

## 6.5 프로퍼티를 이용한 객체 관리

## 6.5 프로퍼티를 이용한 객체 관리

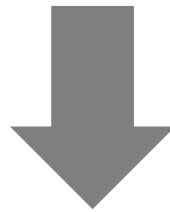
### “학습할 내용”

Dao 및 페이지 컨트롤러의  
객체 준비를 자동화하기

## 6.5 프로퍼티를 이용한 객체 관리

“학습할 내용”

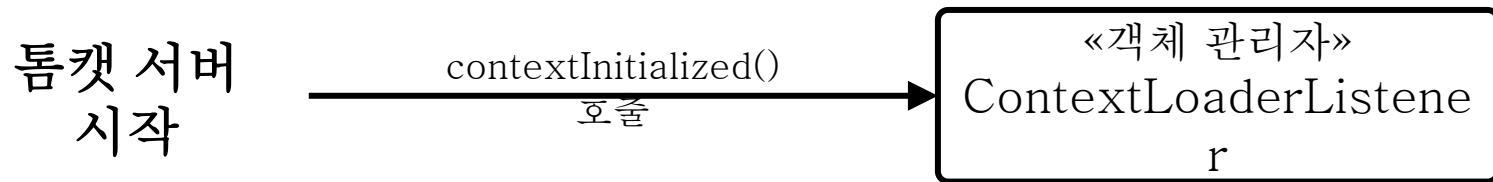
Dao 및 페이지 컨트롤러의  
객체 준비를 자동화하기



“ContextLoaderListener” 클래스 개선

## 6.5 프로퍼티를 이용한 객체 관리

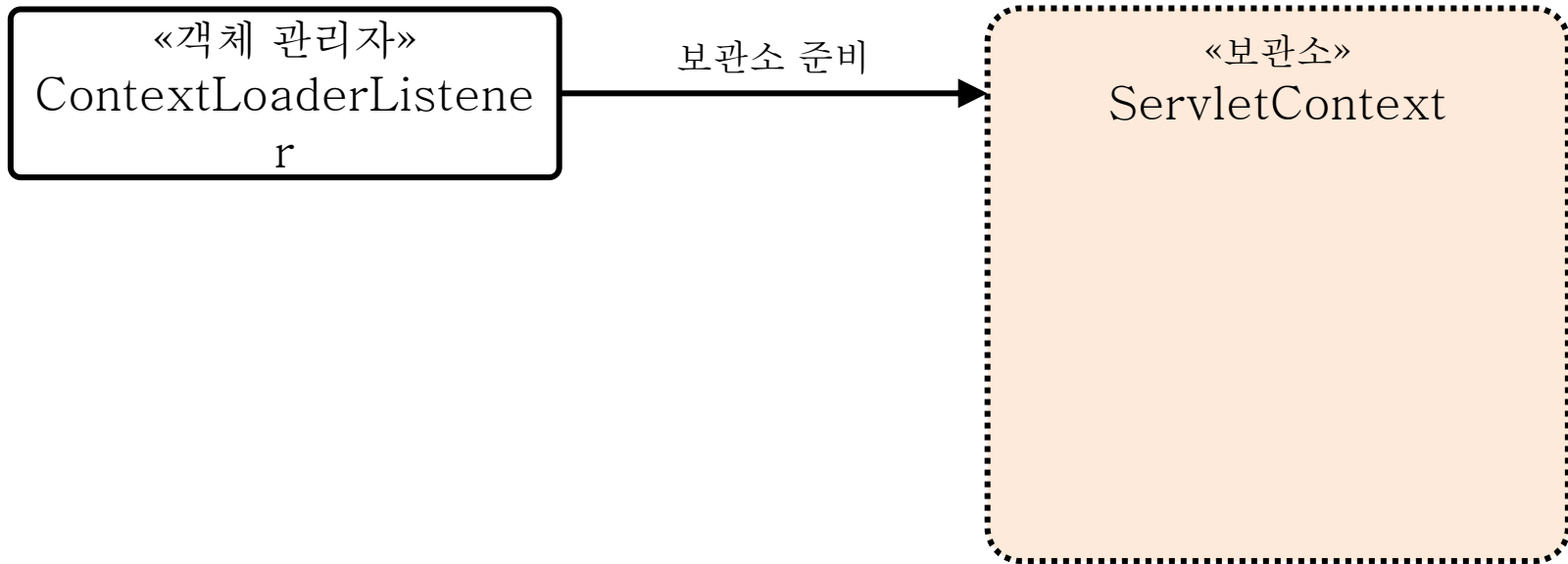
### 기존 방식



```
public void contextInitialized(ServletContextEvent event) {  
    // 객체 준비  
}
```

## 6.5 프로퍼티를 이용한 객체 관리

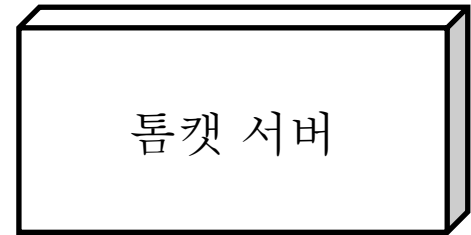
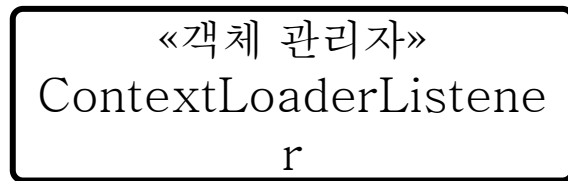
객체를 보관할 때 사용할  
서블릿 컨텍스트 객체를 준비



```
ServletContext sc = event.getServletContext();
```

