response) {  response.setHeader("Access-Control-Allow-Origin", "\*");  Map<String, Object> result = **new** HashMap<String, Object>();  result.put("errorCode", "200");  result.put("errorMsg", "登陆成功");  **return** result;  } |

##### SpringCloud搭建API接口网关

使用SpringCloud Zuul搭建API接口网关

#### Maven初始化依赖参数

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  </parent>  <dependencies>  <!-- SpringBoot 对lombok 支持 -->  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  </dependency>  <!-- SpringBoot web 核心组件 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  </dependency>  <!-- SpringBoot 外部tomcat支持 -->  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency>  <!-- springboot-log4j -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-log4j</artifactId>  <version>1.3.8.RELEASE</version>  </dependency>  <!-- springboot-aop 技术 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-aop</artifactId>  </dependency>  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  </dependency>  <!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.47</version>  </dependency>  </dependencies> |

#### application.yml

|  |
| --- |
| server:  port: 8080  spring:  mvc:  view:  prefix: /WEB-INF/jsp/  suffix: .jsp |

#### HttpClientUtils工具类

|  |
| --- |
| **public** **class** HttpClientUtils {  **private** **static** Logger *logger* = LoggerFactory.*getLogger*(HttpClientUtils.**class**); // 日志记录  **private** **static** RequestConfig *requestConfig* = **null**;  **static** {  // 设置请求和传输超时时间  *requestConfig* = RequestConfig.*custom*().setSocketTimeout(2000).setConnectTimeout(2000).build();  }  /\*\*  \* post请求传输json参数  \*  \* **@param** url  \* url地址  \* **@param** json  \* 参数  \* **@return**  \*/  **public** **static** JSONObject httpPost(String url, JSONObject jsonParam) {  // post请求返回结果  CloseableHttpClient httpClient = HttpClients.*createDefault*();  JSONObject jsonResult = **null**;  HttpPost httpPost = **new** HttpPost(url);  // 设置请求和传输超时时间  httpPost.setConfig(*requestConfig*);  **try** {  **if** (**null** != jsonParam) {  // 解决中文乱码问题  StringEntity entity = **new** StringEntity(jsonParam.toString(), "utf-8");  entity.setContentEncoding("UTF-8");  entity.setContentType("application/json");  httpPost.setEntity(entity);  }  CloseableHttpResponse result = httpClient.execute(httpPost);  // 请求发送成功，并得到响应  **if** (result.getStatusLine().getStatusCode() == HttpStatus.***SC\_OK***) {  String str = "";  **try** {  // 读取服务器返回过来的json字符串数据  str = EntityUtils.*toString*(result.getEntity(), "utf-8");  // 把json字符串转换成json对象  jsonResult = JSONObject.*parseObject*(str);  } **catch** (Exception e) {  *logger*.error("post请求提交失败:" + url, e);  }  }  } **catch** (IOException e) {  *logger*.error("post请求提交失败:" + url, e);  } **finally** {  httpPost.releaseConnection();  }  **return** jsonResult;  }  /\*\*  \* post请求传输String参数 例如：name=Jack&sex=1&type=2  \* Content-type:application/x-www-form-urlencoded  \*  \* **@param** url  \* url地址  \* **@param** strParam  \* 参数  \* **@return**  \*/  **public** **static** JSONObject httpPost(String url, String strParam) {  // post请求返回结果  CloseableHttpClient httpClient = HttpClients.*createDefault*();  JSONObject jsonResult = **null**;  HttpPost httpPost = **new** HttpPost(url);  httpPost.setConfig(*requestConfig*);  **try** {  **if** (**null** != strParam) {  // 解决中文乱码问题  StringEntity entity = **new** StringEntity(strParam, "utf-8");  entity.setContentEncoding("UTF-8");  entity.setContentType("application/x-www-form-urlencoded");  httpPost.setEntity(entity);  }  CloseableHttpResponse result = httpClient.execute(httpPost);  // 请求发送成功，并得到响应  **if** (result.getStatusLine().getStatusCode() == HttpStatus.***SC\_OK***) {  String str = "";  **try** {  // 读取服务器返回过来的json字符串数据  str = EntityUtils.*toString*(result.getEntity(), "utf-8");  // 把json字符串转换成json对象  jsonResult = JSONObject.*parseObject*(str);  } **catch** (Exception e) {  *logger*.error("post请求提交失败:" + url, e);  }  }  } **catch** (IOException e) {  *logger*.error("post请求提交失败:" + url, e);  } **finally** {  httpPost.releaseConnection();  }  **return** jsonResult;  }  /\*\*  \* 发送get请求  \*  \* **@param** url  \* 路径  \* **@return**  \*/  **public** **static** JSONObject httpGet(String url) {  // get请求返回结果  JSONObject jsonResult = **null**;  CloseableHttpClient client = HttpClients.*createDefault*();  // 发送get请求  HttpGet request = **new** HttpGet(url);  request.setConfig(*requestConfig*);  **try** {  CloseableHttpResponse response = client.execute(request);  // 请求发送成功，并得到响应  **if** (response.getStatusLine().getStatusCode() == HttpStatus.***SC\_OK***) {  // 读取服务器返回过来的json字符串数据  HttpEntity entity = response.getEntity();  String strResult = EntityUtils.*toString*(entity, "utf-8");  // 把json字符串转换成json对象  jsonResult = JSONObject.*parseObject*(strResult);  } **else** {  *logger*.error("get请求提交失败:" + url);  }  } **catch** (IOException e) {  *logger*.error("get请求提交失败:" + url, e);  } **finally** {  request.releaseConnection();  }  **return** jsonResult;  }  } |

#### Jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  <script type=*"text/javascript"*  src=*"http://code.jquery.com/jquery-1.8.0.min.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $.ajax({  type : "POST",  async : **false**,  url : "http://www.itmayiedu.com/b/ajaxB",  dataType : "json",  success : **function**(data) {  alert(data["errorCode"]);  },  error : **function**() {  alert('fail');  }  });  });  </script>  </head>  <body>显示 ....  </body>  </html> |

## 分布式锁解决方案

## 分布式任务调度平台

## 分布式日志收集系统

## 分布式事务解决方案