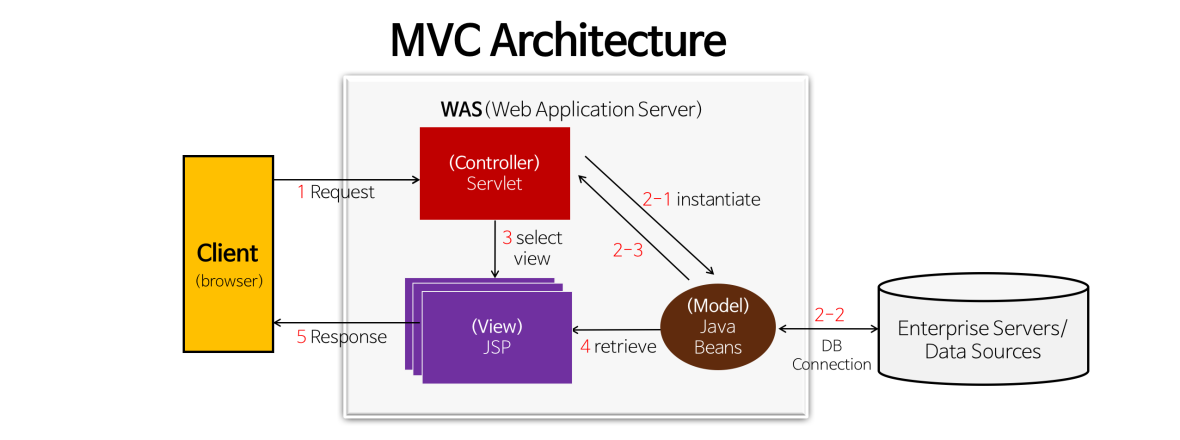
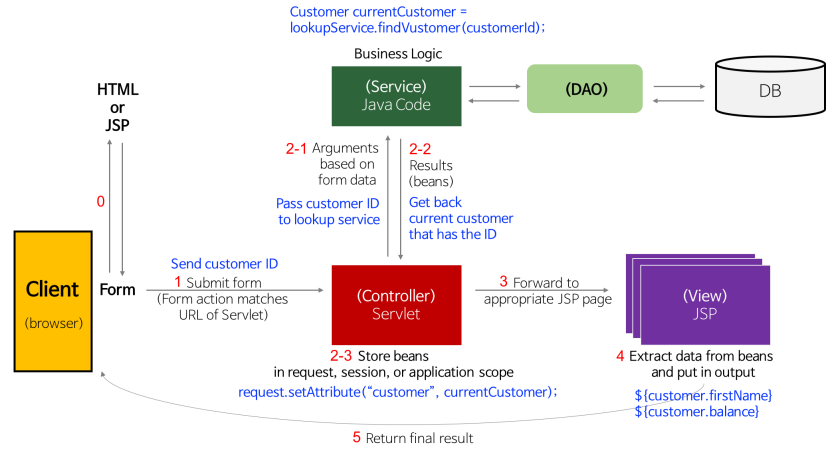
* MVC Architechture
* Mvc architecture란
  + Model, view, Controlle를 분리한 디자인 패턴
    - Model
      * 애플리케이션의 상태(data)를 나타낸다
      * Java beans
    - Views
      * 디스플레이 데이터 또는 프레진테이션
      * Jsp
    - Controller
      * View와 model사이의 인터페이스 역할
      * Model/View에 대한 사용자 입력 및 명령을 수신하여 그에 따라 적절하게 변경
      * Servlet
* JSP와 Servlet을 모두 이용하는 모델(MVC Architecture)



* + JSP와 Servlet을 모두 사용하여 프레젠테이션 로직(View)와 비즈니스 로직(Controller)을 분리한다.
    - View(보여지는 부분)는 html이 중심이 되는 JSP를 사용
    - Controller(다른 자바 클래스에 데이터를 넘겨주는 부분)는 JAVA 코드가 중심이 되는 Servlet을 사용
    - Model은 Java Beans로 DAO를 통해 MYSql과 같은 Data Strorage에 접근
  + 동작 과정

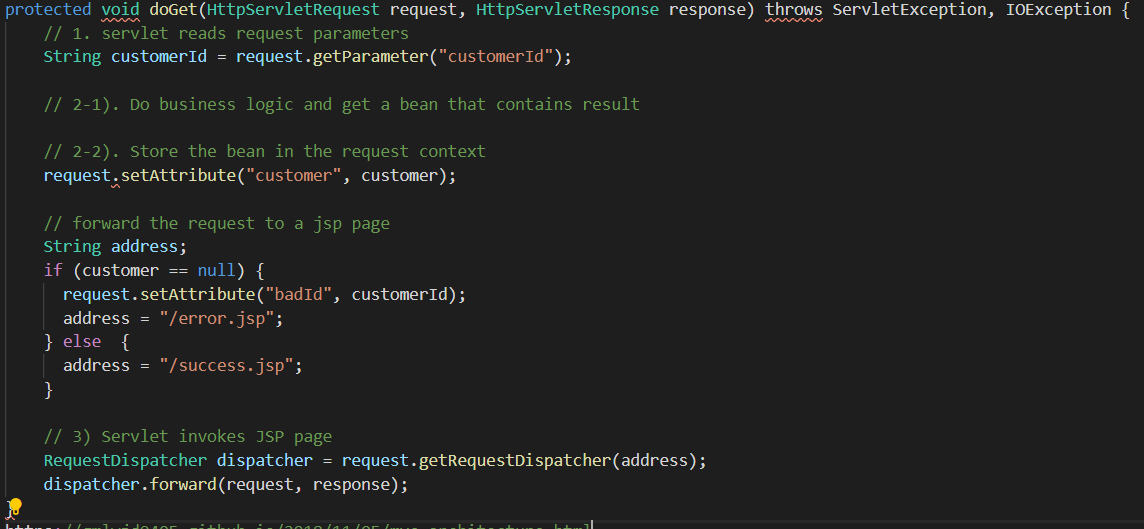
1. 클라이언트(브라우저)는 Servlet으로 요청을 보낸다
2. Servlet은 DB와 연결된 Java Bean 객체를 생성한다
3. Java Bean은 DB에서 적절한 정보를 가져와 저장한다.
4. Servlet에서 추가적인 비즈니스 로직 과정을 수행한다.
5. Servlet 은 JSP페이지와 통신한다
6. JSP페이지는 Java Bean 과 통신한다
7. JSP페이지가 클라이언트에 응답한다.