## 6.5 프로퍼티를 이용한 객체 관리

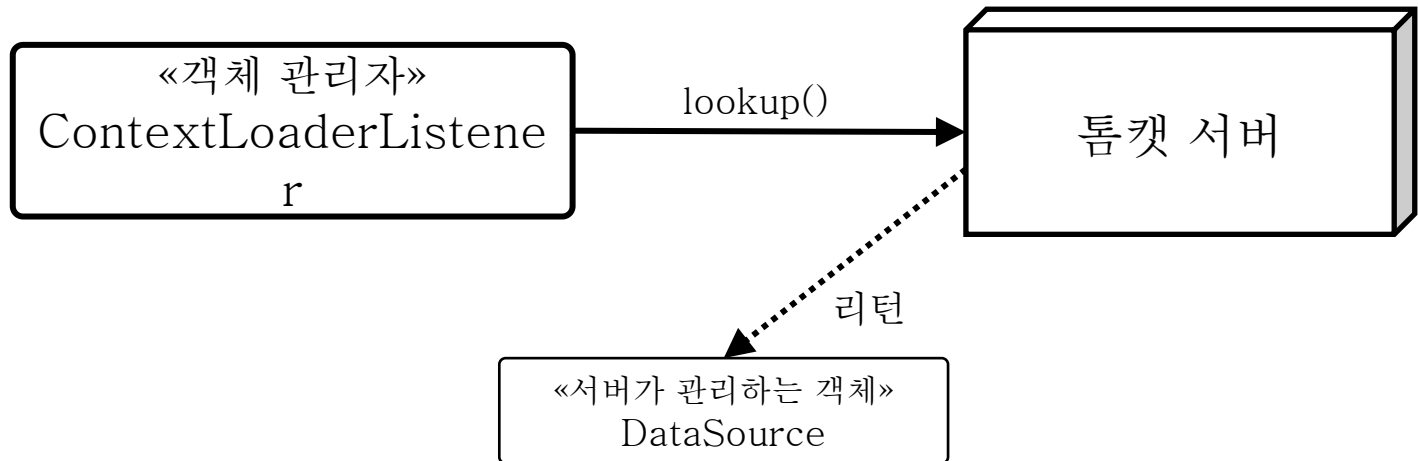
# 톰캣 서버가 관리하는 객체를 얻기 위한 준비



```
InitialContext initialContext = new InitialContext();  
DataSource ds = (DataSource)initialContext.lookup("java:comp/env/jdbc/studydb");
```

## 6.5 프로퍼티를 이용한 객체 관리

### DAO가 사용할 DataSource 객체를 얻기



```
InitialContext initialContext = new InitialContext();  
DataSource ds = (DataSource)initialContext.lookup("java:comp/env/jdbc/studydb");
```

## 6.5 프로퍼티를 이용한 객체 관리

### 페이지 컨트롤러가 사용할 DAO 객체 준비

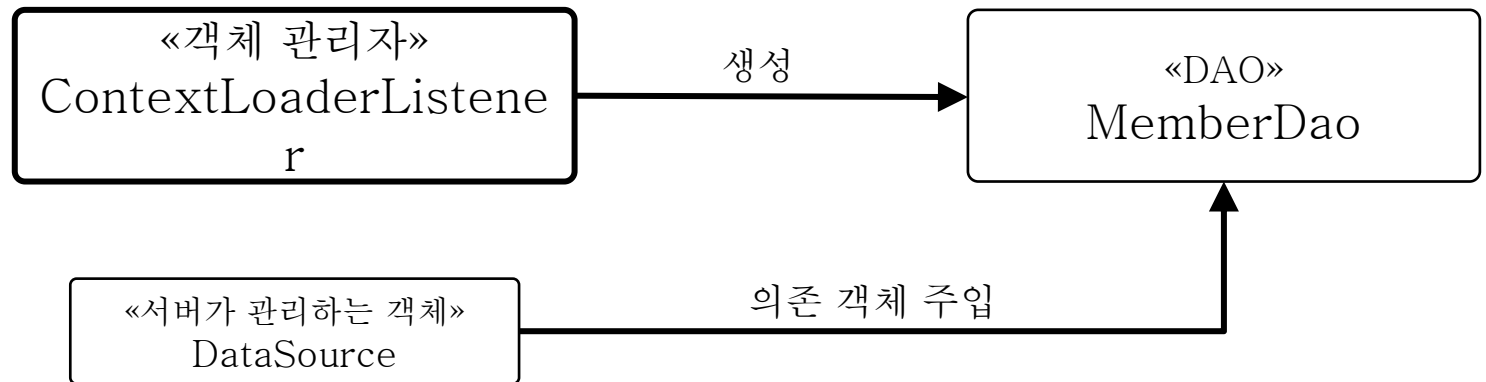


```
MySQLMemberDao memberDao = new MySQLMemberDao();
memberDao.setDataSource(ds);
```



## 6.5 프로퍼티를 이용한 객체 관리

# DAO 객체 생성 및 의존 객체(DataSource) 주입



```
MySQLMemberDao memberDao = new MySQLMemberDao();  
memberDao.setDataSource(ds);
```

