서블릿

|  |  |
| --- | --- |
| 1 | 다음 설명 중 맞는 것을 모두 고르면?  (1) 오직 서버에서만 실행될 수 있는 JavaEE 기반의 클래스를 서블릿이라 한다.  (2) 서블릿의 인스턴스는 프로그래머가 new 연산자를 통하여 메모리에 올려야 한다.  ->(X) 톰캣이 최초의 접속자에 의해서 메모리에 올린다  (3) 서블릿의 인스턴스는 클라이언트의 요청이 있을 때마다 메모리에 생성되어진다.  ->(X) 인스턴스는 메모리에 한번만 생성된다  (4) 클라이언트가 서블릿을 호출하기 위해서는, URL매핑을 작성해야 하는데, 매핑의 작성은 web.xml  이라는 DD( Deployment Descriptor )파일을 편집하면 된다.  ->DD : 배포설명자  (5) web.xml 파일은 톰켓에서만 지원되는 파일이며, 서버 제품마다 서블릿을 매핑하는 방법이 서로 틀리다 ->(X)  (6) 서버가 가동될때 web.xml을 읽어들이므로, 만약 web.xml의 내용이 수정되었다면 서버를 재가동해야  변경된 web.xml이 적용되어 진다 |
| 2 | 서블릿에 대한 설명 중 틀린 것은?  (1) 서블릿 코드 안에 HTML 태그를 작성할 경우 디자인을 표현하는 코드가 복잡하고, 유지보수성이  떨어진다는 서블릿의 단점은 JSP로 해결할 수 있다.  (2) JSP는 서블릿과는 공통점이 전혀 없는 또 다른 기술이다.  (3) JSP는 서블릿에 대한 지식 및 작성에 대한 부담이 없으므로, 비교적 쉽게 웹서비스를 구축할 수 있도록  돕는 스크립트 언어 이다  (4) 개발자가 JSP를 작성하더라도, 결국 JSP는 서블릿으로 변환되어 실행된다 |
| 3 | 다음 중 틀린 것은?  (1) 서블릿은 톰켓과 같은 웹컨테이너에 의해서만 자동으로 생성되어지며 프로그래머가 작성할 수는 없다.  ->(X) 톰캣이 작성하기도하고 프로그래머가 JSP를 작성 시 톰캣이 코딩함  (2) JSP는 바로 실행되어질 수 없으며, 실행 직전에 서블릿으로 자동 변환되어 실행된다.  (3) 사실 개발자는 JSP를 통해 간접적으로 서블릿을 작성하는 것과 같다 .  -> JSP는 사실 서블릿이야!  (4) JSP의 <%! 선언부 %> 영역에 코드를 작성하면, 서블릿 클래스의 멤버 영역에 위치하게 된다.  (5) JSP의 <% 스크립틀릿%> 영역에 코드를 작성하면, 서블릿 클래스의 service() 메서드 영역에 위치하게 된다  -> service()메서드의 지역변수가 됨 |
| 4 | 다음 설명 중 맞는 것은?  (1) 서블릿은 JSP이다 ->(X) JSP가 서블릿이다  (2) JSP가 서블릿으로 변경되는 시점은 클라이언트의 요청이 있을 때마다 이다.  ->(X) 최초에 들어온 JSP만 서블릿으로 변경되고 그 이후에 들어온 아이들은 실행만 된다  ->최초의 요청이 있을 때에만 서블릿으로 변경된다  (3) JSP를 서블릿으로 변경하는 주체는 개발자이다. ->(X) 톰캣과 같은 웹컨테이너이다  (4) JSP는 클라이언트의 최초 요청이 있을때 서블릿 클래스로 변경되고 인스턴스가 생성되며, 이 시점부터  는 매 요청이 있을 때마다 1:1로 쓰레드가 생성되어 요청을 처리하게 된다. |
| 5 | 다음 설명 중 틀린 것은?  (1) 서블릿 클래스는 HttpServlet 형 이며, 서블릿 계보의 최상위 객체는 Servlet 인터페이스이다, 이 인터 페이스에는 서블릿으로서 갖추어쟈 할 가장 기본이 되는 메서드가 정의되어 있다  (2) Sun에서 서블릿을 추상클래스로 선언한 이유는, 웹 요청이 있을때 어떠한 기능을 구현할지는 개발자가  결정해야 하기 때문이다.  ->doXXX메서드는 개발자가 만듦  ->톰캣은 service메서드만 호출함  (3) JavaEE 기반의 서버 제품마다 각각 기술력이 틀리므로, Tomcat에서는 웹요청을 서블릿이 처리하지만, JBoss에서는 전혀 다른 명칭의 객체와 메서드가 웹요청을 처리한다 ->(X)    (4) 개발자는 서블릿 클래스를 작성한 후, new 연산자로 인스턴스를 미리 생성해 놓아야 클라이언트의  요청을 처리할 수 있게 된다.  ->(X) 최초의 요청에 의해 한번만 생성됨 |
| 6 | 다음 설명 중 맞는 것은?  (1) 서블릿의 메서드 중 init(),service(),destroy() 를 생명주기 메서드라 한다.  (2) init() 메서드는 서블릿의 초기화를 담당하는 생성자이다. ->(X)  ->생성자가 먼저 호출된 후 init()호출한다 둘은 별개!  ->init()은 출생의 비밀을 알려줌  (3) init() 메서드는 service() 메서드가 호출 되고 난 직후에 컨테이너에 의해 호출되어 진다.  ->(X) init()은 단 한번 호출되고 service()는 클라이언트가 요청할때마다 호출됨  (4) init() 메서드는 서블릿의 일생에 단 한번만 호출되어 진다. |
