测试用例生成器集成

1. 准备工作

1.1 前置环境准备——安装Squaretest插件

基于Squaretest插件生成代码，需要先安装好相应的插件。

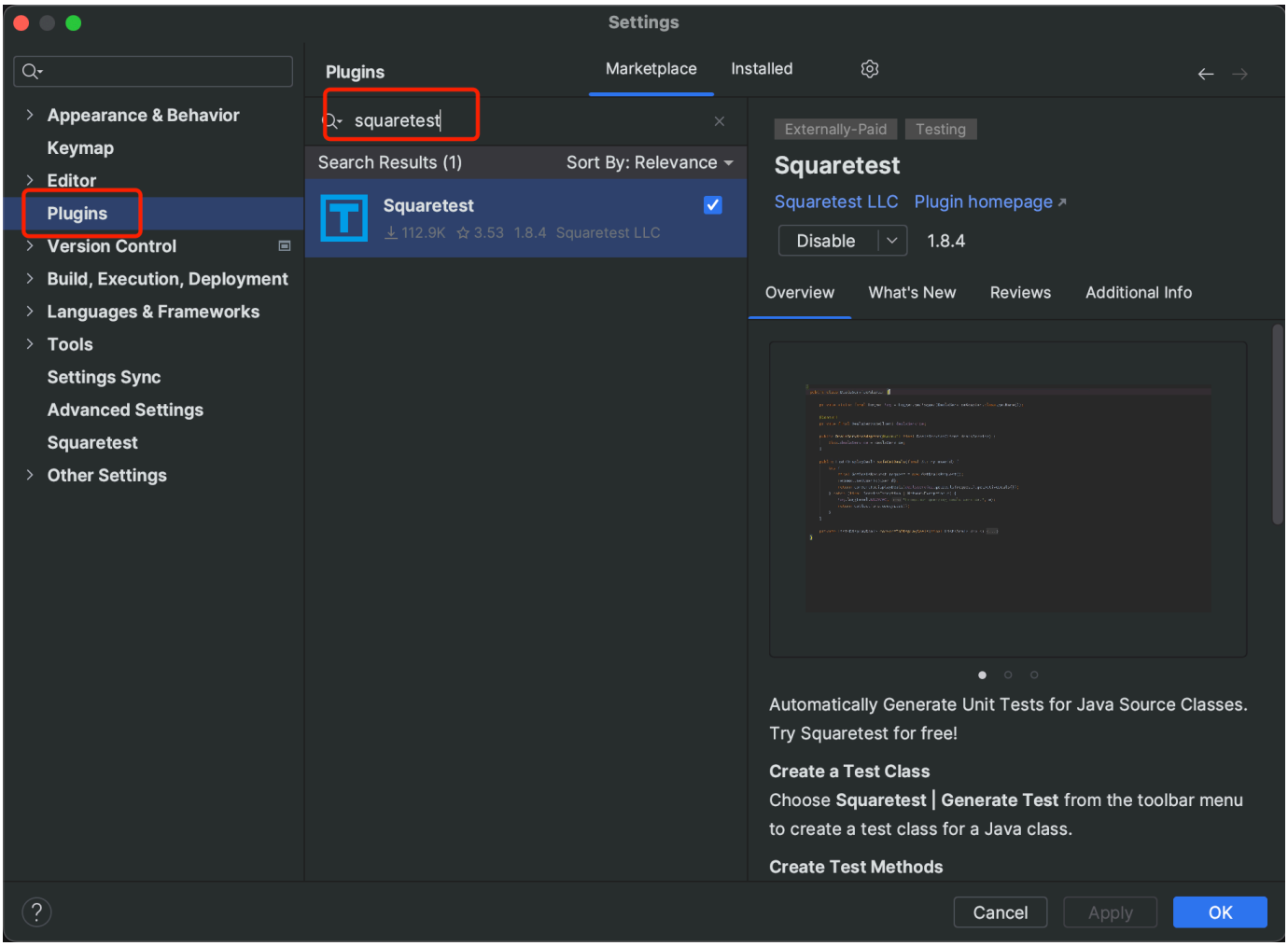
1.2 开发工具

本系统采用IDEA作为开发工具。

2. IDEA环境配置

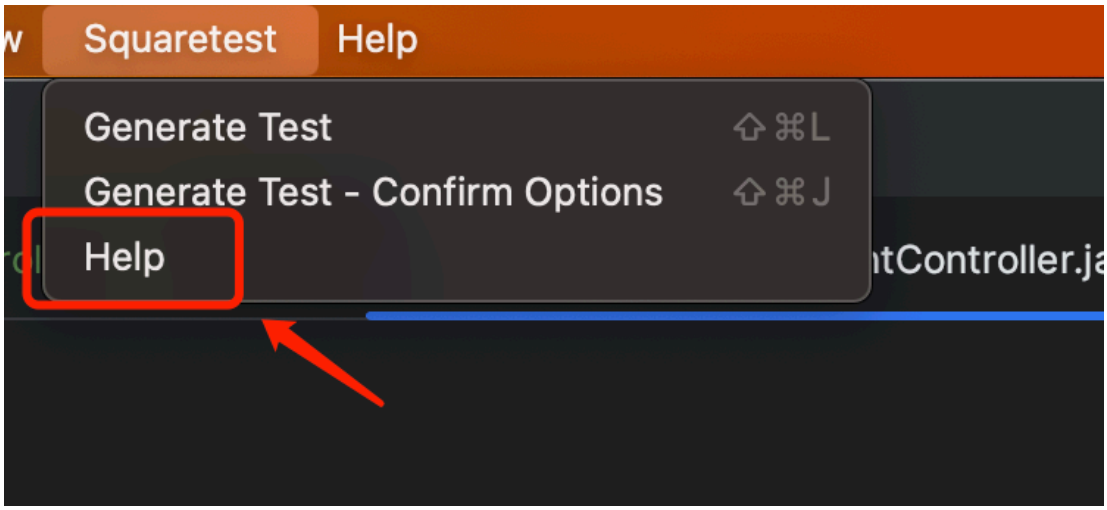
2.1 Squaretest插件安装

`File—>Settings—>PluginsSquaretest`



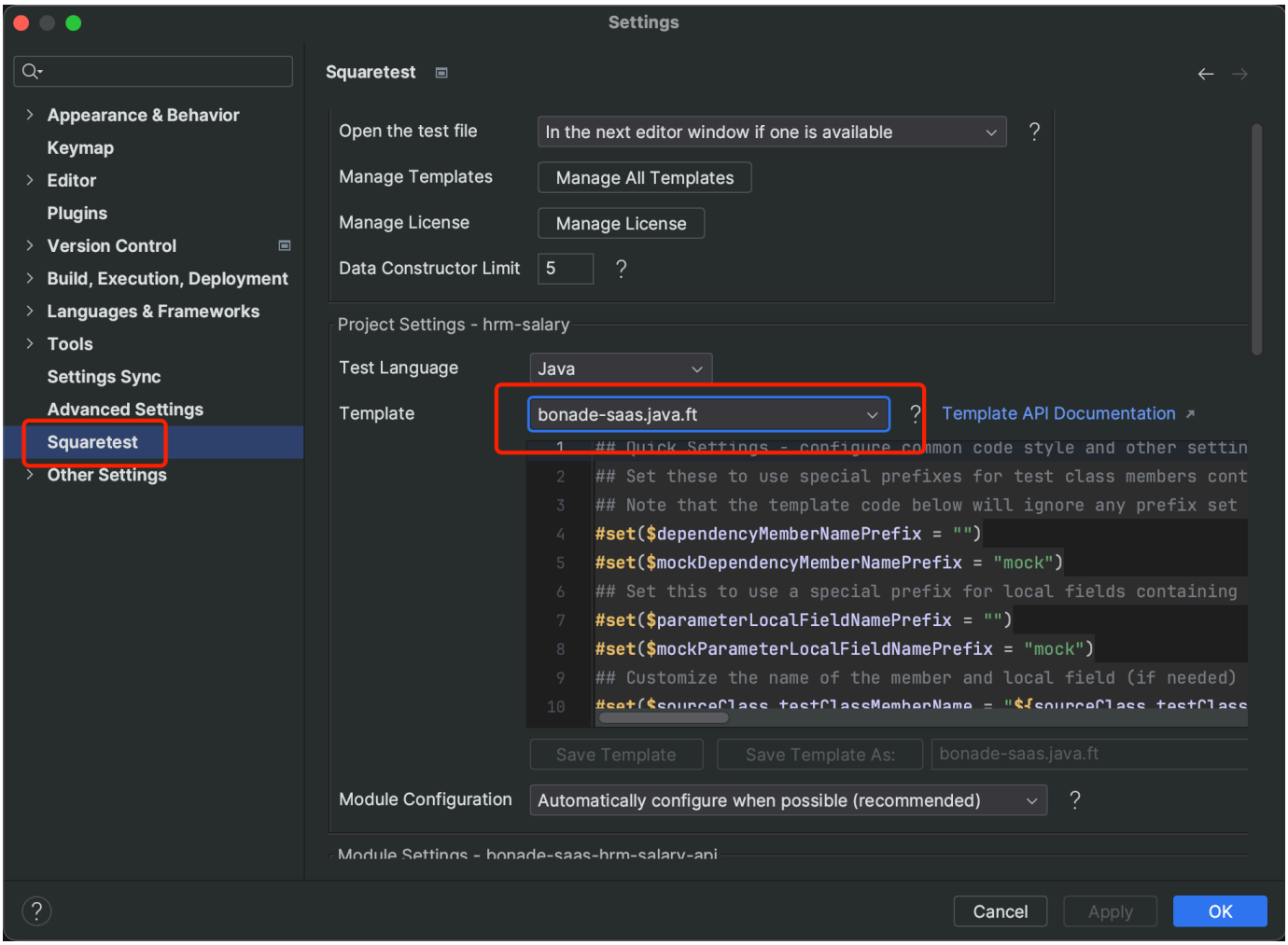
2.2 配置使用

SquaretestHelp

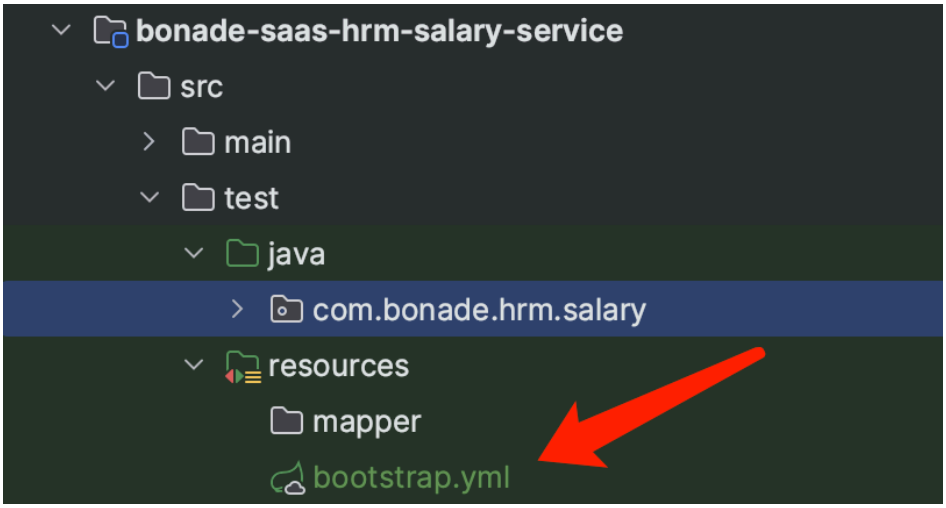


2.3 导入模板

File—>Settings—>Squaretest，选择右边的Template，导入模板：bonade-saas-template.java.ft

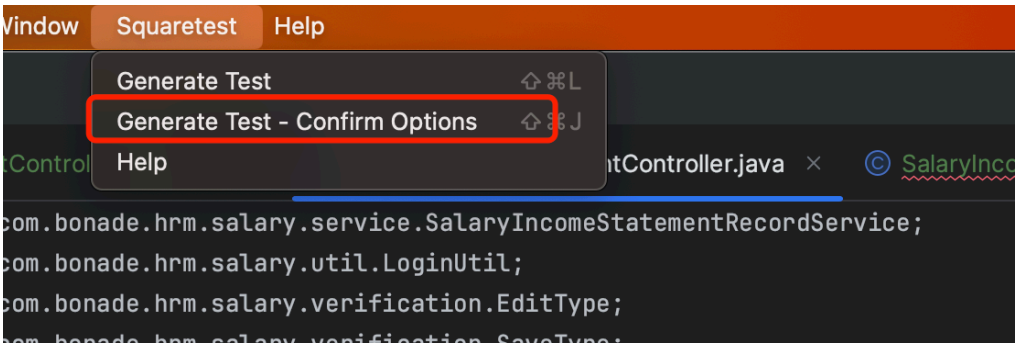


配置对应的nacos环境：bootstrap.yml

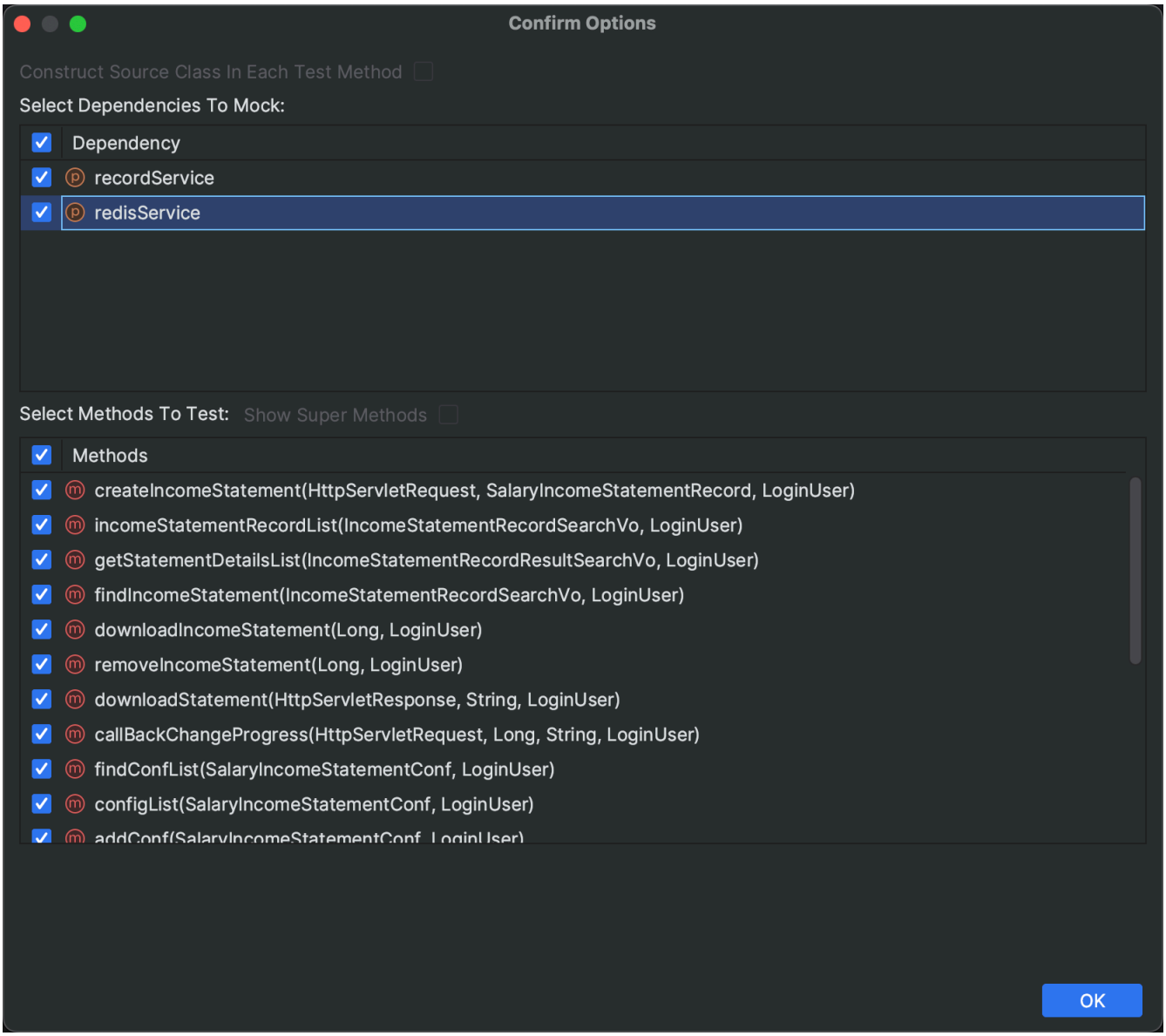


3. 生成单元测试代码

```
-->Squaretest-->Generate Test - Confirm Options
```



3.1 按需求选择要生成的内容



3.2 生成对应的需要测试的内容

```java
new *