## 6.5 프로퍼티를 이용한 객체 관리

### 페이지 컨트롤러 준비

«객체 관리자»  
ContextLoaderListener

«페이지 컨트롤러»  
LogInController

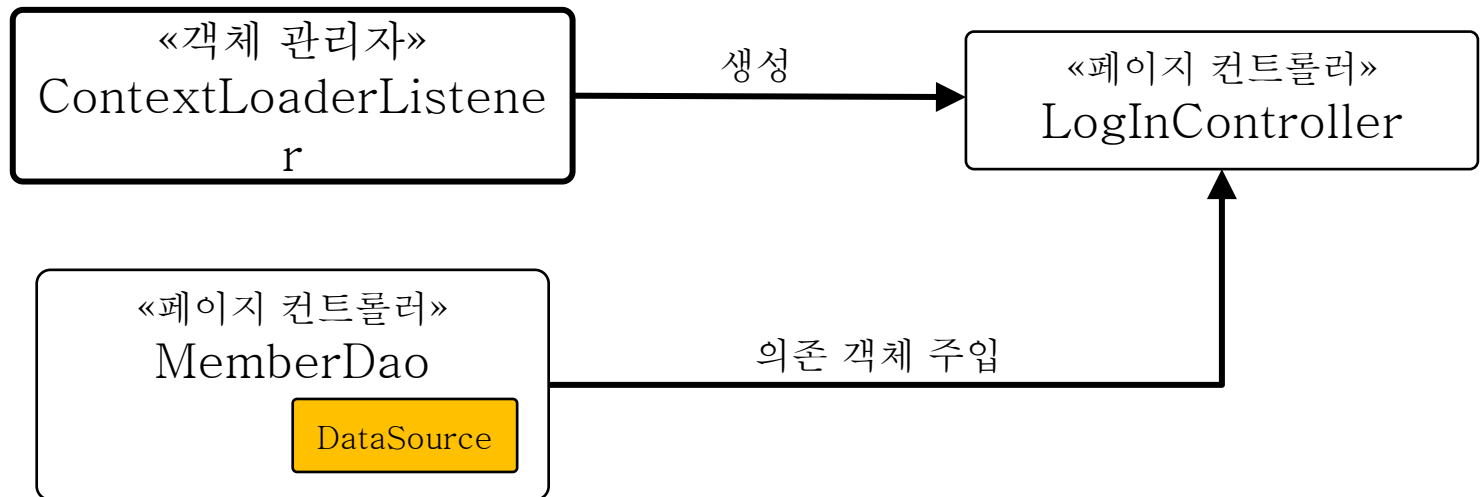
«페이지 컨트롤러»  
MemberDao

DataSource

```
sc.setAttribute("/auth/login.do",  
    new LogInController().setMemberDao(memberDao));
```

## 6.5 프로퍼티를 이용한 객체 관리

# 페이지 컨트롤러 객체 생성 및 의존 객체 주입

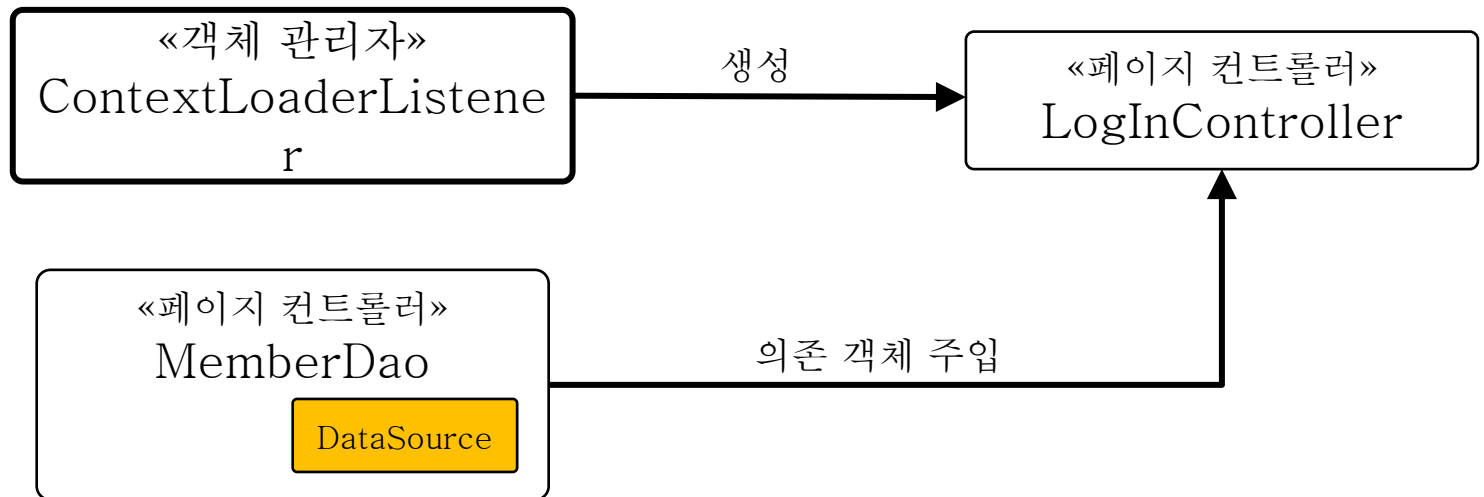


```
sc.setAttribute("/auth/login.do",  
new LoginController().setMemberDao(memberDao));
```