| 7 | HttpServlet 클래스의 생명주기에 대한 설명 중 틀린 것은?  (1) 개발자에 의해 작성된 서블릿은 최초의 요청을 시도하는 클라이언트에 의해 인스턴스가 생성되며, 이때 서블릿은 초기화도 진행이 되는데, 이때 초기화를 담당하는 생명주기 메서드는 service() 이다  ->(X) init()이다  (2) 일단 서블릿의 인스턴스가 생성된 후에는 클라이언트의 요청이 추가로 발생하더라도 인스턴스는 중복  생성되지 않으며, 대신 쓰레드가 생성되어 요청을 처리하게 된다, 이때 이 요청을 처리하기 위한 생명주기 메서드가 바로 init() 메서드 이다  ->(X) service()이다  (3) 클라이언트의 요청이 있을때, 서버측에서는 이 요청을 처리할 쓰레드가 1:1로 생성되며, 이 쓰레드에  의해 서블릿의 service() 메서드가 실행되어 진다.  (4) service() 메서드가 요청과 응답을 처리하기 위해서는 클라이언트의 요청에 대한 정보뿐만 아니라,  응답을 위해서 사용될 객체가 필요하므로, service() 메서드의 API 에는 매개변수로써 HttpServletRequest형과 HttServletResponse 형을 받도록 정해져 있다.  (5) 클라이언트의 요청이 Get방식일 경우, 서블릿의 doGet() 메서드가 작동하며, doGet()은 다시 service()메서드를 호출한다.  ->(X)service()가 doGet()을 호출한다 |
| 8 | 다음 설명 중 틀린 것은?  (1) 개발자에 의해 작성된 서블릿을 서버에 배포해 놓으면 최초의 요청을 시도하는 클라이언트에 의해  인스턴스가 생성되며, 이때 서블릿은 초기화가 진행되는데, 이때 초기화를 담당하는 생명주기 메서드가  service() 이다  ->(X) init()  (2) doGet이나 doPost 와 같은 doXXX()형 메서드는 service() 메서드에 의해 호출되어 진다.  (3) service() 메서드는 쓰레드에 의해 실행된다.  (4) init() 메서드는 쓰레드에 의해 실행된다. ->(X)톰캣에 의해 호출된다  (5) 서블릿의 인스턴스가 메모리에 올라가는 시점은, 개발자가 컴파일을 수행 한 직후부터 이다.  ->(X) 톰캣에 의해 올라간다  (6) 실질적으로 http상의 요청을 처리하기 위해 개발자가 재정의해야 하는 메서드는 doXXX()형 메서드가 아 닌 service() 메서드이다.  ->(X) service()메서드가 doXXX()형 메서드를 호출한다 |
| 9 | 다음 설명 중 맞는 것은?  (1) 서블릿의 생명주기 메서드 중 가장 먼저 호출되어 지는 메서드는 init() 이며 클라이언트의 요청이 있을때마다 호출되어 진다. ->(X) init()은 최초의 한번만 호출된다  (2) init() 메서드는 서블릿의 초기화를 담당하므로, 서블릿의 생성자이다. ->(X)  (3) 서블릿은 웹컨테이너가 생성하므로, 생성자가 존재할 수 없다. ->(X)  (4) 서블릿의 생명주기 와 관련된 메서드에는 init() , service(), destory() 가 있다. |
| 10 | 다음 설명 중 틀린 것은?  (1)서블릿의 인스턴스가 생성되는 시점에 특정 문자열 정보를 전달할 수 있는데, 이 정보를 서블릿에게  전달해주는 객체가 바로 ServletConfig 인터페이스이다  (2)서블릿 객체 생성시, 생성자 보다 앞서 호출되는 메서드가 바로 init() 메서드이다.->(X)  (3) ServletConfig 객체는 init() 메서드의 인수로 전달되어 진다.  (4) init() 메서드를 통해 ServletConfig 정보를 전달하기 위해서는 web.xml 파일에 필요한 데이터를  작성하면 된다. |
| 11 | 서블릿에 대한 설명 중 틀린 것은?  (1) HttpServletRequest는 사실, JSP에서의 request 내장 객체의 데이터 타입이다.  (2) HttpServletResponse는 사실 JSP에서의 response 내장 객체의 데이터 타입이다.  (3) HttpServletRequest, HttpServletResonse 객체는 인터페이스이므로 프로그래머가 new 할 수 없으며  요청이 있을 때마다 컨테이너에 의해 메모리에 생성된다  (4) 클라이언트의 요청에 대한 응답처리 한 사이클이 끝날 때, 서버에 생성된 요청처리 쓰레드와 request  , response 객체들은 모두 소멸된다   * 11번은 틀린 답이 없음 |
| 12 | 다음 설명 중 틀린 것은?  (1) Servlet 인터페이스는 서블릿으로서 반드시 가져야 할 가장 기본이 되는 5가지 메서드만을 정의해 놓은 서블릿 API의 최상위 객체이다    (2) 프로토콜에 중립적인 서블릿을 정의해 놓은 클래스는 GenericServlet 클래스이다.  (3) HttpServlet 클래스는 GenericServlet 클래스를 상속 받으며, HTTP기반의 요청을 처리하기에 최적화 된 서블릿이다.    (4) GenericServlet 은 Servlet 클래스를 상속받고 있으며, HttpServlet 은 GenericServlet 인터페이스를 구현 하고 있다.  ->(X) Servlet은 인터페이스이다  ->GenericServlet은 인터페이스가 아니다  (5) Sun에서는 서블릿 API의 스팩만을 명시하며 이 서블릿의 API를 실제 구현할 의무는 서버 제작자에게  있으므로 서블릿과 관련 JavaEE API의 구현체인 jar 파일은 현재 사용중인 서버가 보유하고 있다. |