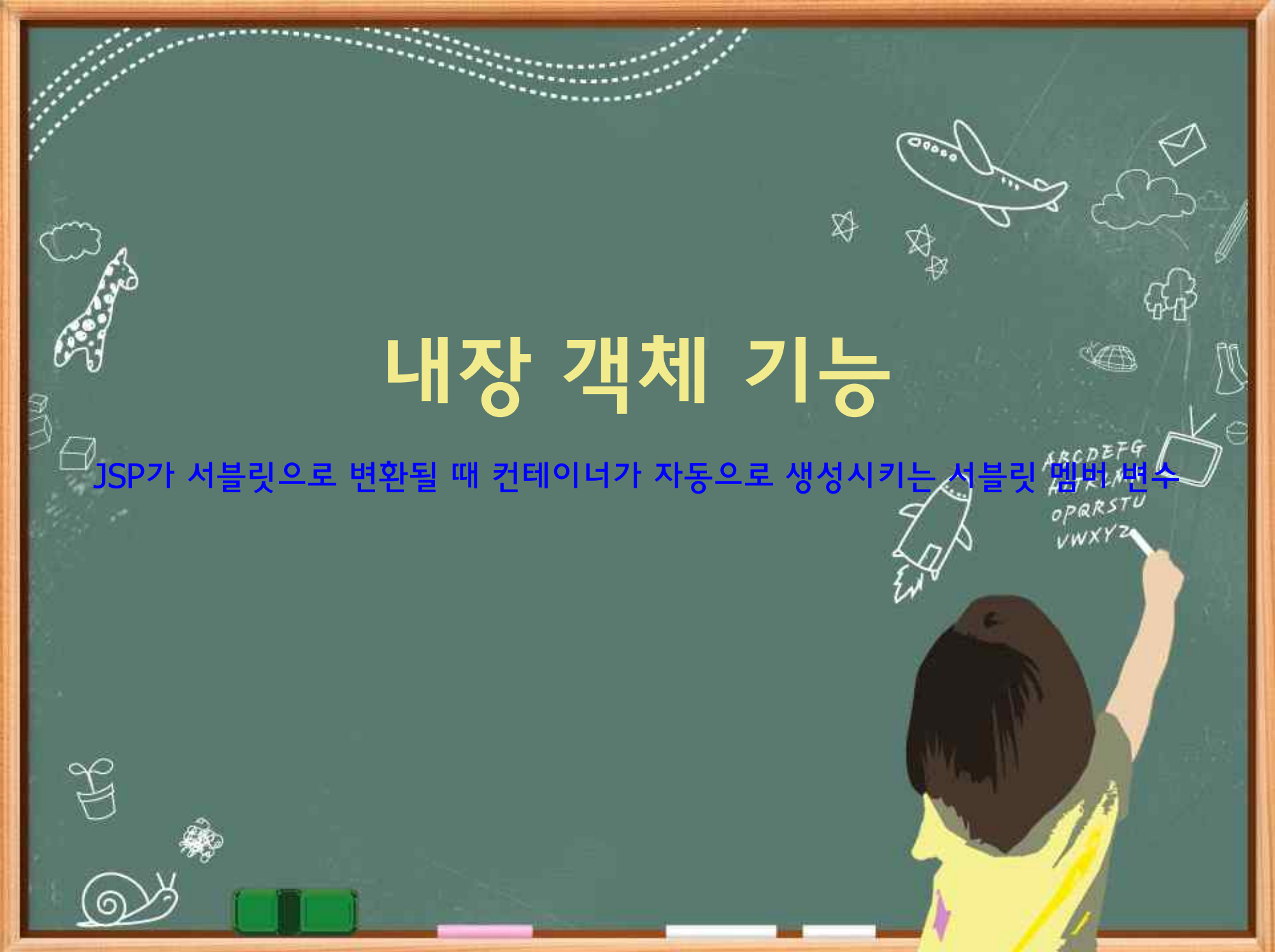

내장 객체 기능
JSP가 서블릿으로 변환될 때 컨테이너가 자동으로 생성시키는 서블릿 멤버변수

자주 사용되는 내장 객체

- request : 한 번의 요청에 대해 같은 요청을 공유하는 JSP페이지 공유
- session : 같은 브라우저에서 공유
- application : 같은 애플리케이션에서 공유