객체 생성

## 6.5 프로퍼티를 이용한 객체 관리

# 페이지 컨트롤러 객체 생성 및 의존 객체 주입



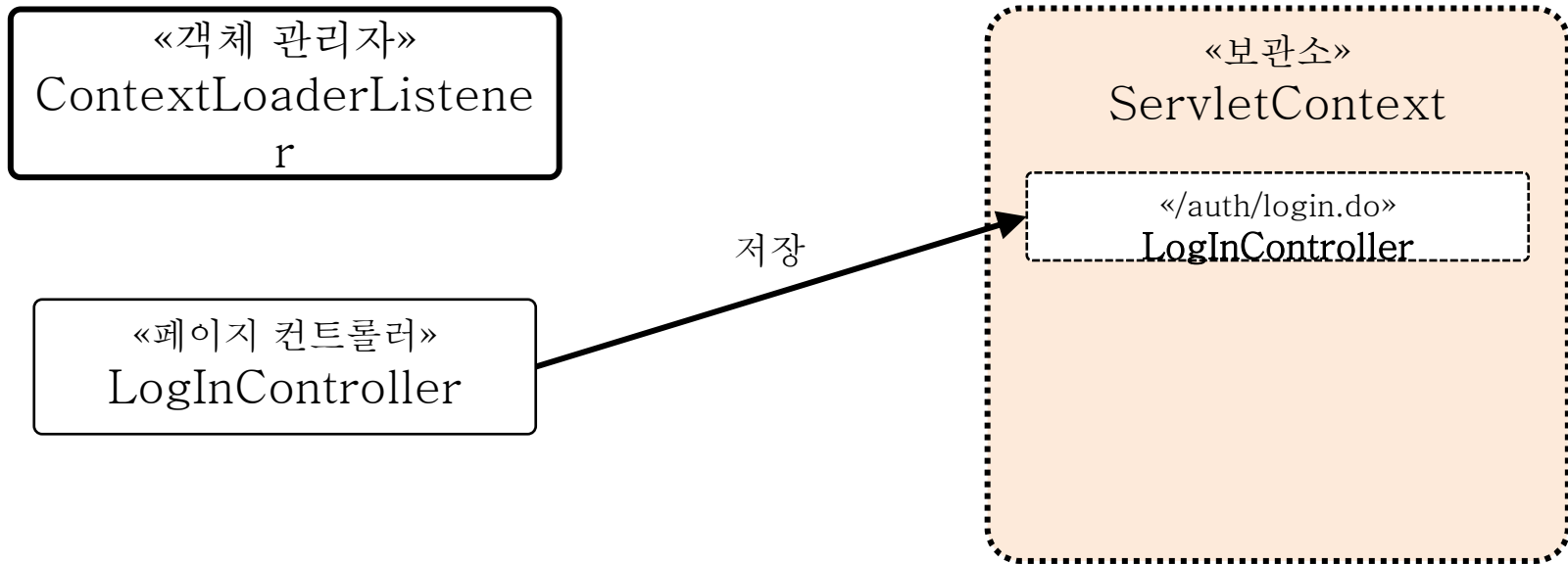
```
sc.setAttribute("/auth/login.do",  
new LoginController().setMemberDao(memberDao));
```

객체 생성

의존 객체  
주입

## 6.5 프로퍼티를 이용한 객체 관리

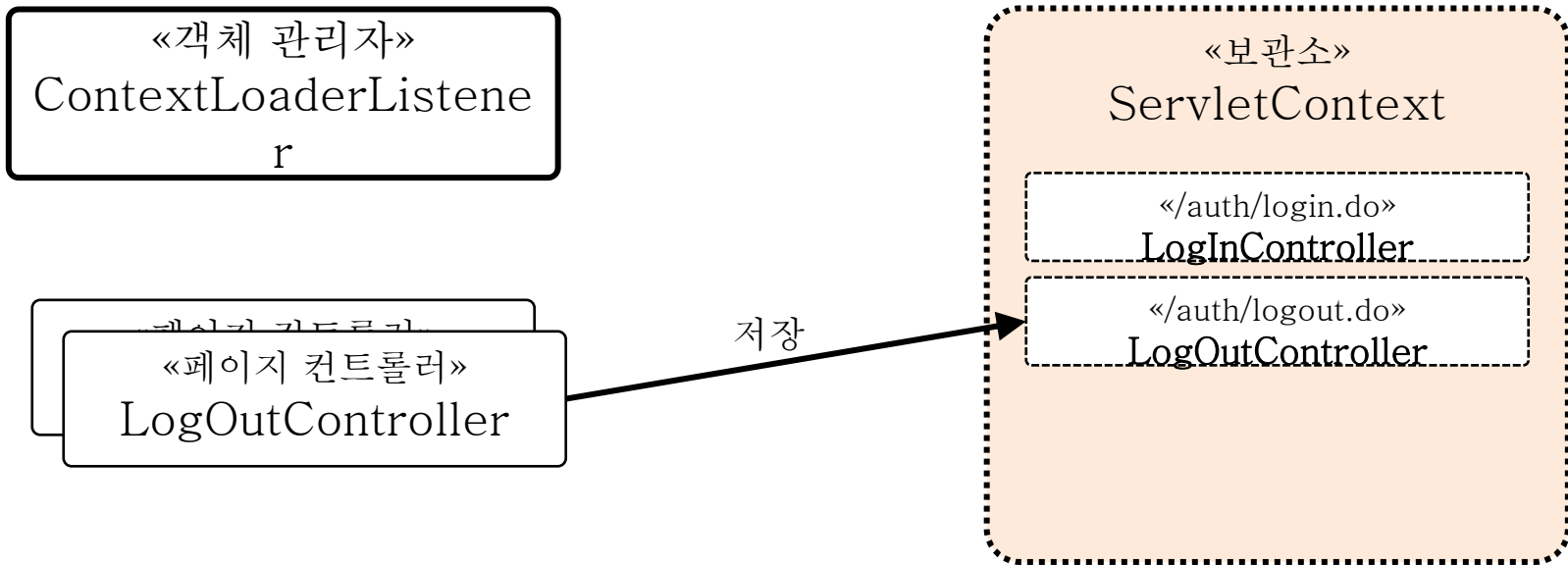
# 페이지 컨트롤러를 서블릿 컨텍스트에 보관



```
sc.setAttribute("/auth/login.do",  
    new LogInController().setMemberDao(memberDao));
```

