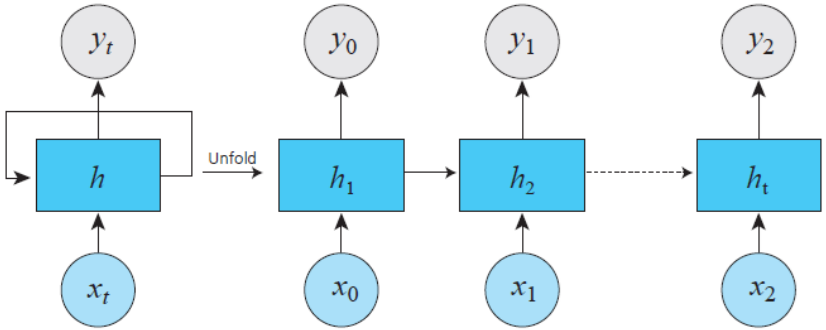
장단기 기억 네트워크

Long-Short Term Memory Network

장단기 기억 네트워크(LSTM) 구현

- 순환신경망(Recurrent Neural Network, RNN) 이란?
  - 문서 감정 분류, 필기체 인식, 음성 인식과 같은 자연어 처리
  - 주가 등 시간을 중심으로 앞, 뒤의 내용이 연관 관계가 있는 시계열 데이터를 처리에 좋은 성능
  - 시간 스탭 t에서의 입력값 xt, 출력값 yt와 h인 은닉층이 존재
    - 은닉층의 출력이 다음 시간 스탭에서의 은닉층으로 입력되는 구조가 반복되는 형태
    - 하나의 네트워크 구조가 여러 개가 연결되어 다음 단계로의 정보를 전달
    - 메모리 셀(memory cell) : 이전 정보를 은닉층에서 일시적으로 메모리(memory) 형태로 기억
    - 은닉 상태(hidden state) : 메모리 셀의 상태
    - 은닉 상태값은 현재 입력값과 이전의 은닉 상태의 값을 가중치를 곱하고 편향을 더함
    - 활성화 함수로 하이퍼볼릭 탄젠트(tanh) 함수



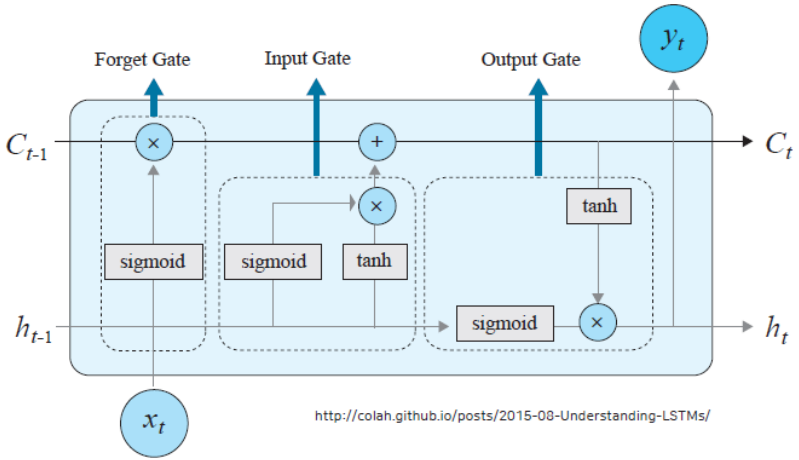
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

장단기 기억 네트워크(LSTM) 구현

- 순환신경망(Recurrent Neural Network, RNN) 이란?
  - 순환 신경망의 학습에는 경사 하강법을 이용하며 출력에서의 경사가 현재 시간에만 의존하는 게 아니라 이전 시간 스탭에도 의존
  - 시간 기반 역전파(BackPropagation Through Time, BPTT)라는 변형된 알고리즘으로 가중치를 업데이트
  - 경사도 사라짐 문제(Gradient Vanishing Problem)
    - 시간을 많이 거슬러 올라가게 되면 신경망이 곱하기 연산으로 되어 있기 때문에 역전파에서의 경사가 점점 줄어들어 학습 능력이 저하하는 단점
  - 문장에서의 단어를 예측할 때 오래전 데이터를 고려하여 예측을 해야 되는 경우에는 순환 신경망에서의 성능이 감소하는 결과
- 장단기 기억 네트워크(Long-Short Term Memory Network, LSTM) 이란?
  - 1997년Hocheiter & Schmidhuber 이 제안한 알고리즘
  - 순환 신경망에서의 장기 의존성(Long-Term Dependencies) 문제를 해결
  - 순차적으로 입력되는 데이터의 시간 흐름이 길더라도 잊어야 할 정보들은 잊고 유지해야 될 정보들은 유지하면서 성능을 최적화

