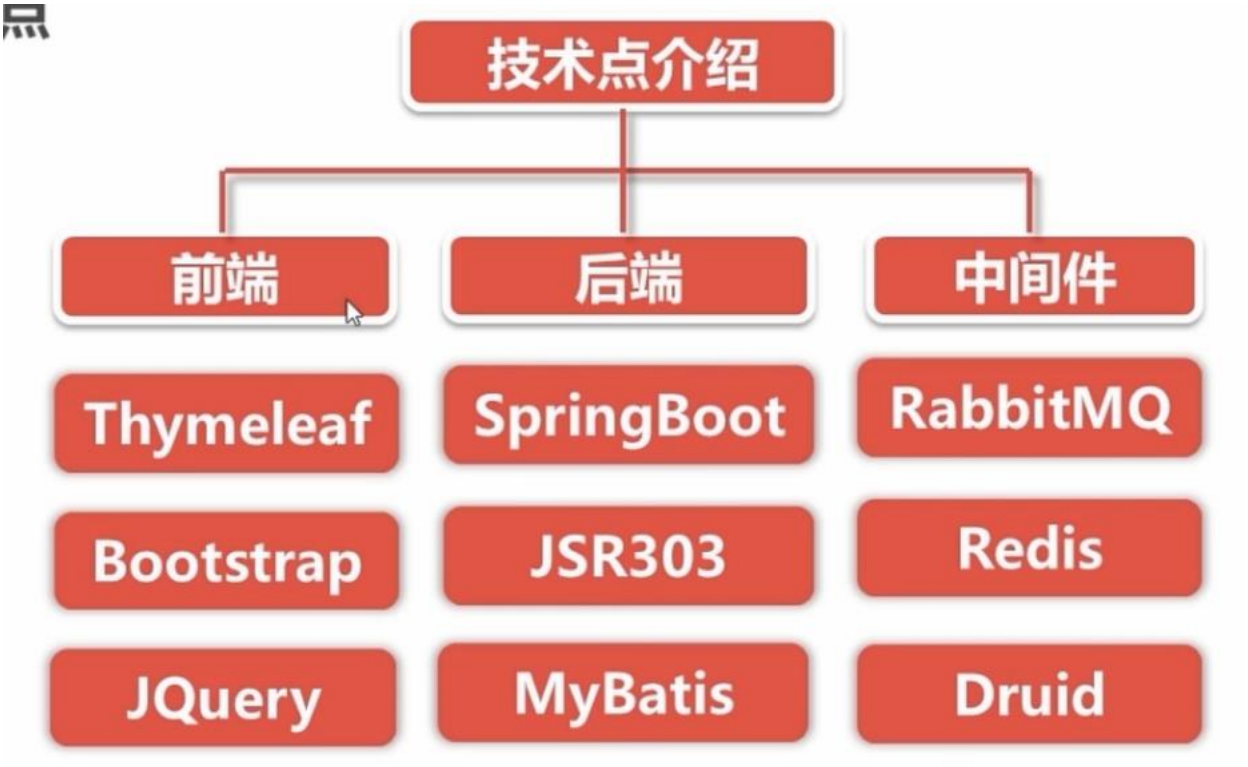

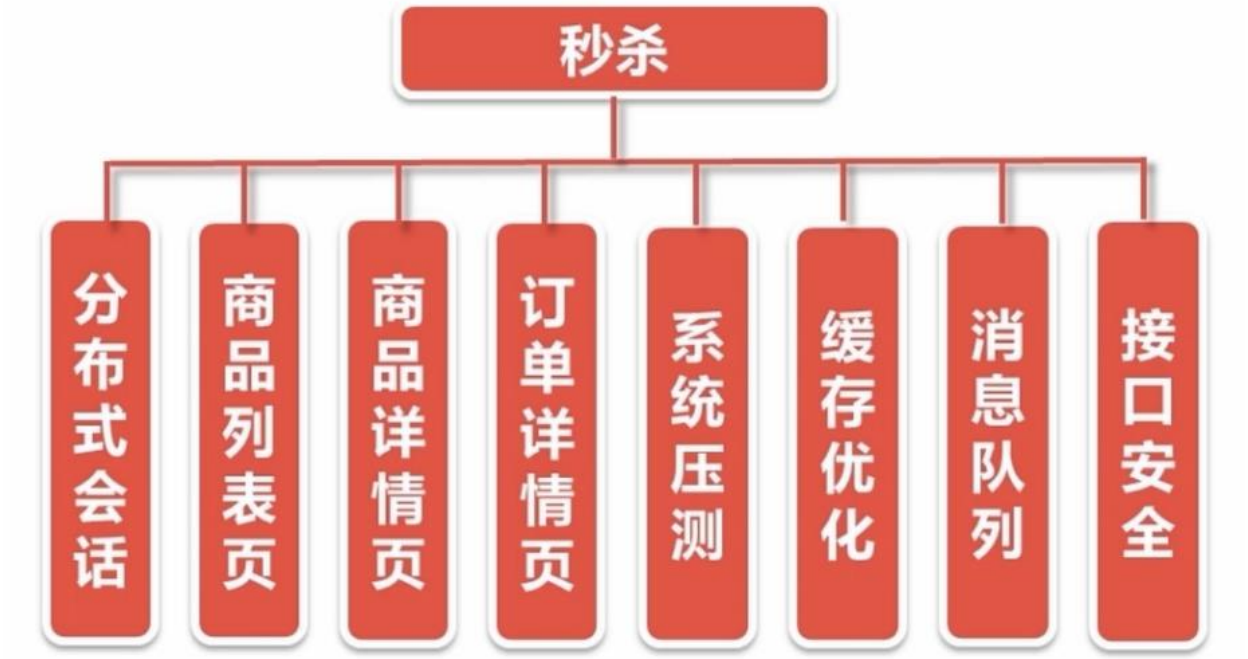

应对高并发的思路：

1、如何利用缓存

2、如何使用异步

application.yml：

```yaml
spring:
  thymeleaf:
    cache: false
    servlet:
      content-type: text/html
    enabled: true
    encoding: UTF-8
    mode: HTML5
    prefix: classpath:/templates/
    suffix: .html
  # druid
  datasource:
    url: jdbc:mysql://localhost:3306/miaosha?useUnicode=true&characterEncoding=utf-8&allowMultiQueries=true&useSSL=false&serverTimezone=GMT%2B8&amp
    username: root
    password: 123
    type: com.alibaba.druid.pool.DruidDataSource
    filters: stat
    maxActive: 2
    initialSize: 1
    maxWait: 60000
    minIdle: 1
    timeBetweenEvictionRunsMillis: 60000
    minEvictableIdleTimeMillis: 300000
    validationQuery: select 'x'
    testWhileIdle: true
    testOnBorrow: false
    testOnReturn: false
    poolPreparedStatements: true
    maxOpenPreparedStatements: 20
    driver-class-name: com.mysql.cj.jdbc.Driver
mybatis:
  type-aliases-package: com.imooc.miaosha.domain
  configuration:
    map-underscore-to-camel-case: true
    default-fetch-size: 100
    default-statement-timeout: 3000
  mapperLocations: classpath:com/imooc/miaosha/dao/*.xml
#redis
redis:
```

```
  host: 192.168.152.132
  timeout: 3
  password: 123456
  poolMaxTotal: 10
  poolMaxIdle: 10
  poolMaxWait: 3
```

pom:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.8.RELEASE</version>
</parent>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

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

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>1.3.1</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.11</version>
    </dependency>

    <dependency>
```

```
        <groupId>com.alibaba</groupId>
        <artifactId>druid</artifactId>
        <version>1.0.5</version>
    </dependency>

    <dependency>
        <groupId>redis.clients</groupId>
        <artifactId>jedis</artifactId>
    </dependency>

    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