## 6.5 프로퍼티를 이용한 객체 관리

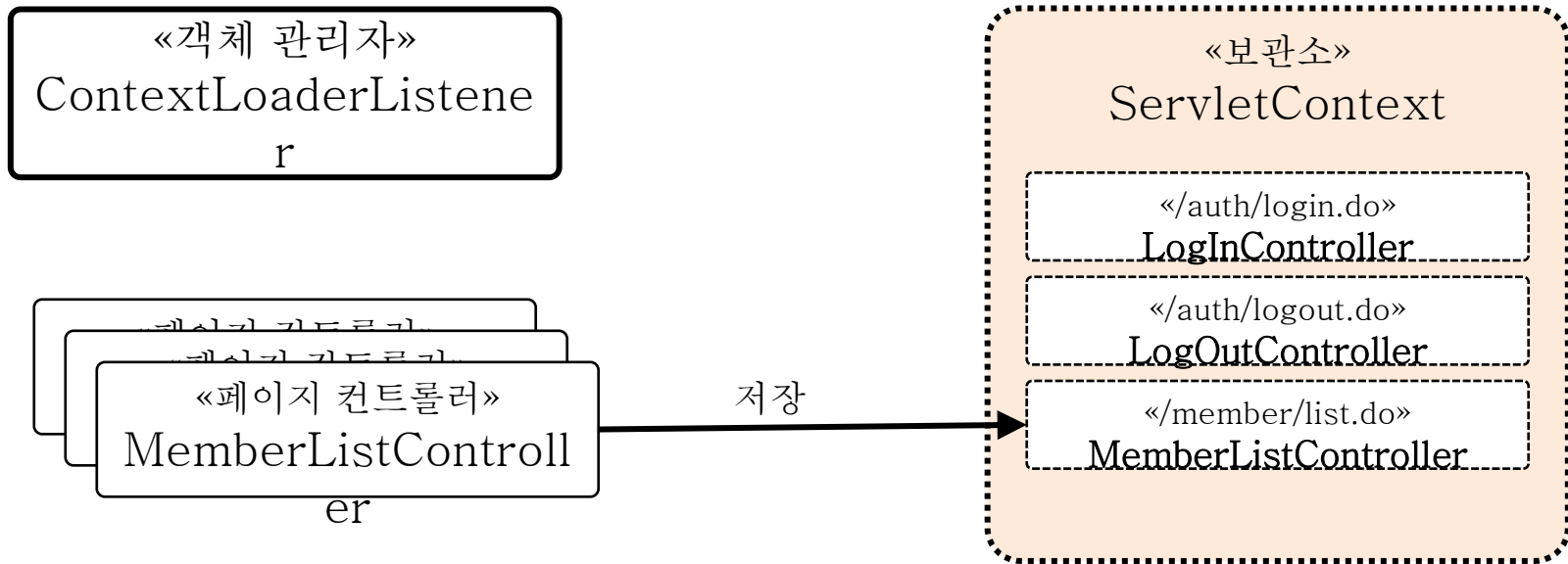
# 페이지 컨트롤러를 서블릿 컨텍스트에 보관



```
sc.setAttribute("/auth/logout.do", new LogOutController());
```

## 6.5 프로퍼티를 이용한 객체 관리

# 페이지 컨트롤러를 서블릿 컨텍스트에 보관



```
sc.setAttribute("/member/list.do",  
    new MemberListController().setMemberDao(memberDao));
```