session 내장 객체 실습

```java
@WebServlet("/sess")
public class SessionTest extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
      response.setContentType("text/html; charset=utf-8");
      PrintWriter pw = response.getWriter();
      HttpSession session = request.getSession();
      session.setAttribute("name", "이순신");

      pw.println("<html><body>");
      pw.println("<h1>세션에 이름을 바인딩합니다.</h1>");
      pw.println("<a href='/pro12/test01/session1.jsp'>첫번째 페이지로 이동하기 </a>");
      pw.println("</body></html>");
  }
}
```

getSession() 메서드를 이용해 session 객체를 얻은 후 name을 바인딩

session 내장 객체 실습

```
<%
   String name = (String)session.getAttribute("name");
   session.setAttribute("address", "서울시 강남구");
%>
```

getAttribute() 메서드를 이용해서 session에 바인딩된 name 값을 가져온 후
setAttribute() 메서드를 이용해 session에 address를 바인딩

session 내장 객체 실습

```jsp
<%
    String name=(String)session.getAttribute("name");
    String address=(String)session.getAttribute("address");
%>
```

getAttribute()를 이용해 서블릿과 JSP에서 session에 바인딩된 name과 address 값을 가져옴.

application 내장 객체 실습

```jsp
<%
    session.setAttribute("name", "이순신");
    application.setAttribute("address", "서울시 성동구");
%>
```

이름과 주소를 session과 application 내장 객체에 바인딩

```jsp
<%
    String name=(String)session.getAttribute("name");
    String address=(String)application.getAttribute("address");
%>
```

저장한 데이터를 session과 application 내장 객체에서 가져옴

request 내장 객체 실습

```jsp
<%
    request.setAttribute("name", "이순신");
    request.setAttribute("address", "서울시 강남구");
%>
```

이름과 주소를 request 내장 객체에 바인딩

```jsp
<%
    RequestDispatcher dispatch = request.getRequestDispatcher("request2.jsp");
    dispatch.forward(request, response);
%>
```

request 객체를 다른 JSP로 포워딩

request 내장 객체 실습

```jsp
<%
    String name=(String)session.getAttribute("name");
    String address=(String)session.getAttribute("address");
%>
```

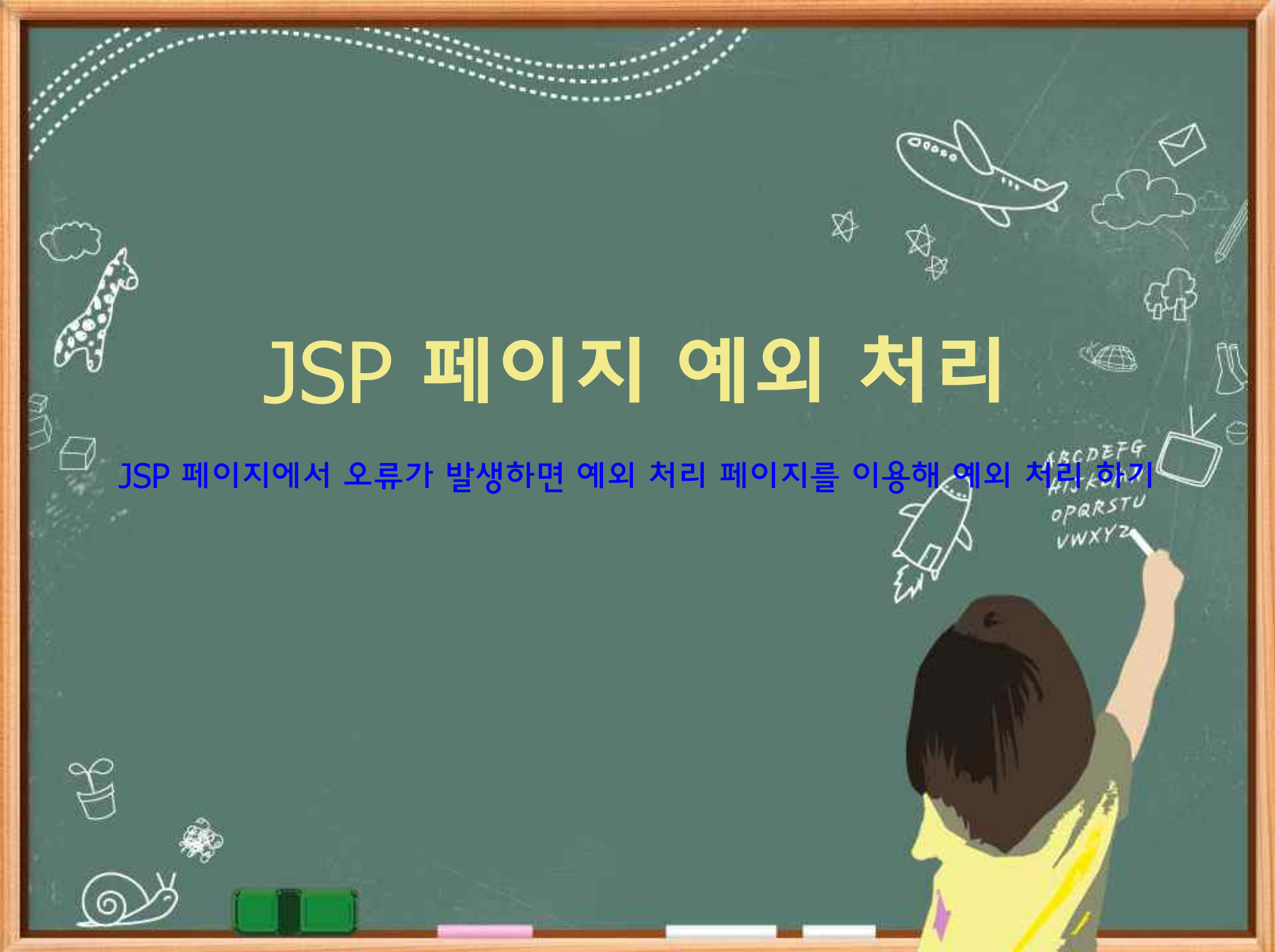
포워딩된 request객체에서 getAttribute()를 이용해 정보를 가져옴

```jsp
<body>
    <h1>이름은 <%=name %>입니다.</h1>
    <h1>주소는 <%=address %>입니다.</h1>
</body>
```

