어플리케이션 개발

학습된 알고리즘 모델에 대한 검증 구현

- 합성곱 신경망과 장단기 기억 네트워크 알고리즘으로 학습된 모델을 불러옴
- 사용자가 선택한 알고리즘 및 테스트 문장을 입력받아 긍정/부정의 감정 결과를 도출
- Step 1) 변수 초기화
  - python 실행 위치
    - 웹어플리케이션에서 해당 파일 실행
    - python 파일의 절대 경로 기준으로 파일 위치 설정
  - word2vec 모델 파일 위치
  - 학습된 CNN, LSTM 알고리즘 저장 위치

```
7   # py 파일이 실행되는 폴더 위치
8   base_dir = os.path.dirname(os.path.realpath(__file__)) + '/'
9   source_dir ='./data/'
10  # word2vec 파일 이름
11  w2v_file_name = '3_word2vec_nsmc.w2v'
12  # 알고리즘 학습  모델 저장 경로
13  cnn_model_dir = './cnn_model'
14  lstm_model_dir = './lstm_model'
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 2) 사용자가 입력할 문장에 대한 전처리
  - 네이버 맞춤법 검사 -> 품사 부착 -> 단어에 대한 벡터 변환

```
16  # 사용자 입력 문장에 대해 단어 벡터 변환
17  def data_setting(w2v_model, embedding_dim, max_word_length, evaluation_text):
18      eval_arrays = np.zeros((1, max_word_length, embedding_dim))
19      # 네이버 맞춤법 검사 적용
20      eval_spell_chcker = pre.naver_spell_cheker(evaluation_text)
21      # 품사 부착 진행
22      eval_pos_tag = pre.konlpy_pos_tag(eval_spell_chcker)
23      #문장 내 단어 벡터 변환
24      eval_arrays[0] = pre.max_word_length_word2vec(w2v_model, embedding_dim, max_word_length, eval_pos_tag)
25
26      return eval_arrays
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 3) 학습된 알고리즘에 대한 검증 - 변수 초기화
  - CNN, LSTM 모델 학습 시 사용했던 문장 내 최대 단어 길이
    - 단어 길이를 다르게 설정하면 텐서 형태가 다르다는 에러 발생
  - 사용자가 입력한 문장
  - word2vec 모델에서의 vector 크기

```
28  def evaluation(params) :
29      # 각 알고리즘 별 최대 단어 개수 지정
30      # 모델 학습 시 사용했던 값 사용
31      cnn_max_sentence_length = 50
32      lstm_max_sentence_length = 100
33      evaluation_text = params[2]
34
35      w2v_model = Word2Vec.load(base_dir + source_dir + w2v_file_name)
36      embedding_dim = w2v_model.vector_size
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 3) 학습된 알고리즘에 대한 검증
  - 전처리 데이터 저장 및 저장된 모델 파일 위치 저장

```
38      # 알고리즘 별 데이터 셋팅 및 모델의 마지막 저장된 checkpoint 파일 이름 검색
39      if params[1] == 'CNN' :
40          x_eval = data_setting(w2v_model, embedding_dim, cnn_max_sentence_length, evaluation_text)
41          checkpoint_file = tf.train.latest_checkpoint(base_dir + cnn_model_dir)
42
43      elif params[1] == 'LSTM' :
44          x_eval = data_setting(w2v_model, embedding_dim, lstm_max_sentence_length, evaluation_text)
45          checkpoint_file = tf.train.latest_checkpoint(base_dir + lstm_model_dir)
```

  - 학습된 모델의 Tensorflow graph 저장 및 모델 로드
    - graph에서는 변수, 오퍼레이션, 컬렉션이 저장되어 있음

```
with sess.as_default():
    # 저장된 그래프를 재생성하여 모델을 불러옴
    # Tensorflow graph를 저장 하게 된다. 즉 all variables, operations, collections 등을 저장 한다. .meta로 확장자를 가진다.
    saver = tf.train.import_meta_graph("{}.meta".format(checkpoint_file))
    saver.restore(sess, checkpoint_file)
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 3) 학습된 알고리즘에 대한 검증
  - 그래프 내 오퍼레이션 확인

```
# 그래프 내 operation 리스트 확인
#for op in graph.get_operations():
#    print(op.name)
```

```
input_x
input_y
dropout_keep_prob
ExpandDims/dim
ExpandDims
conv-maxpool-2/truncated_normal/shape
conv-maxpool-2/truncated_normal/mean
conv-maxpool-2/truncated_normal/stddev
… 이하 생략
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 3) 학습된 알고리즘에 대한 검증
  - 그래프 내에서의 오퍼레이션 불러오기
    - 모델에 넣을 변수 - input_x, dropout_keep_prob, predictions, result
    - feed_dict를 모델에 사용될 실제 값 입력