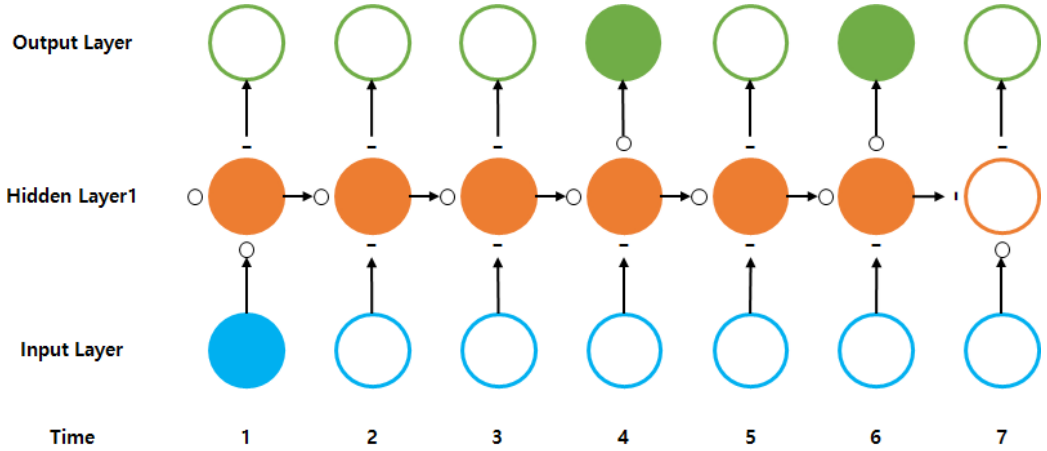
장단기 기억 네트워크(LSTM) 구현

- 장단기 기억 네트워크구조
  - 순환 신경망에서 존재하지 않던 ct인 셀 스테이트(Cell State)가 추가
  - 망각, 입력, 출력의 정도를 조절하는 3개의 게이트(Gate)가 추가
    - 셀 스테이트 : 각 게이트의 정보들이 다음 단계로 진행될 수 있도록 역할
    - 망각 게이트 : 셀 스테이트에서 버릴 정보를 정하는 단계
      - 입력값과 이전 은닉층에서 입력된 값과 함께 시그모이드 출력값 생성
      - 시그모이드 출력값이 1인 경우 과거의 값을 그대로 유지하고, 0인 경우에는 완전히 값을 버림
    - 입력 게이트 : 새로운 정보에 대해 셀 스테이트에 저장할지를 결정하는 단계
      - 시그모이드를 통해 업데이트할 정보 결정
      - tanh 레이어를 통해 셀 스테이트에 더할 새로운 후보 값을 만들고 두 값을 합쳐 새로운 셀 스테이트로 정보를 업데이트
    - 출력 게이트
      - 어떤 값을 출력할지 시그모이드 레이어를 통해 결정
      - 셀 스테이트를 tanh 레이어를 통한 결괏값을 곱하여 원하는 결괏값만 반영



장단기 기억 네트워크(LSTM) 구현

- 장단기 기억 네트워크구조
  - 시간에 따라 각 게이트 동작
    - 망각, 입력, 출력 게이트를 열고 닫으면서 오랜 시간이 지나더라도 기억을 오랫동안 보존
    - 직선은 닫힌 게이트, 동그라미는 열린 게이트
    - 은닉층의 위, 왼쪽, 아래는 게이트가 출력, 망각, 입력 게이트를 표현
    - 입력층에서는 2~6번째의 시간에서 입력 게이트를 닫음
    - 출력에서는 4, 6번째 시간에서만 출력 게이트를 열어 경사도 사라짐을 방지



https://skymind.ai/wiki/lstm/

장단기 기억 네트워크(LSTM) 구현

- 전처리 단계 - Step 1) 파라미터 설정
  - 파라미터 설정
    - 단어의 벡터 차원 수 -> embedding_dim : 100
    - 레이블 차원 수 -> class_sizes : 2
    - 문장 내 최대 단어 개수 -> max_sentence_length : 100
    - LSTM 셀 결과 크기 -> hidden_size : 30
    - 학습 시 드롭아웃 변수 -> dropout_keep_prob : 0.5
    - 학습 횟수 -> num_epochs : 20
    - 패치 사이즈 -> batch_size : 1000
    - 검증을 위한 조건 -> evaluate_every : 150
    - 최적화 알고리즘 학습률 -> learn_rate : 0.001
    - LSTM 셀 레이어 개수 -> num_layers : 2

```
73  def run_lstm(params) :
74      class_sizes = 2
75      max_sentence_length = int(params[1])
76      hidden_size = int(params[2])
77      dropout_keep_prob = float(params[3])
78      num_epochs = int(params[4])
79      batch_size = int(params[5])
80      evaluate_every = int(params[6])
81      learn_rate = float(params[7])
82      num_layers = int(params[8])
```

장단기 기억 네트워크(LSTM) 구현

- 장단기 기억 네트워크 모델 - Step 1) 모델 생성을 위한 변수 초기화
  - batch_size : dynamic_rnn에서의 초기 상태를 0으로 초기화하는 데 사용

```
5           # 학습 데이터가 들어갈 플레이스 홀더 선언
6           self.input_x = tf.placeholder(tf.float32, shape=[None, sequence_length, embedding_size], name='input_x')
7           self.input_y = tf.placeholder(tf.float32, shape=[None, num_classes], name='input_y')
8           self.dropout_keep_prob = tf.placeholder(tf.float32, name='dropout_keep_prob')
9           self.batch_size = tf.placeholder(tf.int32, [], name="batch_size")
```

- 장단기 기억 네트워크 모델 - Step 2) LSTM 레이어
  - BasicLSTMCell 함수 보다 옵션(peephole 연결 등)이 추가된 고급 모델
    - num_units : LSTM 셀의 유닛 개수로 출력값의 크기
    - forget_bias : 망각 게이트(forget gate)의 편향(bias)으로 1인 경우 학습 시 망각의 규모를 줄임
    - state_is_tuple : 튜플의 상태를 true인 경우에는 c_state, m_state를 반환, false의 경우에는 c_state, m_state를 합쳐서 하나로 연결
  - 드롭아웃을 적용

```
11          # LSTM Layer
12          with tf.name_scope("lstm"):
13              def lstm_cell():
14                  #tf.nn.rnn_cell.(Basic)LSTMCell / tf.nn.rnn_cell.(Basic)RNNCell / tf.nn.rnn_cell.GRUCell
15                  # LSTM Cell 및 DropOut 설정
16                  lstm = tf.nn.rnn_cell.LSTMCell(num_units=hidden_unit, forget_bias=1.0, state_is_tuple=True)
17                  return tf.nn.rnn_cell.DropoutWrapper(cell=lstm, output_keep_prob=self.dropout_keep_prob)
```

장단기 기억 네트워크(LSTM) 구현

- 장단기 기억 네트워크 모델 - Step 2) LSTM 레이어
  - MultiRNNCell 함수에서 사용자가 설정한 layer 개수에 따라 LSTM 레이어 생성
  - 설정된 LSTM 셀은 dynamic_rnn 함수를 통해 모델의 결과와 마지막 상태 값이 반환
  - 출력값은 [batch_size, sequence_length, hidden_unit]의 형태
  - 최종 결과는 가장 마지막 결괏값인 [batch_size, hidden_unit]을 사용
  - transpose를 사용하여 행렬의 순서를 [sequence_length, batch_size, hidden_unit]의 형태로 변경
  - gather로 출력의 마지막 결괏값만 사용하여 [batch_size, hidden_unit] 형태의 값을 저장

```
# RNN Cell을 여러 층 쌓기
lstm_cell = tf.nn.rnn_cell.MultiRNNCell([lstm_cell() for _ in range(num_layer)])
# 초기 state 값을 0으로 초기화
self.initial_state = lstm_cell.zero_state(self.batch_size, tf.float32)
# outputs : [batch_size, sequence_length, hidden_unit]
outputs, state = tf.nn.dynamic_rnn(lstm_cell, self.input_x, initial_state=self.initial_state , dtype=tf.float32)
# output : [sequence_length, batch_sie, hidden_unit)
output = tf.transpose(outputs, [1, 0, 2])
# 마지막 출력만 사용
output = tf.gather(output, int(output.get_shape()[0]) - 1)
```