출력

JSP 페이지 예외 처리

JSP 페이지에서 오류가 발생하면 예외 처리 페이지를 이용해 예외 처리 하기



JSP 페이지 예외 처리 실습

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    errorPage="addException.jsp"%>
```

예외 발생시 예외를 처리할 JSP 페이지 지정

JSP 페이지 예외 처리 실습

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    isErrorPage="true"%>
```

<addException.jsp>
예외를 처리하는 예외 페이지로 지정

```jsp
<body>
    ==== toString() 내용 ==== <br>
    <h1><%=exception.toString() %></h1>
    ========getMessage() 내용 =======<br>
    <h1><%=exception.getMessage() %></h1>
    ========printStackTrace() 내용 ====<br>
    <h1><% exception.printStackTrace(); %></h1>
    <h3>
```

exception 내장 객체를 사용해 예외 처리

이클립스 콘솔로 예외 메시지 출력

에러 코드에 따른 예외 페이지

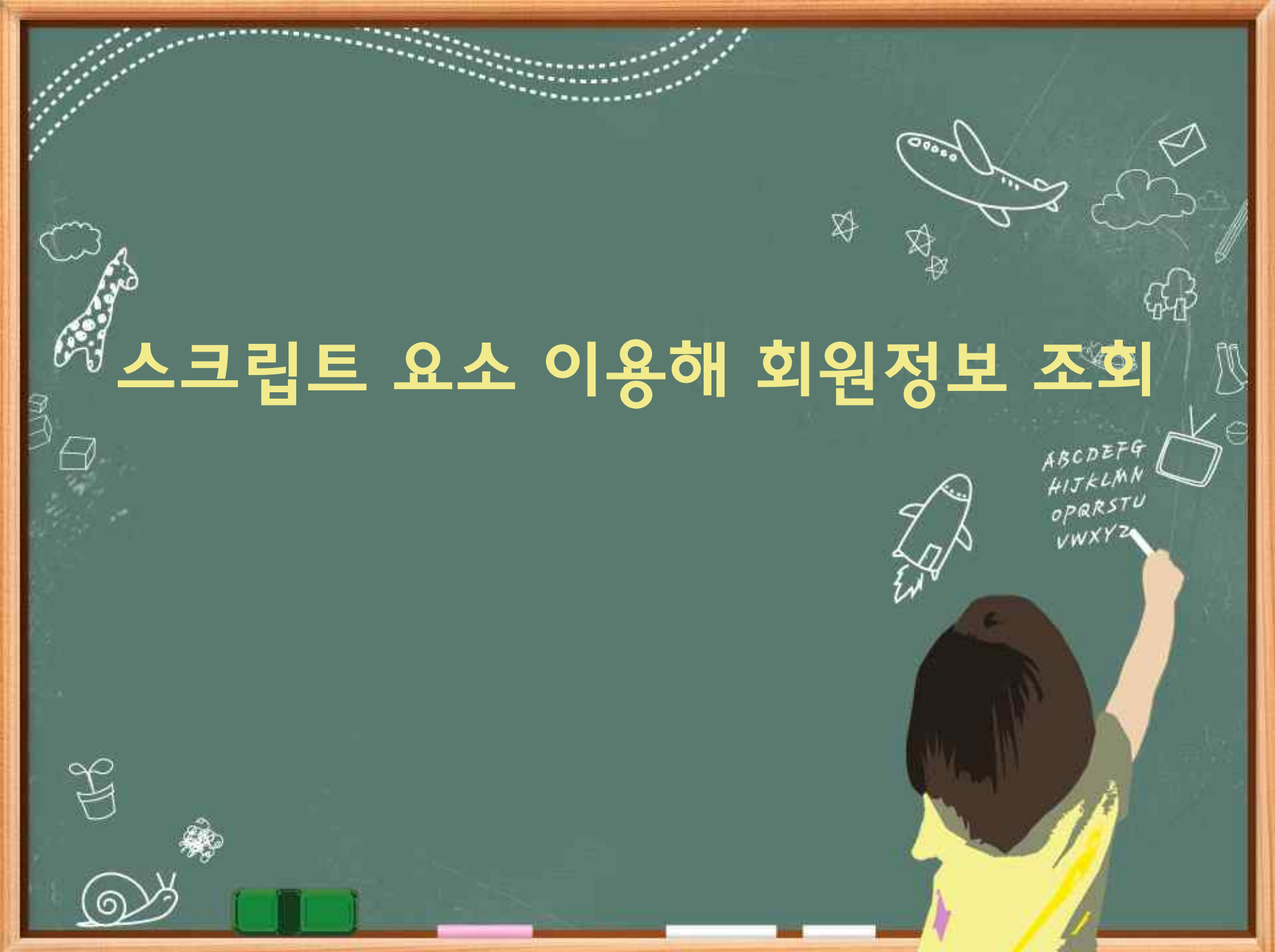
```
<error-page>
  <error-code>404</error-code>
  <location>/err/error_404.jsp</location>
</error-page>

<error-page>
  <error-code>500</error-code>
  <location>/err/error_500.jsp</location>
</error-page>
```

<web.xml>에 오류 페이지를 지정

404와 500 오류 발생 시 예외 처리를 할 페이지 지정

스크립트 요소 이용해 회원정보 조회



스크립트 요소 이용해 회원정보 조회

```
<form method="post" action="member.jsp">
 이름 : <input type="text" name="name"><br>
 <input type="submit" value="조회하기">
</form>
```

<search.jsp>
찾고자 하는 이름을 입력하면 member.jsp로 전송.

스크립트 요소 이용해 회원정보 조회

```jsp
<%