```
# 그래프에서의 Operation 불러오기
input_x = graph.get_operation_by_name("input_x").outputs[0]
dropout_keep_prob = graph.get_operation_by_name("dropout_keep_prob").outputs[0]
predictions = graph.get_operation_by_name("output/predictions").outputs[0]
result = graph.get_operation_by_name("output/result").outputs[0]

if params[1] == 'CNN' :
    feed_dict = {input_x: x_eval, dropout_keep_prob: 1.0}
elif params[1] == 'LSTM' :
    batch_size = graph.get_operation_by_name("batch_size").outputs[0]
    feed_dict = {input_x: x_eval, batch_size: 1, dropout_keep_prob: 1.0}
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 3) 학습된 알고리즘에 대한 검증
  - 모델을 통한 입력 값 검증, 결과 값 출력
    - 결과 예측 값이 1인 경우 긍정, 0인 경우 부정
    - Softmax 를 통해 나온 값으로 확률 값 측정
    - 웹어플리케이션에 출력될 문장에 대해 print 문 작성

```
eval_pred, eval_result = sess.run([predictions, result], feed_dict)

# 예측된 결과에 대해 긍정/부정으로 나누고 Softmax를 통해 나온 값을 통해 확률 계산
result_pred = '긍정' if(eval_pred == 1) else '부정'
result_score = eval_result[0][1] if(eval_pred == 1) else eval_result[0][0]

print('입력된 [' + evaluation_text + ']는 ')
print('[' + str('{:.2f}'.format(result_score * 100)) + ']%의 확률로 [' + result_pred + ']으로 예측됩니다.')
```

학습된 알고리즘 모델에 대한 검증 구현

- Step 3) 학습된 알고리즘에 대한 검증
  - 모델을 통한 입력 값 검증, 결과 값 출력
    - 결과 예측 값이 1인 경우 긍정, 0인 경우 부정
    - Softmax 를 통해 나온 값으로 확률 값 측정
    - 웹어플리케이션에 출력될 문장에 대해 print 문 작성

```
eval_pred, eval_result = sess.run([predictions, result], feed_dict)

# 예측된 결과에 대해 긍정/부정으로 나누고 Softmax를 통해 나온 값을 통해 확률 계산
result_pred = '긍정' if(eval_pred == 1) else '부정'
result_score = eval_result[0][1] if(eval_pred == 1) else eval_result[0][0]

print('입력된 [' + evaluation_text + ']는 ')
print('[' + str('{:.2f}'.format(result_score * 100)) + ']%의 확률로 [' + result_pred + ']으로 예측됩니다.')
```

[합성곱 신경망 예측 결과]
입력된 [다시 찾아 보고 싶은 영화입니다.]는
[94.78]%의 확률로 [긍정]으로 예측됩니다.

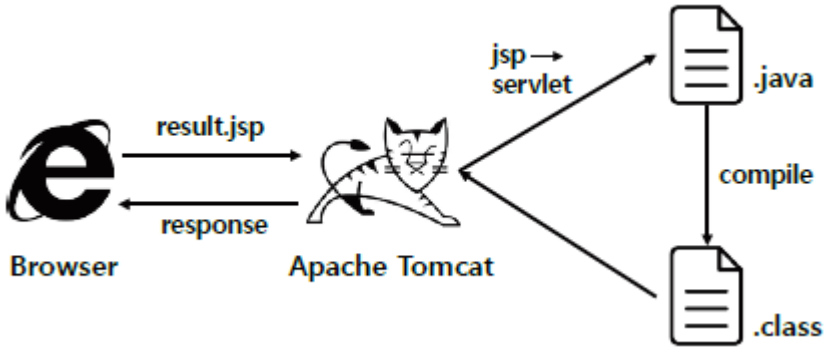
[장단기 기억 네트워크 예측 결과]
입력된 [다시 찾아 보고 싶은 영화입니다.]는
[96.44]%의 확률로 [긍정]으로 예측됩니다.