장단기 기억 네트워크(LSTM) 구현

- 장단기 기억 네트워크 모델 학습 실행
  - LSTM 모델에 사용할 변수 초기화

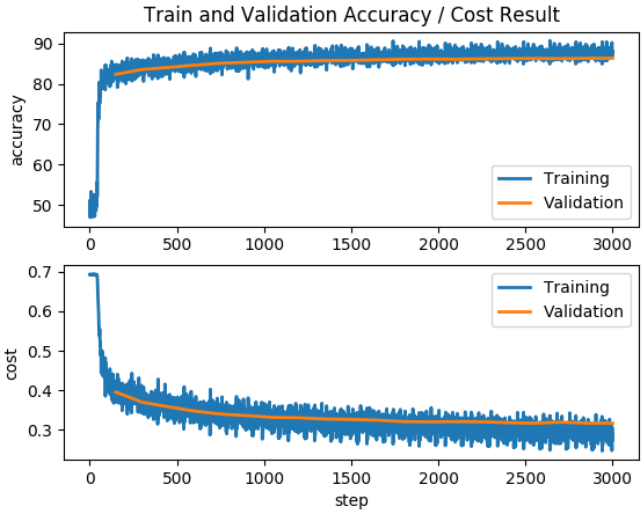
```
with tf.Graph().as_default():
    #sess_config = tf.ConfigProto(device_count = {'GPU': 0})
    sess_config = tf.ConfigProto()
    sess_config.gpu_options.allow_growth = True
    sess = tf.Session(config=sess_config)
    with sess.as_default():
        lstm = lstm_model(
            hidden_unit = hidden_size,
            sequence_length=x_train.shape[1],
            num_classes=y_train.shape[1],
            num_layer=num_layers,
            embedding_size=embedding_dim)
```

```
def train_step(x_batch, y_batch):
    feed_dict = {
        lstm.input_x: x_batch,
        lstm.input_y: y_batch,
        lstm.batch_size: len(x_batch),
        lstm.dropout_keep_prob: dropout_keep_prob
    }
def dev_step(x_batch, y_batch, epoch):
    feed_dict = {
        lstm.input_x: x_batch,
        lstm.input_y: y_batch,
        lstm.batch_size: len(x_batch),
        lstm.dropout_keep_prob: 1.0
    }
```

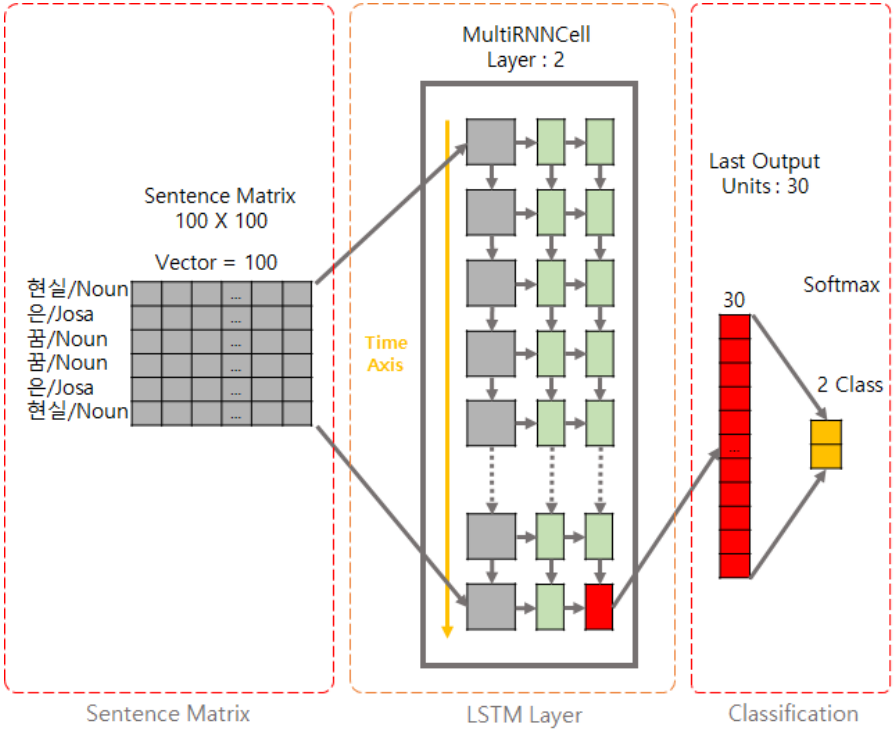
  - 학습 결과
    - 학습 결과
    - 총 20회의 학습
    - 비용 값은 0.395에서 0.316로 감소
    - 정확도는 82.3%에서 86.3%로 증가

장단기 기억 네트워크(LSTM) 구현

- 장단기 기억 네트워크 모델 학습 실행
  - 학습 결과
    - 학습 결과
    - 총 20회의 학습
    - 비용 값은 0.395에서 0.316로 감소
    - 정확도는 82.3%에서 86.3%로 증가