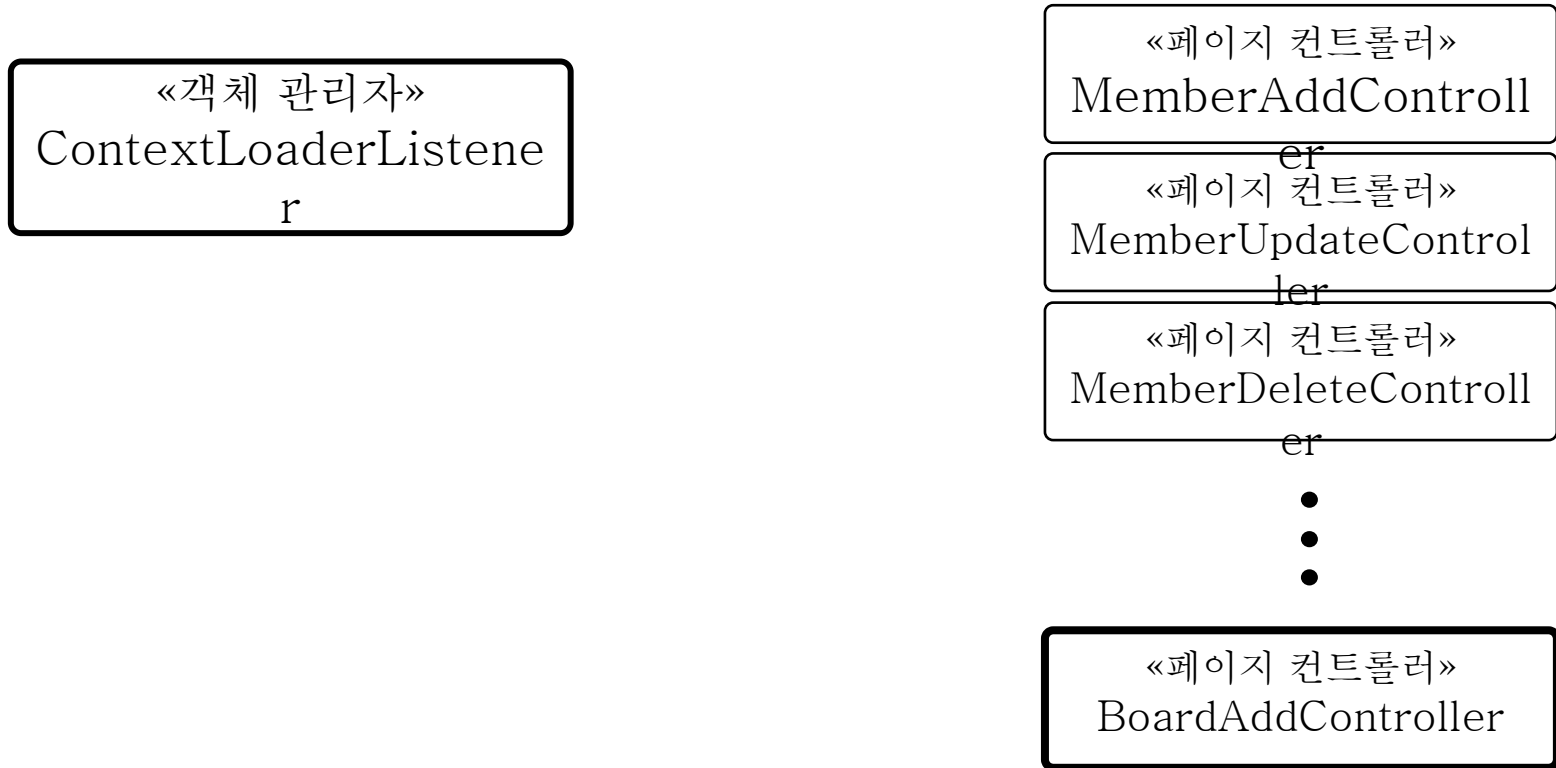
## 6.5 프로퍼티를 이용한 객체 관리

### 기존 방식의 문제점



## 6.5 프로퍼티를 이용한 객체 관리

새 페이지 컨트롤러가 추가되면,



## 6.5 프로퍼티를 이용한 객체 관리

ContextLoaderListener 클래스에  
해당 코드를 추가해야 함.

«객체 관리자»  
ContextLoaderListene  
r

```
...
sc.setAttribute("/auth/login.do", ...);
sc.setAttribute("/auth/logout.do", ...);
sc.setAttribute("/member/list.do", ...);
sc.setAttribute("/member/add.do", ...);
sc.setAttribute("/member/update.do", ...);
sc.setAttribute("/member/delete.do", ...);
MySqlBoardDao boardDao = new MySqlBoardDao();
boardDao.setDataSource(ds);
sc.setAttribute("/board/add.do",
    new BoardAddController().setBoardDao(boardDao));
...
```

«페이지 컨트롤러»  
MemberAddControll  
er

«페이지 컨트롤러»  
MemberUpdateControl  
ler

«페이지 컨트롤러»  
MemberDeleteControll  
er

•  
•  
•

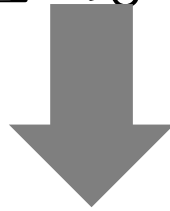
«페이지 컨트롤러»  
BoardAddController

## 6.5 프로퍼티를 이용한 객체 관리

DAO나 페이지 컨트롤러가  
추가되더라도  
ContextLoaderListener 클래스를  
변경하는 않는 방법?

## 6.5 프로퍼티를 이용한 객체 관리


DAO나 페이지 컨트롤러가  
추가되더라도  
ContextLoaderListener 클래스를  
변경하는 않는 방법?



“구조 변경”

## 6.5 프로퍼티를 이용한 객체 관리

# 준비해야 할 객체 정보를 외부 파일로 분리



```
application-  
context.properties
```

## 6.5 프로퍼티를 이용한 객체 관리

# 준비해야 할 객체 정보를 외부 파일로 분리



application-  
context.properties

```
jndi.dataSource=java:comp/env/jdbc/studydb  
memberDao=spms.dao.MySqlMemberDao  
/auth/login.do=spms.controls.LogInController  
/auth/logout.do=spms.controls.LogOutController  
/member/list.do=spms.controls.MemberListController  
/member/add.do=spms.controls.MemberAddController  
/member/update.do=spms.controls.MemberUpdateController  
/member/delete.do=spms.controls.MemberDeleteController
```

## 6.5 프로퍼티를 이용한 객체 관리

객체 정보는 key=value 형태로 작성.

memberDao=spms.dao.MySqlMemberDao

/WEB-INF/application-context.properties

jndi.dataSource=java:comp/env/jdbc/studydb

**memberDao=spms.dao.MySqlMemberDao**

/auth/login.do=spms.controls.LogInController

/auth/logout.do=spms.controls.LogOutController

/member/list.do=spms.controls.MemberListController

/member/add.do=spms.controls.MemberAddController

/member/update.do=spms.controls.MemberUpdateController

/member/delete.do=spms.controls.MemberDeleteController

## 6.5 프로퍼티를 이용한 객체 관리

객체 정보는 key=value 형태로 작성.