웹 어플리케이션 구현 및 실행

- 웹 어플리케이션에 대해 HTML과 JSP 를 통해 구현
  - HTML(HyperText Markup Language)
    - 제목, 단락, 목록 등과 같은 본문을 위한 구조적 의미를 나타냄
    - 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공
    - 꺾쇠 괄호에 둘러싸인 “태그”로 되어 있는 HTML 요소로 작성

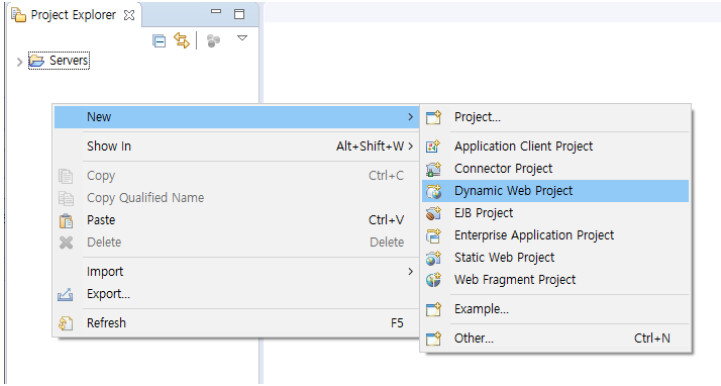
웹 어플리케이션 구현 및 실행

- 웹 어플리케이션에 대해 HTML과 JSP 를 통해 구현
  - JSP(Java Server Pages)
    - HTML 내에 자바 코드를 삽입하여 동적으로 웹 페이지를 생성한 후 웹 브라우저에 돌려주는 언어
    - Java EE(Java Platform Enterprise Edition)의 중 일부로 Apache Tomcat, Jetty, JEUS와 같은 웹 애플리케이션 서버에서 동작
    - 실행 시에는 자바 서블릿(Servlet)으로 변환된 후 실행

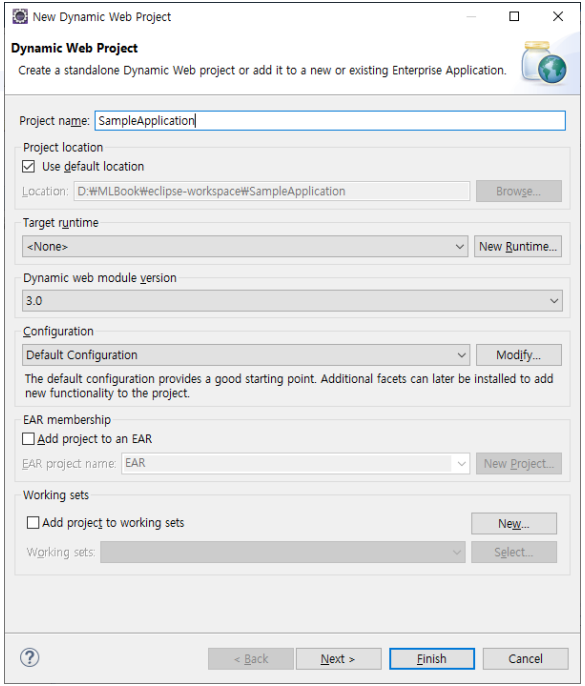


웹 어플리케이션 구현 및 실행

- 이클립스를 통한 개발 환경 셋팅



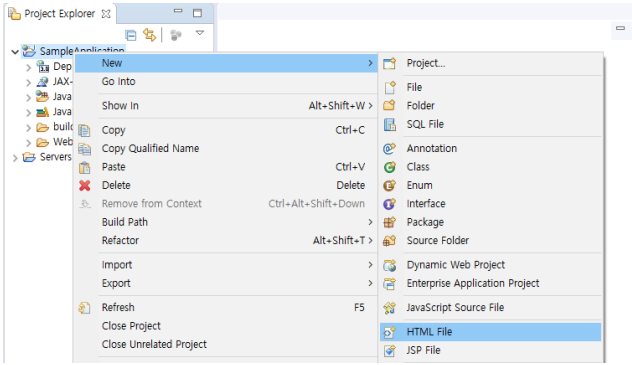
- 이클립스 실행
- [Project Explorer]에서 마우스 우클릭 -> [New-Dynamic Web Project]를 클릭하여 프로젝트를 생성



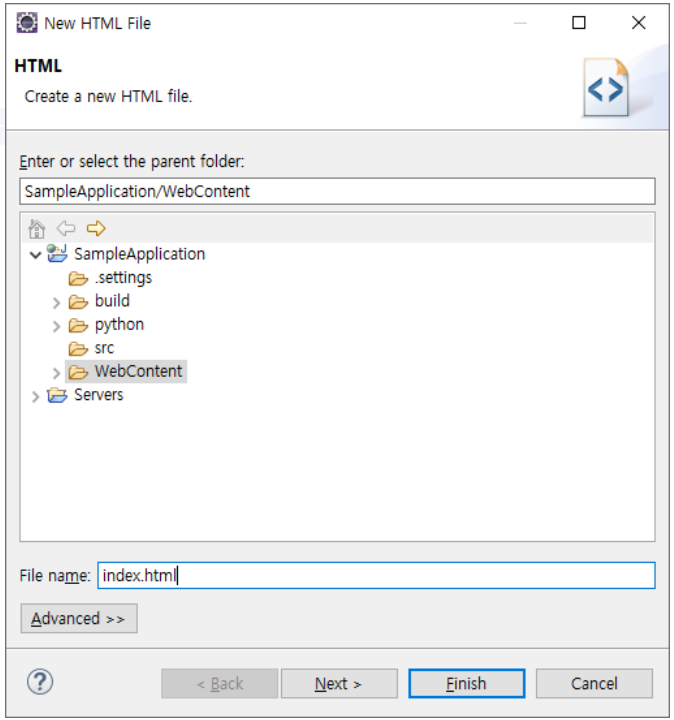
- New Dynamic Web Project 창에서 Project Name을 “SampleApplication”으로 입력
- [FINISH] 버튼을 클릭하면 [Project Explorer]에 SampleApplication 폴더가 생성