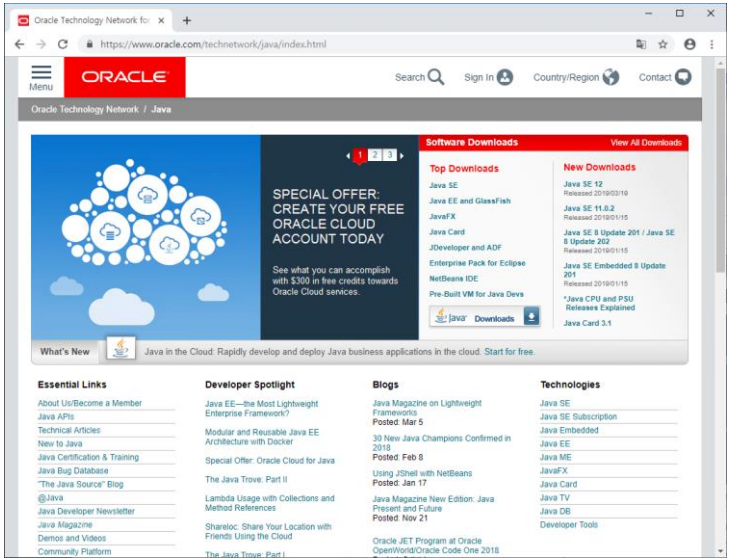
- 장단기 기억 네트워크 구현 전반적 구조



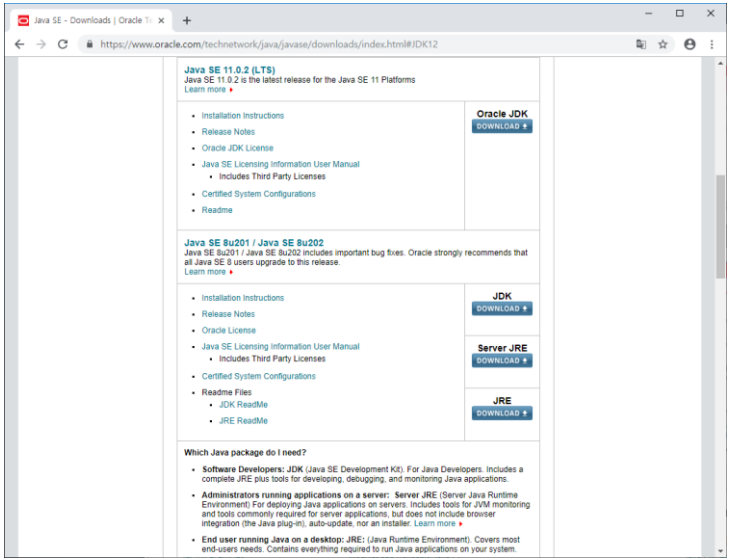
웹어플리케이션 실행환경 구축

자바(JAVA) 설치하기

- 웹 애플리케이션을 개발하기 위한 이클립스는 자바를 기반으로 동작



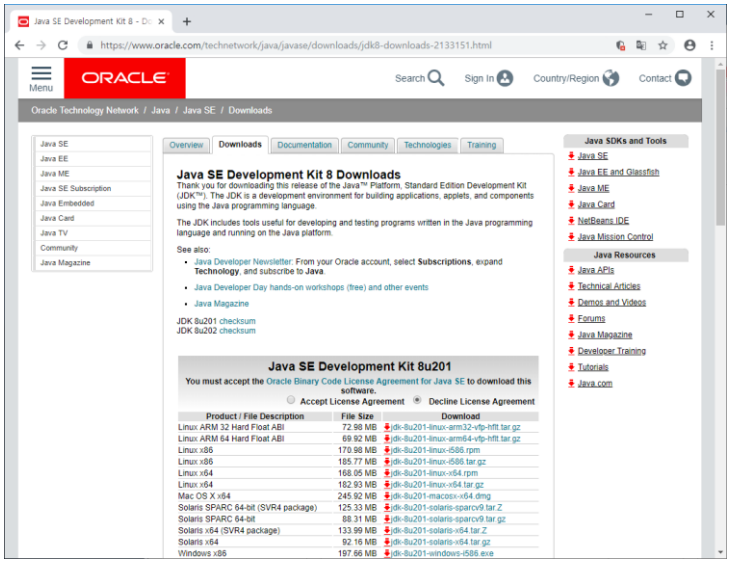
1)Oracle 홈페이지 접속
https://www.oracle.com/technetwork/java/index.html
2)[New Download - Java SE 8 Update 201 / Java SE 8 Update 202] 클릭
Java SE 8 다운로드 홈페이지 이동



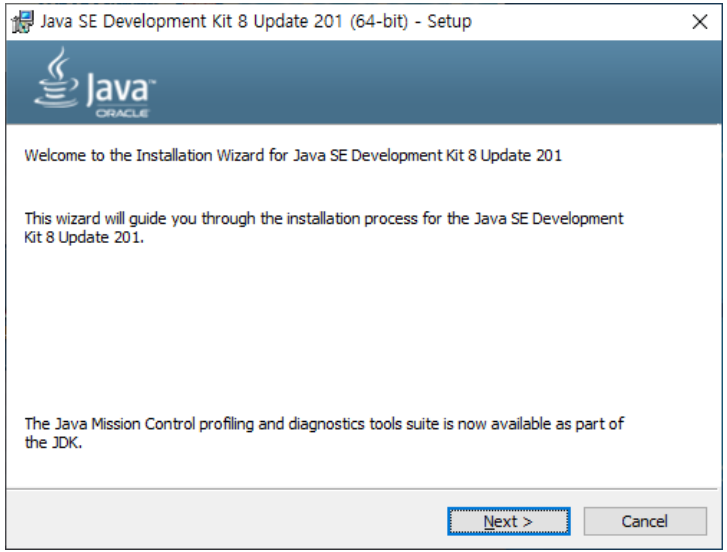
[Java SE 8u201 / Java SE 8u202-JDK Download] 버튼을 클릭하여 다운로드 화면으로 이동

자바(JAVA) 설치하기

- 웹 애플리케이션을 개발하기 위한 이클립스는 자바를 기반으로 동작



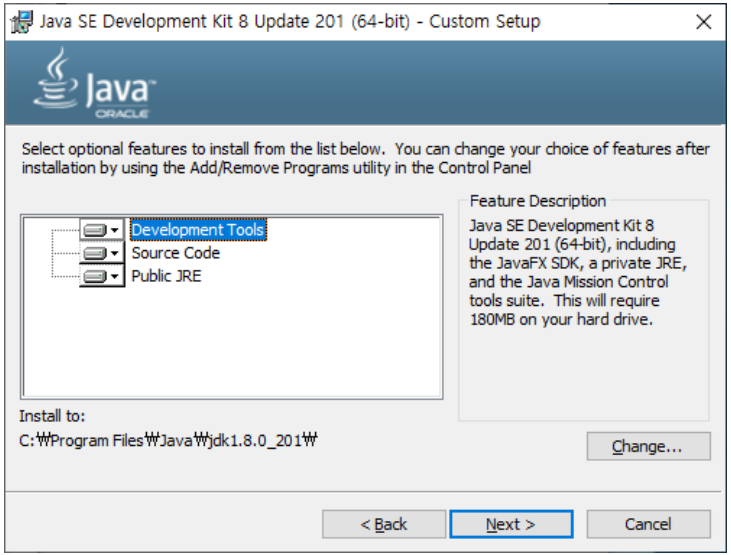
1)java SE Develop Kit 8u201 부분에서 “Accept License Agreement”를 클릭
2)컴퓨터의 운영 체제에 맞는 자바 파일을 다운로드
- jdk-8u201-windows-x64.exe 파일을 다운로드