**memberDao**=spms.dao.MySqlMemberDao

보관소에 저장할 때 사용할

/WEB-INF/application-context.properties

jndi.dataSource=java:comp/env/jdbc/studydb  
memberDao=spms.dao.MySqlMemberDao  
/auth/login.do=spms.controls.LogInController  
/auth/logout.do=spms.controls.LogOutController  
/member/list.do=spms.controls.MemberListController  
/member/add.do=spms.controls.MemberAddController  
/member/update.do=spms.controls.MemberUpdateController  
/member/delete.do=spms.controls.MemberDeleteController



## 6.5 프로퍼티를 이용한 객체 관리

객체 정보는 key=value 형태로 작성.

memberDao=**spms.dao.MySqlMemberDao**

생성할 객체의 클래스 이름



/WEB-INF/application-context.properties

```
jndi.dataSource=java:comp/env/jdbc/studydb
memberDao=spms.dao.MySqlMemberDao
/auth/login.do=spms.controls.LogInController
/auth/logout.do=spms.controls.LogOutController
/member/list.do=spms.controls.MemberListController
/member/add.do=spms.controls.MemberAddController
/member/update.do=spms.controls.MemberUpdateController
/member/delete.do=spms.controls.MemberDeleteController
```

## 6.5 프로퍼티를 이용한 객체 관리

.properties 파일에 작성된 내용에 따라  
객체를 준비할 자동화 클래스 추가

application-  
context.properties



«객체 관리자»  
ApplicationContext

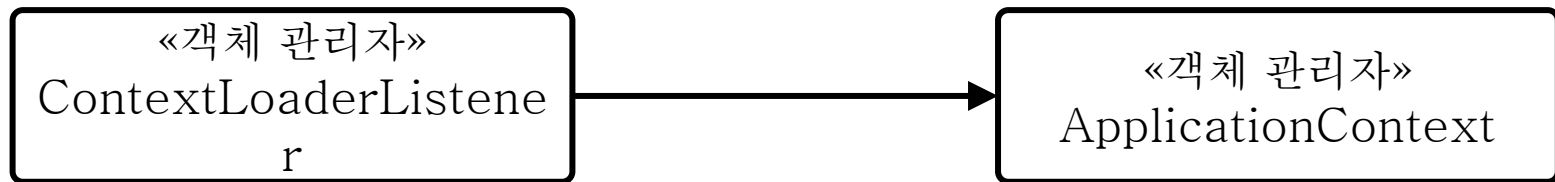


```
package spms.context;  
...  
public class ApplicationContext {  
  ...  
}
```

«페이지 컨트롤러»  
MemberListControll  
er

## 6.5 프로퍼티를 이용한 객체 관리

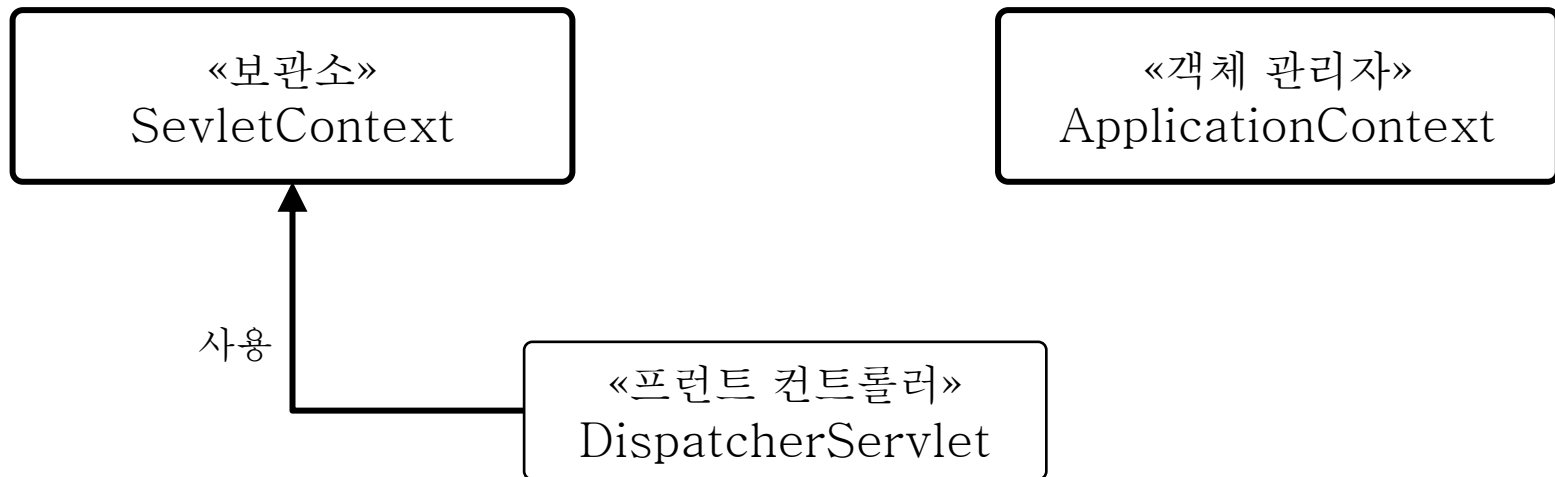
ContextLoaderListener는 객체 생성 및 관리를 ApplicationContext에게 위임.



```
public void contextInitialized(ServletContextEvent event) {  
    try {  
        ServletContext sc = event.getServletContext();  
  
        String propertiesPath = sc.getRealPath(  
            sc.getInitParameter("contextConfigLocation"));  
        applicationContext = new ApplicationContext(propertiesPath);  
  
    } catch (Throwable e) {  
        e.printStackTrace();  
    }  
}
```