웹 어플리케이션 구현 및 실행

- 이클립스를 통한 개발 환경 셋팅



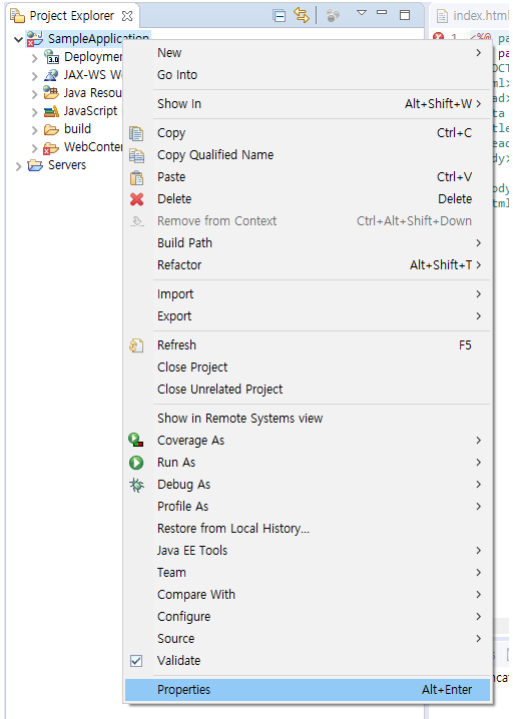
- 프로젝트 이름인 SampleApplication 에서 마우스 오른쪽 버튼을 클릭하여 [New-HTML File] 메뉴를 선택



- New HTML File 창의 file_name에 “index.html”을 입력 [FINISH] 버튼을 클릭 WebContent 폴더 내에 파일 생성
- 생성 과정을 반복하여 [New-JSP File]을 선택하고 file_name을 “result.jsp”로 입력하여 파일을 생성