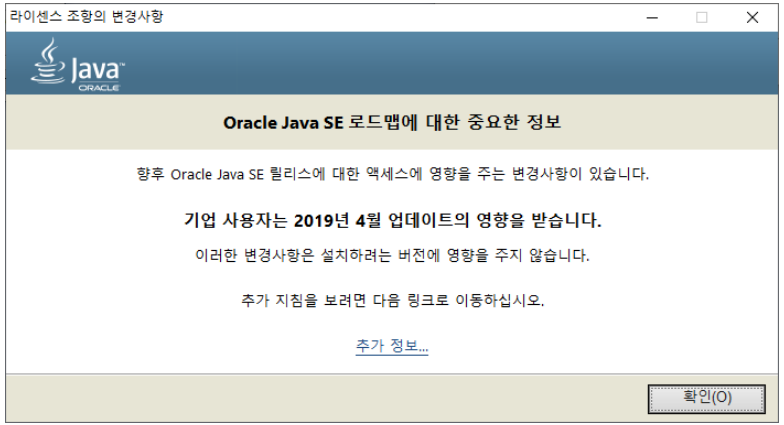
1)다운로드한 위치에서 jdk-8u201-windows-x64.exe 파일을 실행
2) Next 클릭

자바(JAVA) 설치하기

- 웹 애플리케이션을 개발하기 위한 이클립스는 자바를 기반으로 동작



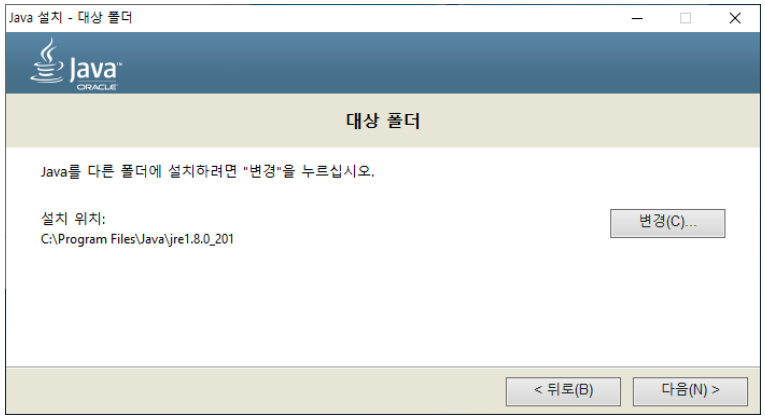
1)JDK를 설치할 폴더를 설정
기본 설치 위치는 “C:\Program Files\Java\jdk1.8.0_201\”
2)폴더를 변경하기 위해서 [Change...] 버튼을 클릭한 후 설정하여 변경



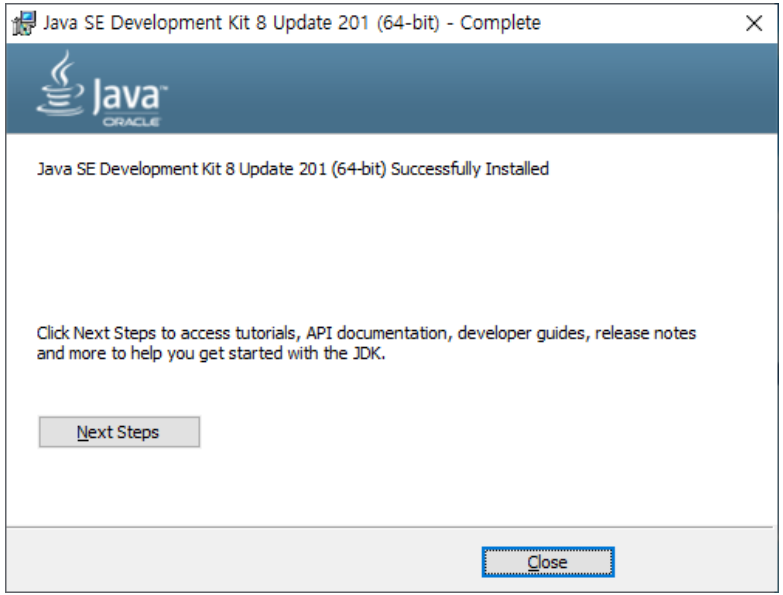
라이선스 조항의 변경 사항 화면으로 Oracle에서 자바의 구독형 라이선스 구매에 따른 변경 사항.
개인 사용자의 경우에는 사용이 무료이고 2020년 말까지 업데이트 사용이 가능.
확인 버튼을 클릭하여 다음 페이지로 이동

자바(JAVA) 설치하기

- 웹 애플리케이션을 개발하기 위한 이클립스는 자바를 기반으로 동작



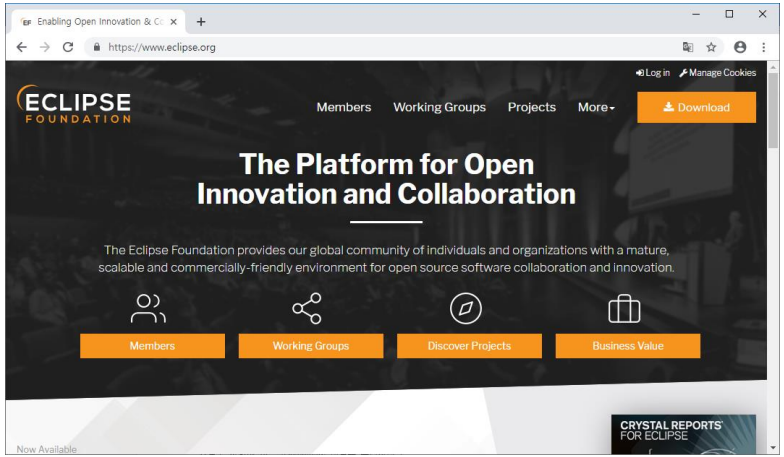
JRE(Java Runtime Environment)에 대한 설치 폴더 변경 확인 후 [다음] 버튼을 클릭하면 JRE 설치가 진행.



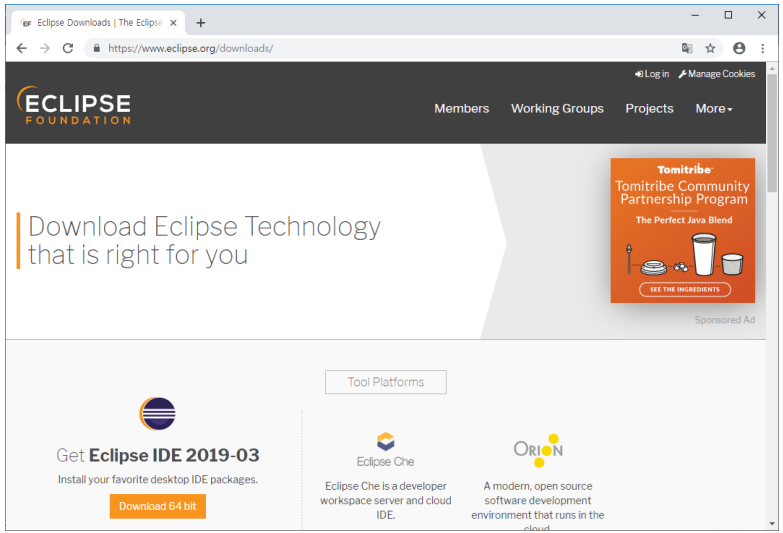
자바 설치가 완료된 화면으로 [Close] 버튼을 클릭하여 설치를 마무리.