        <version>1.2.38</version>
    </dependency>

    <dependency>
        <groupId>commons-codec</groupId>
        <artifactId>commons-codec</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.6</version>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.16.22</version>
    </dependency>
</dependencies>
```

一、

对返回结果的封装 返回的对象都是result类　其中成员变量包括 msg code 以及data

二、对redis的集成

1、propreties读取配合文件的参数数据

```
@Component
@ConfigurationProperties(prefix="redis")
public class RedisConfig {
    private String host;
    private int port;
    private int timeout;//秒
    private String password;
    private int poolMaxTotal;
    private int poolMaxIdle;
    private int poolMaxWait;//秒
```

2、将RedisPool 注入spring容器

```
@Configuration
public class RedisPoolFactory {

    @Autowired
    RedisConfig redisConfig;

    @Bean
    public JedisPool JedisPoolFactory() {
        JedisPoolConfig poolConfig = new JedisPoolConfig();
        poolConfig.setMaxIdle(redisConfig.getPoolMaxIdle());
        poolConfig.setMaxTotal(redisConfig.getPoolMaxTotal());
        poolConfig.setMaxWaitMillis(redisConfig.getPoolMaxWait() * 1000);
        JedisPool jp = new JedisPool(poolConfig, redisConfig.getHost(), redisConfig.getPort(),
                timeout: redisConfig.getTimeout()*1000, redisConfig.getPassword(), database: 0);
        return jp;
    }
```

3、前缀的添加以及存活时间的设置

```
public interface KeyPrefix {

    public int expireSeconds();

    public String getPrefix();

}
```

抽象类BasePrefix类 实现该接口， 由其他的业务类型继承BasePrefix,以设置不同的前缀，默认存在时长为永久

4、RedisService 完成对redis数据库的各种操作

```java
@Service
public class RedisService {

    @Autowired
    JedisPool jedisPool;

    /**
     * 获取当个对象
     * */
    public <T> T get(KeyPrefix prefix, String key,  Class<T> clazz) {
        Jedis jedis = null;
        try {
            jedis =  jedisPool.getResource();
            //生成真正的key
            String realKey  = prefix.getPrefix() + key;
            String  str = jedis.get(realKey);
            T t =  stringToBean(str, clazz);
            return t;
```

剩下的方法略