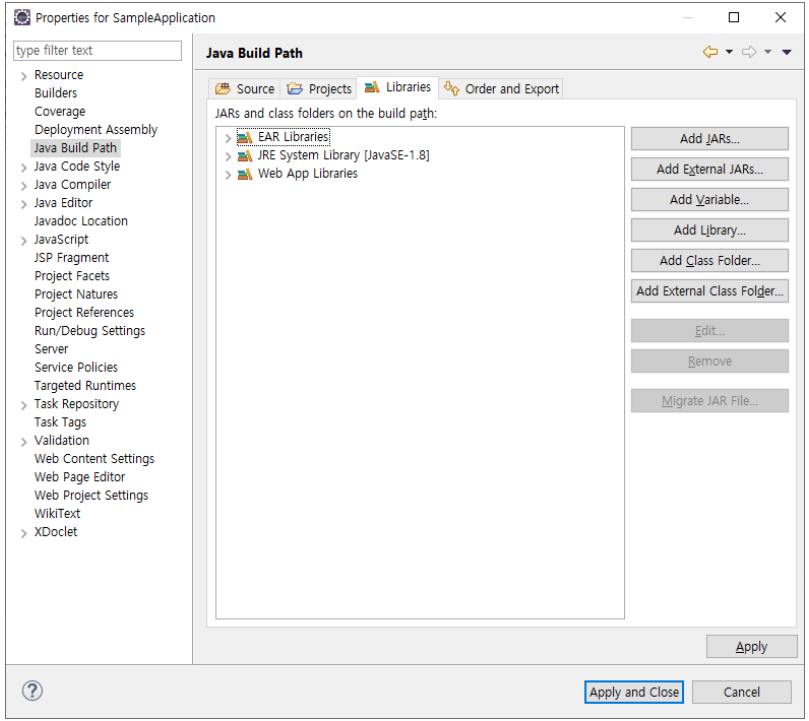
웹 어플리케이션 구현 및 실행

- 이클립스를 통한 개발 환경 셋팅



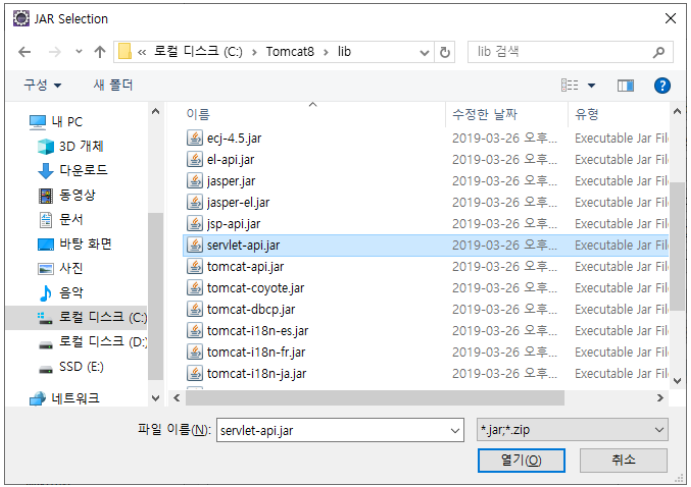
- servlet-api.jar 파일을 classpath로 설정
- SampleApplication에서 마우스 우클릭 [Properties]을 선택

- Properties for SampleApplication 창에서 왼쪽 창의 [Java Build Path]를 클릭
- 오른쪽 창의 [Library] 탭을 클릭하여 [Add External JARs] 버튼을 클릭

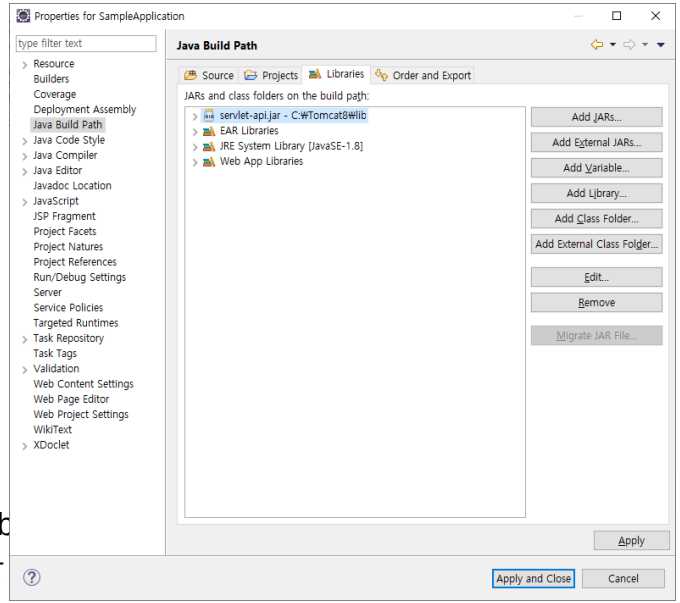


웹 어플리케이션 구현 및 실행

- 이클립스를 통한 개발 환경 셋팅



- Apache Tomcat이 설치된 폴더(C:\Tomcat8\lib로 이동 servlet-api.jar를 선택하고 [열기] 버튼을 클릭



- Properties for SampleApplication 창의 오른쪽 [Library] 탭에 “servlet-api.jar-C:\Tomcat8\lib” 확인
- [Apply and Close] 버튼을 클릭하여 환경 구성 완료

웹 어플리케이션 구현 및 실행

- HTML 코드 작성
  - 합성곱 신경망 또는 장단기 기억 네트워크 알고리즘을 선택할 수 있는 selectbox
  - 테스트를 위한 문장을 작성할 textbox
  - POST 요청을 보내기 위한 submit 버튼으로 구성
    - submit 버튼을 클릭 또는 키보드의 엔터를 클릭
    - <form> 요소의 action 속성에 따라 웹 페이지인 result.jsp 파일로 페이지를 이동, 파라미터 전송.

```
<body>
        [영화 리뷰 감정 분석 테스트]
        <br/>
        <form method="post" action="result.jsp">
                알고리즘 선택 :
                <select name="algorithm">
                        <option value="CNN">CNN</option>
                        <option value="LSTM">LSTM</option>
                </select><br/>
                테스트 문장 입력 :
                <input type="text" name="text" size=40 value=""  />
                <input type="submit" value="클릭" /><br />
        </form>
</body>
```

웹 어플리케이션 구현 및 실행

- JSP 코드 작성
  - index.html에서 전달된 파라미터를 통해 python 모델 검증파일을 실행
  - 결과를 웹 페이지에 보여 주는 코드를 구현
  - Step 1) index.html에서 전달된 파라미터 확인
    - request 처리된 파라미터에 대한 인코딩 설정
      - 한글 지원을 위한 UTF-8 설정
    - 사용자 입력 값, 알고리즘 확인 및 출력

```
<%
/* request 처리된 파라미터들의 인코딩 설정 */
request.setCharacterEncoding("UTF-8");

/* request 파라미터로 넘어온 값 확인 */
String text = request.getParameter("text");
String algorithm = request.getParameter("algorithm");
System.out.println("입력 문장 : " + text);
System.out.println("선택된 알고리즘 : " + algorithm);
```