이클립스(Eclipse) 설치하기

- HTML과 JSP를 개발하는데 필요한 통합 개발 환경(Integrated Development Environment)



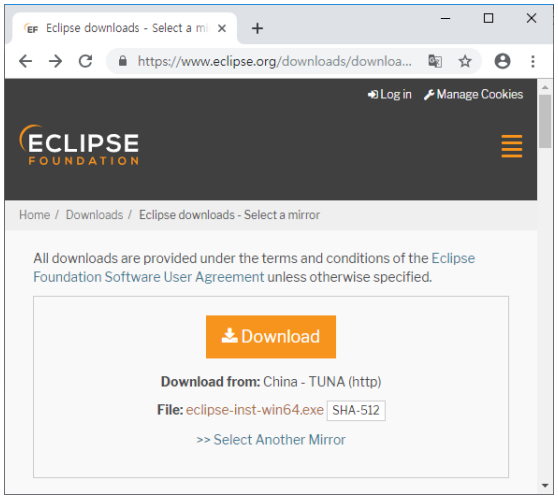
1)이클립스 사이트(https://www.eclipse.org/) 접속
2)오른쪽 상위에 있는 [Download] 버튼을 클릭

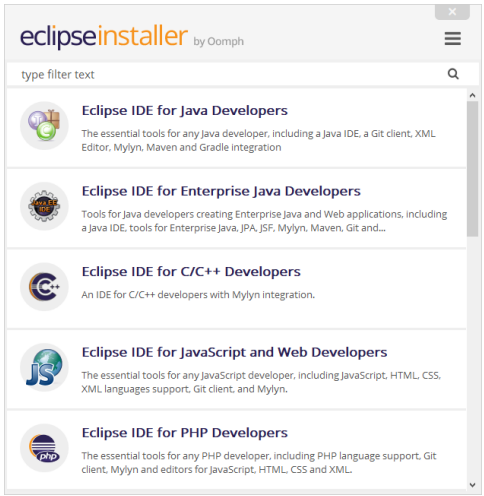


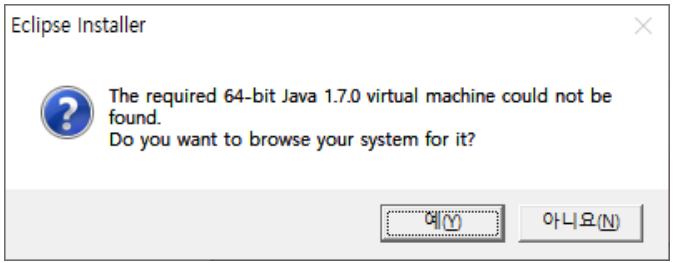
[Download 64bit] 버튼을 클릭

이클립스(Eclipse) 설치하기

- HTML과 JSP를 개발하는데 필요한 통합 개발 환경(Integrated Development Environment)







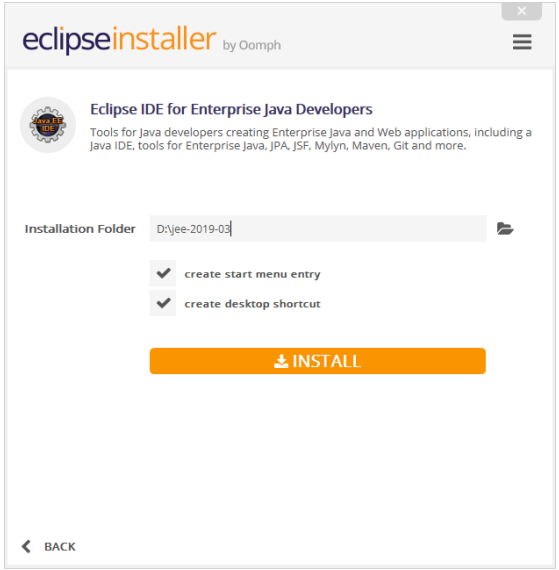
이클립스 설치를 진행하는 시점에
Java가 설치되지 않은 경우 발생함
- 해당 창 발생 시 Java 설치

다운로드받을 미러 사이트를 확인 /
소프트웨어 사용자 계약 내용 확인 후
[Download] 버튼을 클릭

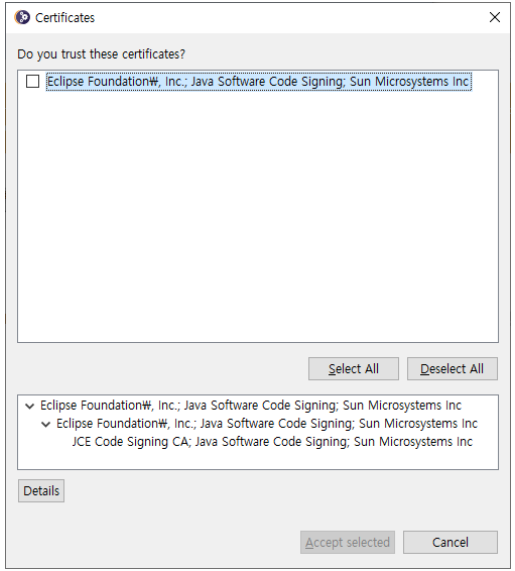
다운로드한 eclipse-inst-win64.exe 파일을 실행
JSP 웹 애플리케이션 개발을 위하여
“Eclipse IDE for Enterprise Java Developers”를 클릭

이클립스(Eclipse) 설치하기

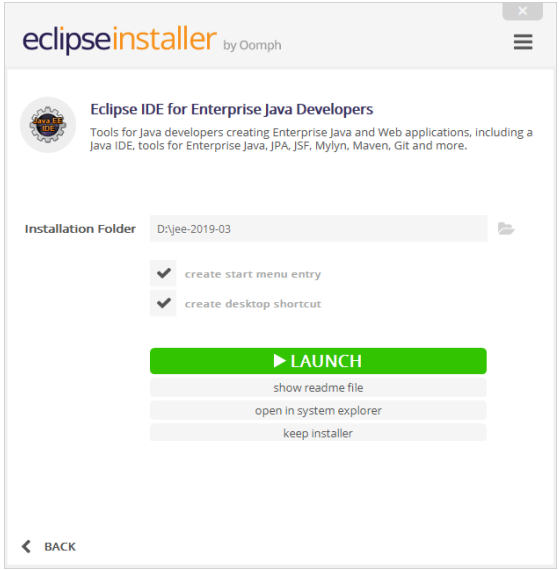
- HTML과 JSP를 개발하는데 필요한 통합 개발 환경(Integrated Development Environment)