## 6.5 프로퍼티를 이용한 객체 관리

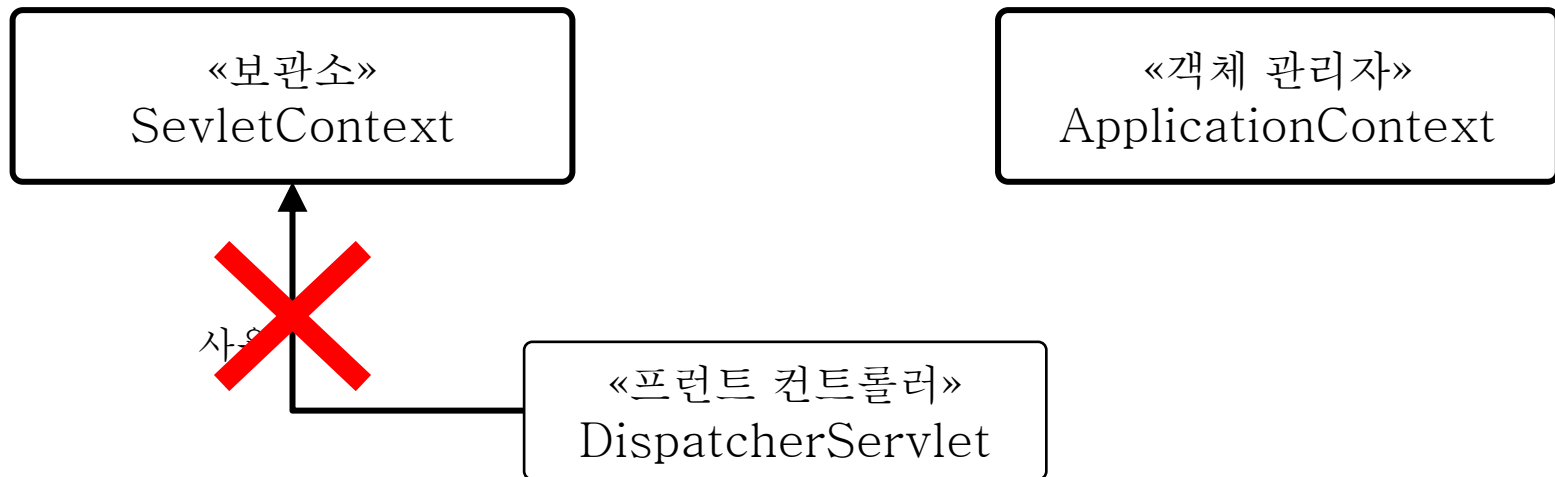
프런트 컨트롤러도 페이지 컨트롤러를 꺼낼 때 ApplicationContext를 사용.



```
ServletContext sc = this.getServletContext();
...
Controller pageController = (Controller) sc.getAttribute(servletPath);
if (pageController instanceof DataBinding) {
    prepareRequestData(request, model, (DataBinding)pageController);
}
```

## 6.5 프로퍼티를 이용한 객체 관리

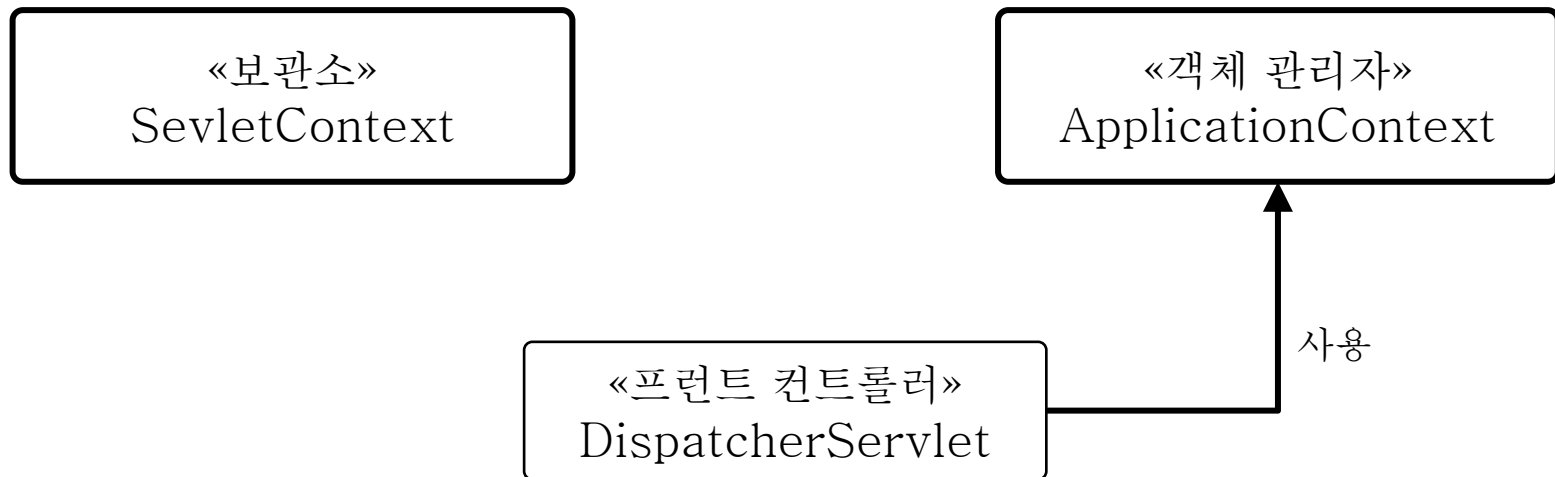
프런트 컨트롤러도 객체를 꺼낼 때  
ApplicationContext를 사용.



```
ServletContext sc = this.getServletContext();  
...  
Controller pageController = (Controller) sc.getAttribute(servletPath);  
if (pageController instanceof DataBinding) {  
    prepareRequestData(request, model, (DataBinding)pageController);  
}
```

## 6.5 프로퍼티를 이용한 객체 관리

프런트 컨트롤러도 객체를 꺼낼 때  
ApplicationContext를 사용.



```
ApplicationContext ctx = ContextLoaderListener.getApplicationContext();
...
Controller pageController = (Controller) ctx.getBean(servletPath);
if (pageController == null) {
    throw new Exception("요청한 서비스를 찾을 수 없습니다.");
}
```