웹 어플리케이션 구현 및 실행

- JSP 코드 작성
  - Step 2) python 파일 실행을 위한 변수 설정
    - 화면에 뿌려주기 위한 output 변수 설정
    - python 실행을 위한 실행 파일 home 위치 설정
    - python이 저장된 위치 설정
      - python으로 구현된 py 파일
      - data 폴더 내 Word2Vec 파일, cnn_model과 lstm_model 폴더
      - 이클립스 프로젝트 내의 python 폴더 안으로 파일들을 복사

```
String output = "";

/* Python 설치 바이너리 폴더 설정 */
String pythonHome = "D:/Python36/";
/* 알고리즘 학습 모델 및 python 파일 저장 폴더 설정 */
String rscPath = "D:/MLBook/eclipse-workspace/SampleApplication/python/";
/* 실행할 검증 구현 python 파일 위치 설정 */
String pyFile = rscPath + "nsmc_evaluation.py";
```

웹 어플리케이션 구현 및 실행

- JSP 코드 작성
  - Step 3) python 파일 실행을 위한 변수 설정
    - 외부 프로세스를 실행을 위한 명령 설정
      - ex) python nsmc_evaluation.py 'CNN' '테스트입니다.'
    - Rutime 클래스를 통해 명령 실행
    - 프로세스의 입력, 에러 스트림 저장

```
if (algorithm != null && text != null) {
        /* 외브 프로세스 실행을 위한 command 설정 */
        String[] command = { pythonHome + "/python", pyFile, algorithm, text };
        String line = null;

        /* 외부 프로세스 실행 */
        Process p = Runtime.getRuntime().exec(command);
        p.waitFor();

        /* 자식 프로세스의 입력 및 에러 스트림 저장 */
        BufferedReader br = new BufferedReader(
                        new InputStreamReader(p.getInputStream()));
        BufferedReader stdError = new BufferedReader(
                        new InputStreamReader(p.getErrorStream()));
```

웹 어플리케이션 구현 및 실행

- JSP 코드 작성
  - Step 3) 외부 프로세스 실행을 통한 python 실행
    - 외부 프로세스를 실행을 위한 명령 설정
      - ex) python nsmc_evaluation.py 'CNN' '테스트입니다.'
    - Rutime 클래스를 통해 명령 실행
      - Runtime.getRuntime().exec()
    - 프로세스의 입력, 에러 스트림 저장

```
if (algorithm != null && text != null) {
        /* 외브 프로세스 실행을 위한 command 설정 */
        String[] command = { pythonHome + "/python", pyFile, algorithm, text };
        String line = null;

        /* 외부 프로세스 실행 */
        Process p = Runtime.getRuntime().exec(command);
        p.waitFor();

        /* 자식 프로세스의 입력 및 에러 스트림 저장 */
        BufferedReader br = new BufferedReader(
                        new InputStreamReader(p.getInputStream()));
        BufferedReader stdError = new BufferedReader(
                        new InputStreamReader(p.getErrorStream()));
```

웹 어플리케이션 구현 및 실행

- JSP 코드 작성
  - Step 3) 외부 프로세스 실행을 통한 python 실행
    - 입력 스트림을 읽어 실행된 결과 저장
    - 에러 스트림을 읽어 에러 메시지 출력
    - 외부 프로세스 실행 종료
    - 파라미터 값이 정상적으로 넘어오지 않았을 경우 에러 메시지 저장

```
/* 파라미터 값이 없는 경우 에러 메시지 저장 */
output = "알고리즘 또는 테스트 문장이 입력되지 않았습니다.";
```

```
/* 입력 스트림을 읽어 내용 확인 및 output변수에 저장 */
while ((line = br.readLine()) != null) {
        System.out.println(line);
        output += line + "\n";
}
br.close();

/* 에러 스트림을 읽어 내용 확인 및 에러 메시지 저장 */
while ((line = stdError.readLine()) != null) {
        System.out.println(line);
        output = "실행 시 에러가 발생하였습니다.";
}
stdError.close();
p.destroy();
```