이클립스 IDE를 설치할 폴더를 선택
“윈도우 시작 메뉴 추가”,
“바탕화면 바로 가기 생성” 등을 선택
[INSTALL] 버튼을 클릭하여 설치



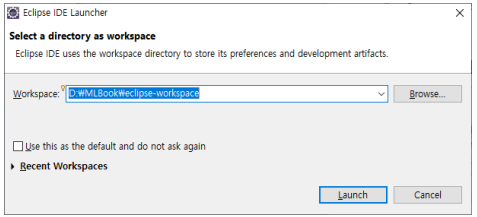
증명서 부분의 네모 부분을 클릭 후 [Accepted select] 버튼을 클릭



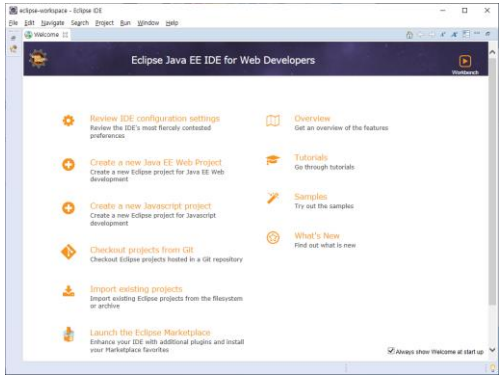
[Launcher] 버튼을 클릭하면 이클립스가 실행
- 바탕화면에는 바로 가기 아이콘과 윈도우 시작 메뉴에 “Eclipse”가 생성되었음을 확인

이클립스(Eclipse) 설치하기

- HTML과 JSP를 개발하는데 필요한 통합 개발 환경(Integrated Development Environment)



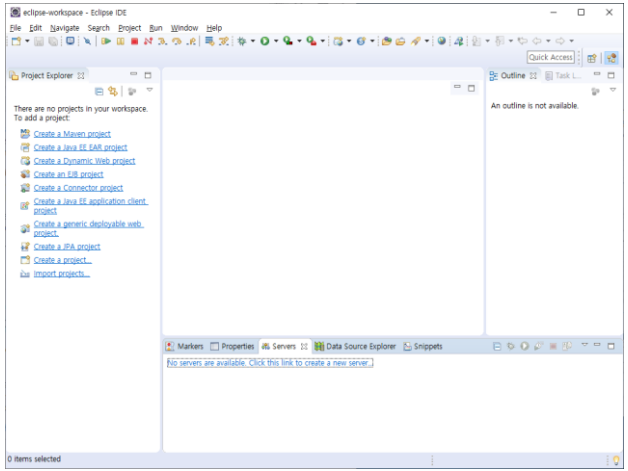
이클립스를 실행하면 작업할 공간인 워크스페이스를 선택하는 창이 나타남
워크스페이스 변경 시 [Browse]를 누르면 되고 설정 완료되면 [Launcher] 버튼을 클릭하면 초기 화면으로 이동



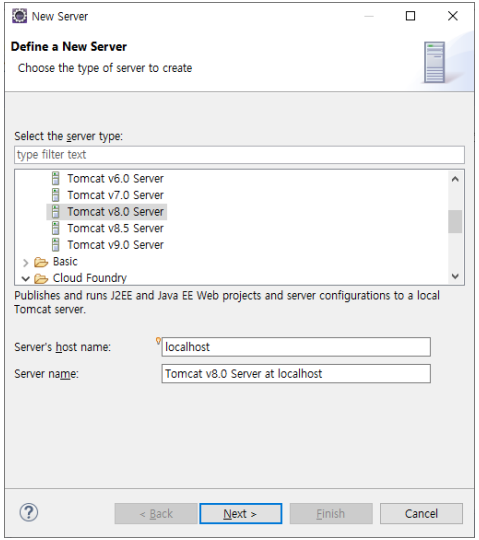
이클립스 IDE 초기 화면으로 “Welcome” 탭의 “x”를 클릭
창을 닫아 작업 공간이 나타나면 프로그래밍 할 준비가 완료

톰켓(Tomcat) 설치하기

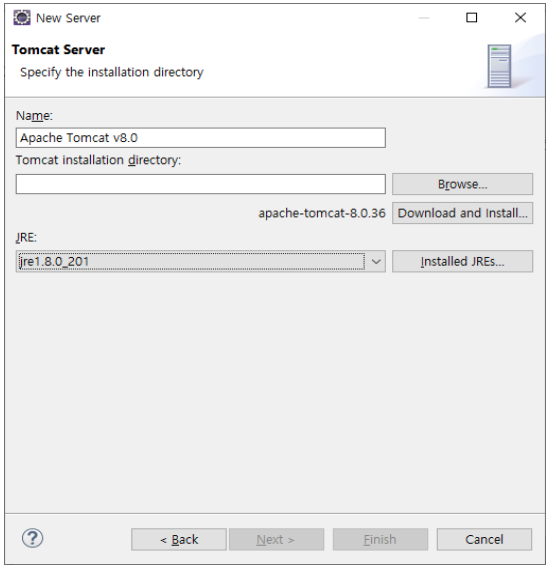
- 웹 애플리케이션 구동을 위해 아파치 재단에서 개발한 오픈 소스 소프트웨어인 Apache Tomcat에 대한 설치



초기 화면의 하위 “Server” 탭을 확인
 - 설정된 서버가 없는 것을 확인
“No servers are available, Click the link to create a new server” 링크를 클릭



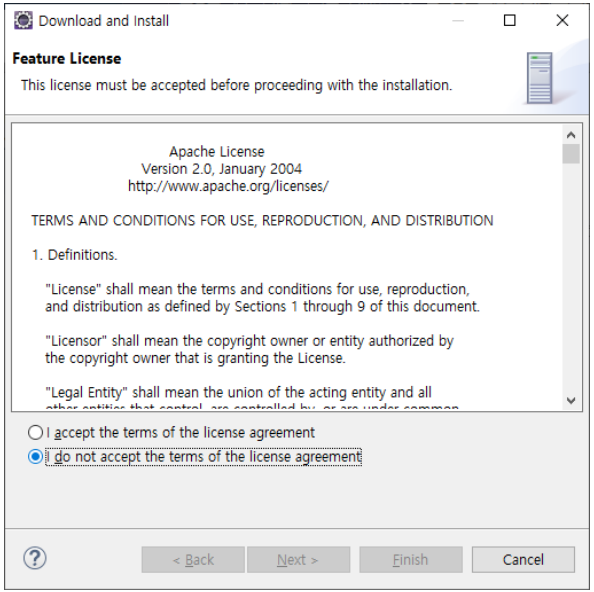
새로운 서버 정의 화면
 - “Apache 폴더 - Tomcat v8.0 Server” 를 선택하고 [Next] 버튼을 클릭
 - v8.5 이상 다운로드 불가능