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
public class SalaryIncomeStatementControllerTest {
    @Autowired
    private MockMvc mockMvc;
    @MockBean
    private SalaryIncomeStatementRecordService mockRecordService;
    @MockBean
    private RedisService mockRedisService;
    @Autowired
    private WebApplicationContext webApplicationContext;
    34 usages
    private static final String  TENANT_ID = "";
    2 usages
    private static final String  AUTHORIZATION = "";
    new *
    @Before
    public void setUp(){
        mockMvc =  MockMvcBuilders.webAppContextSetup(webApplicationContext).addFilter((request, response, chain) -> {
                    MockHttpServletRequest mq = (MockHttpServletRequest)request;
                    mq.addHeader(TENANT_ID, TENANT_ID);
                    mq.addHeader( name: "authorization", AUTHORIZATION);
                    response.setCharacterEncoding("UTF-8");
                    chain.doFilter(request, response);
                }, ...urlPatterns: "/*").build();
        // mysql租戶Id
        TenantContextHolder.setTenantId(TENANT_ID);
        // 设置租户数据源
        TenantDataSourceProvider tenantDataSourceProvider = webApplicationContext.getBean(TenantDataSourceProvider.class);
        tenantDataSourceProvider.pushDs(TENANT_ID);
        // 设置租户mongodb源
        TenantMongoContextHolder.push(TENANT_ID);
    }

    new *
```

3.3 生成代码后，调整对应的断言代码

```
new *
@Test