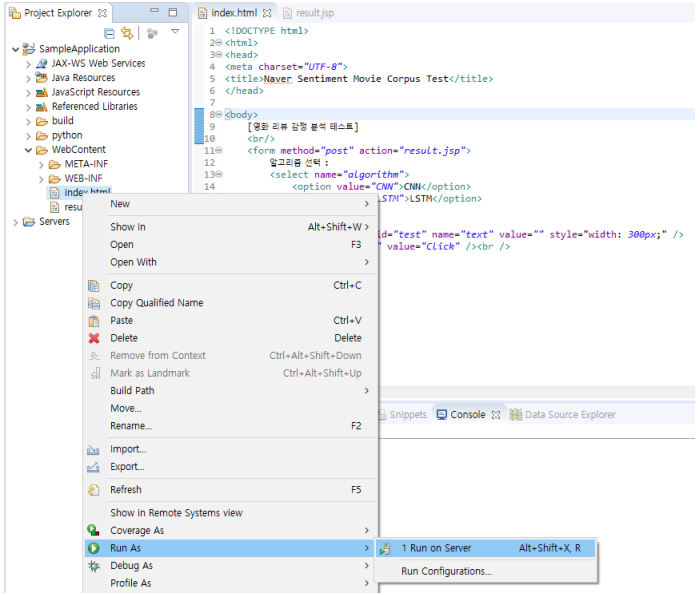
웹 어플리케이션 구현 및 실행

- JSP 코드 작성
  - Step 4) 결과에 대해 웹 페이지에 결과 출력
    - 개행으로 나눠 String 배열로 저장
    - 알고리즘 값과 결과 값에 대해서 화면에 출력
    - 뒤로가기 버튼 클릭 시 index.html 파일로 이동

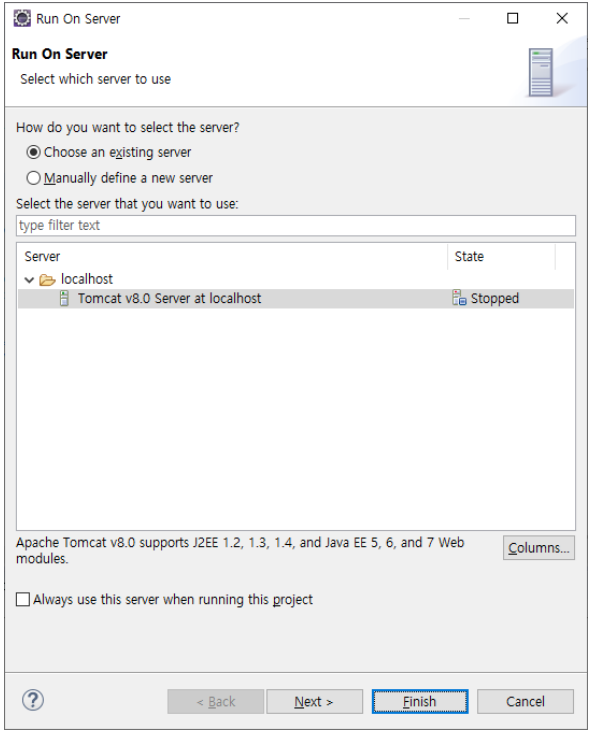
```
/* 개행으로 나눠 String 배열에 저장 */
String[] result = output.split("\\n");
%>
<!-- 선택된 알고리즘 및 결과 값 화면 출력 -->
선택된 알고리즘 : <%=algorithm%><br/>
<%
        for (String str : result) {
%>
<%=str%><br/> <% } %>
<!-- index.html 로 이동 -->
<input type="button" value="뒤로 가기" onclick="location.href='index.html'" />
```

웹 어플리케이션 구현 및 실행

- 웹 어플리케이션 실행



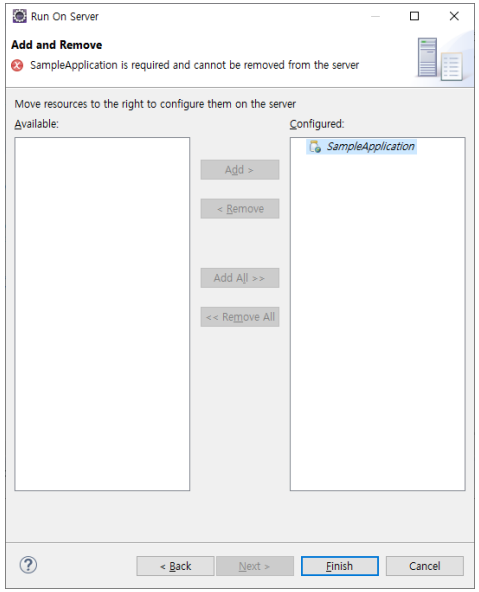
- [Project Explorer-SampleApplication-WebContent-index.html] 에서 우클릭하여 [Run As-Run on Server]를 클릭