* 구체적인 MVC Flow of Control(Annotated)



1. WebServlet 또는 web.xml의 url-pattern에 해당하는 Servlet으로부터
2. 사용자가 입력한 form data를 Service에 인자들로 넘긴다
3. Service
   1. DAO와 DB를 거쳐 결과값을 가져온다.
   2. Business Logic을 처리한다.
4. 결과값을 bean객체에 저장한다.
5. Controller
   1. Request 객체에 결과값(bean)을 저장한다
6. 적절한 JSP페이지로 전달한다.
7. Bean객체로부터 data를 추출하고 output을 생성한다
8. View
   1. JSP Action Tag나 JSP EL(Expression Language)를 사용하여 bean 객체에 적절한 property를 넣는다
   2. 적절한 동적인 페이지를 만들어 반환한다.
9. 동적인 페이지 결과를 클라이언트(브라우저)에 전달한다.

* RequestDispatcher
* Implementing MVC with RequestDispatcher
  + Define bean to represent result data
    - 결과 데이터를 표현하기 위해 beans를 정의한다
    - 하나 이상의 getXXX 메서드가 있는 Java 클래스
  + Use a servlet to handle request
    - 사용자 요청을 다루기 위해 Servlet을 사용한다
    - Servlet은 요청 파라미터를 읽어 누락된 데이터와 잘못된 데이터를 확인하고 비즈니스 로직을 호출
  + Perform business logic and get a bean that contain result
    - 비즈니스 로직을 수행하고 결과를 포함하는 bean을 가져온다
    - Servlet은 비즈니스 로직(application-specific code)또는 데이터 접근 코드를 호출하여 결과를 가져온다
  + Store the bean in the request, session or Servlet context
    - Servlet은 요청의 결과를 나타내는 bean에 대한 참조를 저장하기 위해 request, session또는 servlet context에 대해 setAttribute를 호출한다.
  + Forward the request to a JSP page
    - JSP페이지에 요청을 전달한다.
    - Servlet은 상황에 적합한 JSP페이지를 결정하고, RequestDispatcher forward메서드를 사용하여 해당 JSP 페이지로 제어를 전송한다
  + Extract the data from the beans
    - bean으로부터 data를 추출한다
    - JSP 1.2(이전버전)
      * JSP페이지는 jsp:useBean 및 4단계의 location와 일치하는 scope가 있는 bean에 접근한다.
      * 그런 다음 jsp:getProperty를 사용하여 bean properties를 출력한다.
    - JSP 2.0(선호)
      * JSP페이지는 ${nameFromServlet.property}를 사용하여 bean properties를 출력한다.
    - 어느 쪽이든 JSP페이지는 bean을 생성하거나 수정하지 않는다
    - Servlet이 생성한 데이터를 단순히 추출하여 표시한다.
* 예시



* Servlet이 사용자 요청에 응답한다
  + Request.getPrameter 를 이용하여 form data를 가져온다
* Servlet은 비즈니스로직을 호출한다.
  + 비즈니스 로직의 결과를 포함하는 bean을 가져온다
  + 요청의 결과를 나타내는 bean을 HttpServletRequest httpsession또는 servletContext에 저장한다
* Servlet은 RequestDispatcher.forward를 통해 JSP페이지를 호출한다
* JSP페이지는 bean에서 데이터를 읽는다
  + JSP2.0 : ${beanName.propertyName}
  + JSP1.2 : 적절한 Scope(request, session, 또는 application)의 jsp:useBean 및 jsp:getProperty 이용
* 참고 scopes
  + Scopes는 bean이 저장되는 공간
  + 세가지 scope
    - Request
      * 요청에 저장된 data는 servlet과 servlet이 전달하는 jsp페이지에 표시된다
      * Data는 다른 사용자 또는 다른 페이지에서 볼 수 없다
      * 가장 일반적인 scope
    - Session
      * 요청에 저장된 data는 servlet과 servlet이 전달하는 jsp페이지에 표시된다
      * 동일한 사용자인 경우 data를 다른 페이지 또는 나중에 볼 수 있다
      * 다른 사용자는 data를 볼 수 없다.
      * 흔하게 사용하는 scope
    - Application(Servlet Context)
      * Servlet Context에 저장된 데이터는 모든 사용자의 응용 프로그램의 모든 페이지에서 볼 수 있다.
      * 거의 사용하지 않는 scope
    - Spring Bean Scope참조
* 참조 bean 객체
  + 특정 규칙을 따르는 Java Class
  + 규칙
    - 인수가 0인 기본 생성자가 반드시 있어야 한다
      * 기본 생성자를 명시적으로 정의하거나
      * 모든 생성자를 생략하거나
    - Public 인스턴스 변수(필드)가 없어야 한다
    - Persistent value.s는 getXXX 및 setXXX메서드를 통해 접근해야 한다

