산업인공지능 개론

Mini Project #1
Rule-based System 만들기

학과 : 산업인공지능학과
학번 : 2024254022
이름 : 정현일

2024.03.31.

서비스 점검

1. 웹 서버 장애

1.1. 웹서비스 500에러 - 웹서버 로그 확인
1.2. 웹서버 접속 오류 - Disk 사용량 확인
1.3. 서비스 접속 장애 - mod-jk 로그 확인

2. WAS 서버 장애

2.1. WAS 서비스 장애 - WAS서버 로그 확인
2.2. WAS heapdump 장애 - 메모리 사용량 확인
3.3. 서비스 느림 - CPU 사용량 확인

3. DB 서버 장애

3.1. DB서버 접속 오류 - DB서버 로그 확인
3.2. SQL 속도 느림 - SQL hint 확인
3.3. DB 서비스 오류 - DB 세션 확인

4. 서비스 장애

4.1. 웹서버 접속 안됨 - 웹서버 장애 확인
4.2. WAS 서버 접속 안됨 - WAS 서버 장애 확인
4.3. DB 서버 접속 안됨 - DB 장애 확인

5. 방화벽 장애

5.1. 내부망 접속 오류 - 기관 방화벽 확인
5.2. 외부망 접속 오류 - 전산망 방화벽 확인
5.3. 네크워크 접속 안됨 - DNS 서버 확인

소스코드 (1/2)

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
산업인지능 개론

Mini Project #1 Rule-based System 만들기
자신의 현업에서 특정 분야의 문제에 대해 15개 이상의 규칙을 찾아서 Durable_Rules 로 구현

학번 : 2024254022
이름 : 정현일


@author: chohi
"""

from durable.lang import *


with ruleset('Inspection'):
    @when_all((m.predicate == '웹서버') & (m.error == '500에러'))
    def check1(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '웹서버', 'error': '로그 확인한다.' })

    @when_all((m.predicate == '웹서버') & (m.error == '접속오류'))
    def check2(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '웹서버', 'error': '디스크 사용량 확인' })

    @when_all((m.predicate == '웹서버') & (m.error == '서비스오류'))
    def check3(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '웹서버', 'error': 'mod_jk 확인' })

    @when_all((m.predicate == 'WAS서버') & (m.error == '서비스오류'))
    def check4(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'WAS서버', 'error': '로그확인' })

    @when_all((m.predicate == 'WAS서버') & (m.error == 'headpdump'))
    def check5(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'WAS서버', 'error': '메모리 사용량 확인' })

    @when_all((m.predicate == 'WAS서버') & (m.error == '서비스느림'))
    def check6(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'WAS서버', 'error': 'CPU 사용량 확인' })

    @when_all((m.predicate == 'DB서버') & (m.error == '접속오류'))
    def check7(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'DB서버', 'error': '로그 확인' })

    @when_all((m.predicate == 'DB서버') & (m.error == '속도느림'))
    def check8(c):
```

소스코드 (1/2)

```
50      def check8(c):
51          c.assert_fact({ 'subject': c.m.subject, 'predicate': 'DB서버', 'error': 'SQL Hint 확인' })
52
53      @when_all((m.predicate == 'DB서버') & (m.error == '서비스오류'))
54      def check9(c):
55          c.assert_fact({ 'subject': c.m.subject, 'predicate': 'DB서버', 'error': 'DB세션 로그 확인' })
56
57      @when_all((m.predicate == '서비스장애') & (m.error == '웹서버'))
58      def check10(c):
59          c.assert_fact({ 'subject': c.m.subject, 'predicate': '서비스장애', 'error': '웹서버 장애확인' })
60
61      @when_all((m.predicate == '서비스장애') & (m.error == 'WAS서버'))
62      def check11(c):
63          c.assert_fact({ 'subject': c.m.subject, 'predicate': '서비스장애', 'error': 'was서버 장애확인'})
64
65      @when_all((m.predicate == '서비스장애') & (m.error == 'DB서버'))
66      def check12(c):
67          c.assert_fact({ 'subject': c.m.subject, 'predicate': '서비스장애', 'error': 'db서버 장애확인'})
68
69      @when_all((m.predicate == '방화벽') & (m.error == '내부망'))
70      def check13(c):
71          c.assert_fact({ 'subject': c.m.subject, 'predicate': '방화벽', 'error': '내부망 방화벽 확인' })
72
73      @when_all((m.predicate == '방화벽') & (m.error == '외부망'))
74      def check14(c):
75          c.assert_fact({ 'subject': c.m.subject, 'predicate': '방화벽', 'error': '외부망 방화벽 확인' })
76
77      @when_all((m.predicate == '방화벽') & (m.error == '접속안됨'))
78      def check15(c):
79          c.assert_fact({ 'subject': c.m.subject, 'predicate': '방화벽', 'error': '아이피 확인' })
80      @when_all(+m.subject)   # m.subject와 operation 한번 이상
81      def output(c):
82          print('Fact: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.error))
83
84
85  assert_fact('Inspection', { 'subject': '대민서비스', 'predicate': 'WAS서버', 'error': 'headpdump' })
86  assert_fact('Inspection', { 'subject': '대민서비스', 'predicate': '웹서버', 'error': '500에러' })
87  assert_fact('Inspection', { 'subject': '내부서비스', 'predicate': '웹서버', 'error': '접속오류' })
88  assert_fact('Inspection', { 'subject': '내부서비스', 'predicate': '방화벽', 'error': '내부망' })
89
```

실행 결과

```
In [1]: runfile('/Users/chohi/project/ai/Industrial artificial
intelligence/durable_rules.py', wdir='/Users/chohi/project/ai/
Industrial artificial intelligence')
Fact: 대민서비스 WAS서버 메모리 사용량 확인
Fact: 대민서비스 WAS서버 headpdump
Fact: 대민서비스 웹서버 로그 확인한다.
Fact: 대민서비스 웹서버 500에러
Fact: 내부서비스 웹서버 디스크 사용량 확인
Fact: 내부서비스 웹서버 접속오류
Fact: 내부서비스 방화벽 내부망 방화벽 확인
Fact: 내부서비스 방화벽 내부망

In [2]:
```