public void testIncomeStatementRecordList() throws Exception {
    // Setup
    // Configure SalaryIncomeStatementRecordService.incomeStatementRecordList(...).
    final SalaryIncomeStatementRecordVo salaryIncomeStatementRecordVo = new SalaryIncomeStatementRecordVo();
    salaryIncomeStatementRecordVo.setId(0L);
    salaryIncomeStatementRecordVo.setTenantId("tenantId");
    salaryIncomeStatementRecordVo.setTemplateId(0L);
    salaryIncomeStatementRecordVo.setDataConfId("dataConfId");
    salaryIncomeStatementRecordVo.setVersions(Arrays.asList("value"));
    final Result<List<SalaryIncomeStatementRecordVo>> listResult = Result.data(
            Arrays.asList(salaryIncomeStatementRecordVo));
    final IncomeStatementRecordSearchVo vo = new IncomeStatementRecordSearchVo();
    vo.setId(0L);
    vo.setTenantId("tenantId");
    vo.setProcessType(0);
    vo.setClassifyIds(Arrays.asList(0L));
    vo.setSpecialIds(Arrays.asList(0L));
    when(mockRecordService.incomeStatementRecordList(vo)).thenReturn(listResult);

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(
                    post( urlTemplate: "/hrm-salary/incomeStatementRecord/v1.0/incomeStatementRecordList")
                            .content("content").contentType(MediaType.APPLICATION_JSON)
                            .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo( expected: "expectedResponse");
}

new *
```

调整正确的业务数据值

调整正确的预期返回状态和返回数据断方