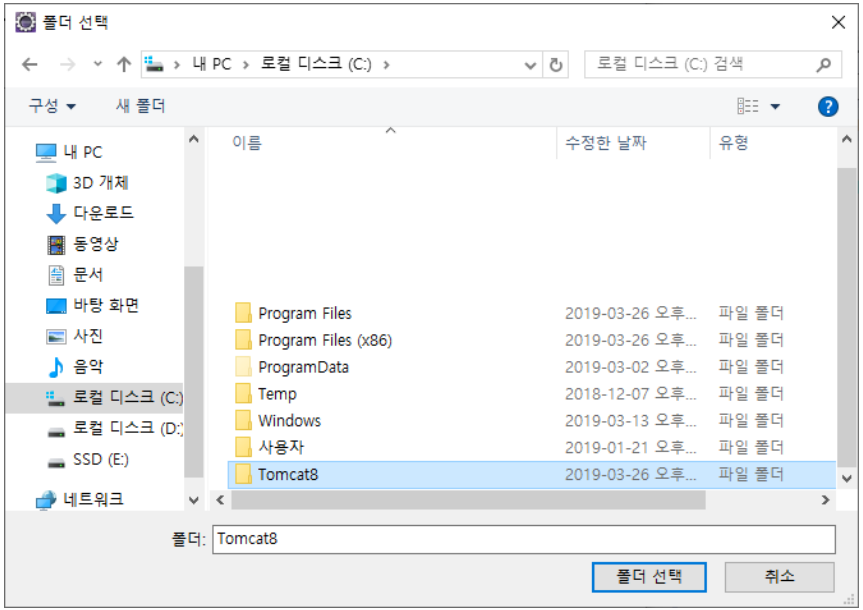
JRE 설정에는 이전에 설치한 jre1.8.0_201을 선택 후 [Download and Install...] 버튼을 클릭하여 Tomcat 설치

톰켓(Tomcat) 설치하기

- 웹 애플리케이션 구동을 위해 아파치 재단에서 개발한 오픈 소스 소프트웨어인 Apache Tomcat에 대한 설치



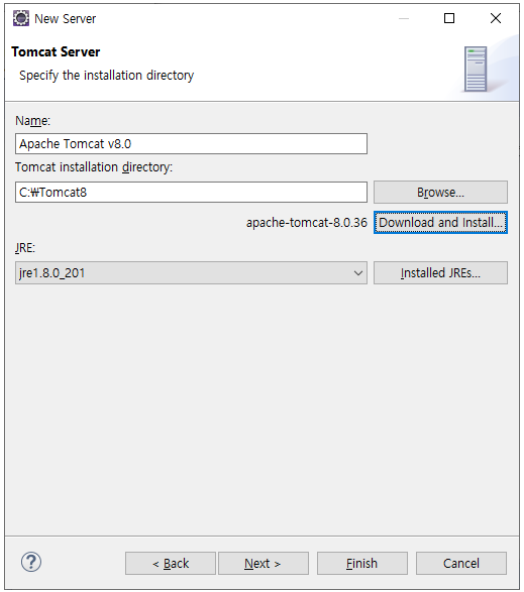
라이선스 관련 창
"I accept the term of the license agreement"
를 선택하여 [Finish] 버튼을 클릭



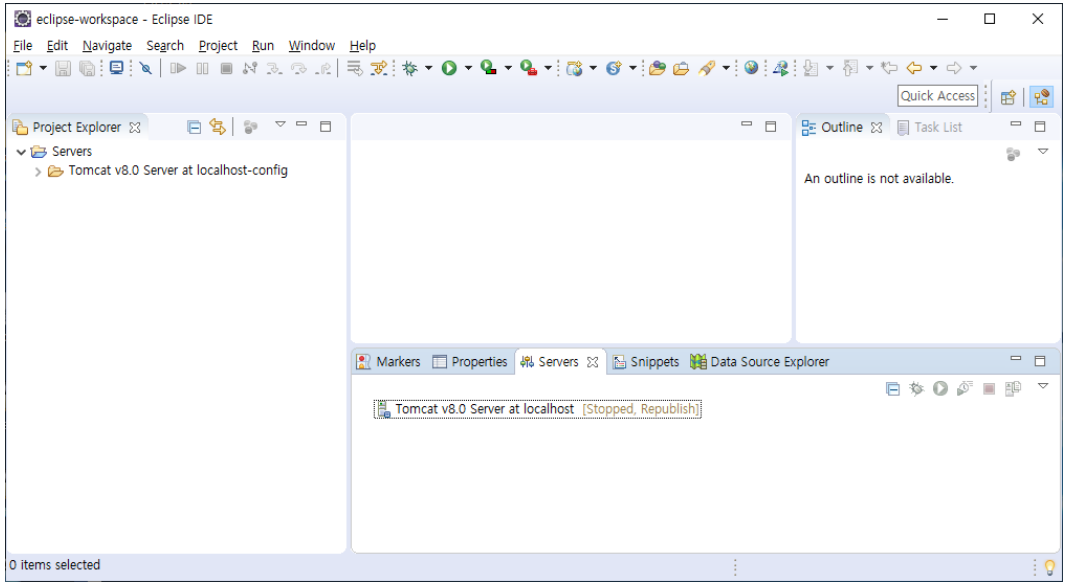
폴더 생성 또는 선택 후 [폴더 선택] 버튼을 클릭하면
Tomcat 다운로드 작업이 수행

톰켓(Tomcat) 설치하기

- 웹 애플리케이션 구동을 위해 아파치 재단에서 개발한 오픈 소스 소프트웨어인 Apache Tomcat에 대한 설치



다운로드가 완료되면 [Finish] 버튼이 활성화 되며 버튼을 클릭하면 설치가 완료



이클립스 IDE의 Project Explorer의 Servers와 하위의 Servers 탭을 확인해 보면 Tomcat v8.0 버전이 설치되었음을 확인 가능