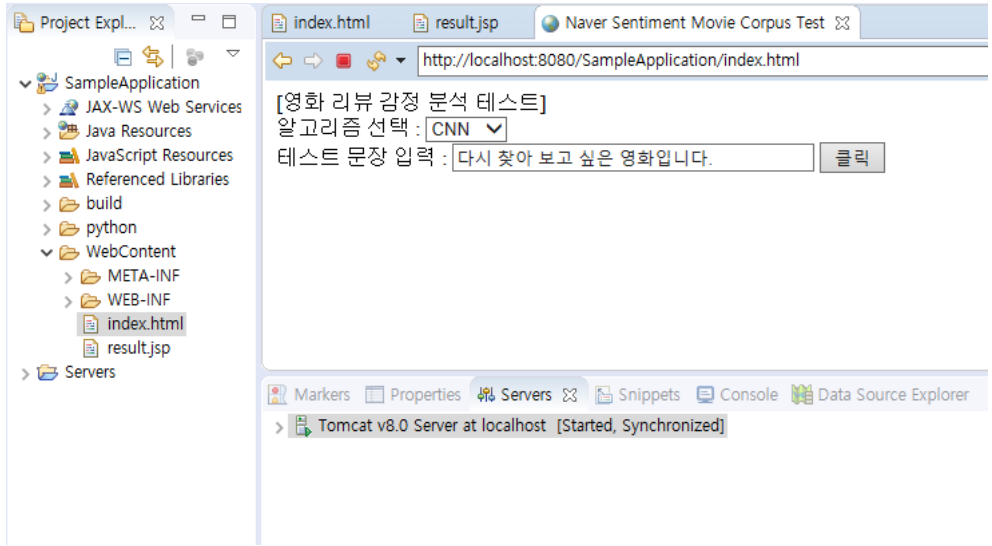
- Run On Server 창에서 [Tomcat v8.0 Server at localhost]를 클릭
- [Next] 버튼을 클릭
- Apache Tomcat Server로 실행할 Application을 설정

웹 어플리케이션 구현 및 실행

- 웹 어플리케이션 실행



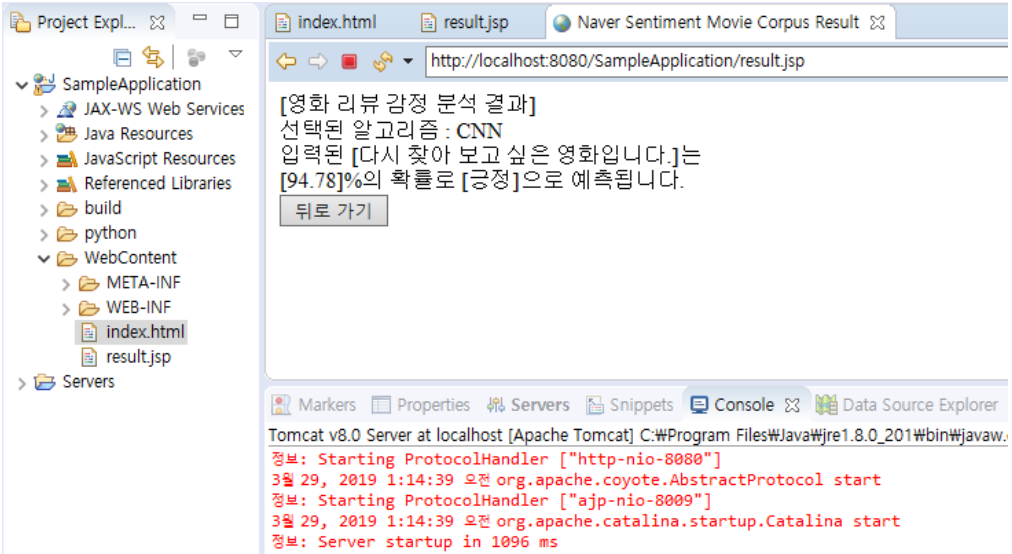
- Available에 존재하는 SampleApplication 프로젝트를 [Add >] 버튼을 클릭하여 Configured로 이동
- [Finish] 버튼을 클릭 Stopped 상태로 되어 있는 Tomcat 서버를 기동 시킴과 동시에 웹 애플리케이션이 실행



- Servers 탭에서 Tomcat v8.0 Server at localhost [Started, Synchronized] 상태로 기동이 완료
- 알고리즘 선택 및 테스트 문장 입력 후 클릭 버튼 클릭 또는 엔터 실행

웹 어플리케이션 구현 및 실행

- 웹 어플리케이션 결과 확인



- python 프로세스를 실행하여 출력된 값을 웹 페이지를 통해 결과 출력
- 외부 프로세스 실행으로 인해 실행 시간 소요