    request.setCharacterEncoding("utf-8");
    String _name = request.getParameter("name");
    MemberVO memberVO = new MemberVO();
    memberVO.setName(_name);
    MemberDAO dao = new MemberDAO();
    List membersList = dao.listMembers(memberVO);
%>
```

<member.jsp>
전송된 name 값을 가져온 후 MemberDAO 객체를 생성
listMembers() 메서드를 호출해 이름에 대한 회원 정보 조회

```jsp
for(int i = 0; i < membersList.size(); i++){
    MemberVO vo = (MemberVO)membersList.get(i);
    String id = vo.getId();
    String pwd = vo.getPwd();
    String name = vo.getName();
    String email = vo.getEmail();
    Date joinDate = vo.getJoinDate();
```

조회한 회원 정보를 for 반복문을 이용해 출력

스크립트 요소 이용해 회원정보 조회

```
public List listMembers(MemberVO memberVO) {
    List memberList = new ArrayList();
    String _name = memberVO.getName();
    try {
        con = dataFactory.getConnection();
        String query = "select * from t_member";
        if((_name != null && _name.length() != 0)) {
            query += " where name =?";
            pstmt = con.prepareStatement(query);
            pstmt.setString(1, _name);
        }else {
            pstmt = con.prepareStatement(query);
        }
```

<MemberDAO>
getName을 통해 조회할 이름을 가져오고 _name 값이 존재하면
SQL문에 where절을 추가하여 해당 이름 조회

_name 값이 없으면 모든 회원 정보 조회