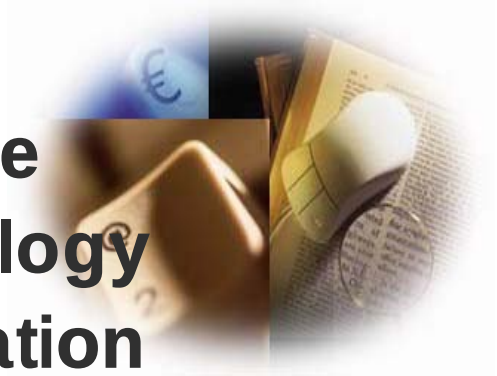
Korea
Software
Technology
Association

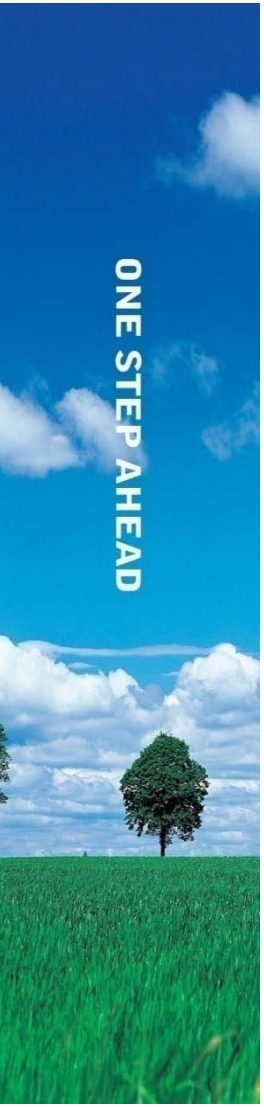


과정명 : 웹프레임워크의 활용
스트럿츠2 와 AJAX

강사명: 김 희 숙(넥스트리소프트 hskim@nextree.co.kr)

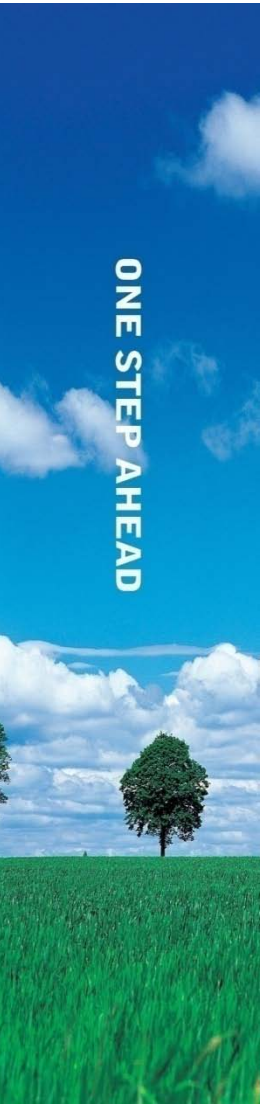
2.0.14 version으로 작성되었으며,
2.1.x version에서 사용하기 위해서는
별도의 Migration작업이 필요합니다.

4. 작업흐름에 Interceptor 추가



1. Interceptor 요청
2. Intercepts 행동
3. 스트럿츠2의 내장 Interceptor
4. Interceptor 선언
5. 사용자 Interceptor 작성

Interceptor 요청

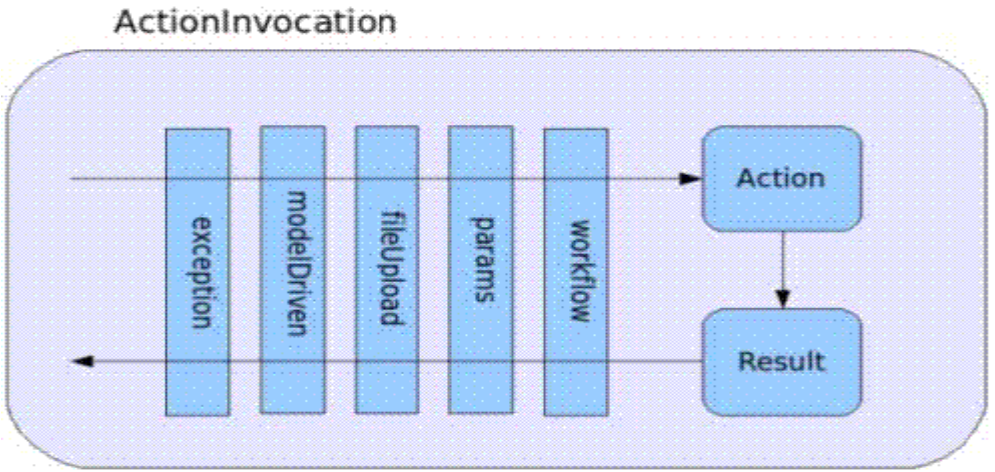


- MVC
- Interceptor 사용의 장점
- Interceptor 개발

Interceptor 요청

MVC

- Interceptor에 의해 Logging과 같은 횡단관심사(crosscutting concern), 데이터 전송과 같은 Preprocessing, Postprocessing을 Action으로부터 분리
- Action과 컨트롤러 사이에 Interceptor가 존재하며, Action은 독립적으로 호출되지 않음
- Action 호출은 execute() 메소드가 호출되기 전, 후로 많은 Interceptor가 항상 실행되는 계층화된 프로세스
- Action이 실행되기 전,후로 실행될 Interceptor 스택을 지닌 ActionInvocation을 생성



Interceptor 요청

❑ Interceptor 사용의 장점

- Layered 구조는 소프트웨어를 명확하게 하고, 가독성, 테스트용이성, 유연성까지 높임
- Layered 구조로 이루어진 Interceptor는 Logging이나 Validation등을 캡슐화하고 쉽게 재구성
- Interceptor를 통해 로직을 분리해내고 이를 다른 소프트웨어에서 재사용
- defaultStack을 사용함으로써 스트럿츠2 개발자들이 구현한 data transfer 코드, validation 코드, 국제화를 위한 코드 등을 재사용 가능

Interceptor 요청

❑ Interceptor 개발

- 스트럿츠2는 이미 웹 애플리케이션에서 일반적으로 사용되는 Interceptor를 내장
- Interceptor 들을 재구성(재배치,재배열) 하고 디버깅을 용이하게 하기 위해서는 스트럿츠2에 내장된 Interceptor들에 대해 잘 알고 있어야 함

Intercepts 행동

- **책임자 : ActionInvocation**
  - ActionInvocation은 요청에 대한 특정 Action의 전체 프로세스를 캡슐화
  - 요청이 들어오면 프레임워크는 해당 Action을 인스턴스화 하여 ActionInvocation에 추가
  - 애플리케이션의 xml이나 java annotation에 의해 생성된 관련 Interceptors를 찾아 ActionInvocation에 추가
  - ActionInvocation은 기타 다른 중요한 정보(servlet request와 같은)의 레퍼런스를 가짐

Intercepts 행동

❑ Interceptors 동작 방법(1)

- 프레임워크에서는 ActionInvocation의 invoke() 메소드를 호출하고 ActionInvocation은 Interceptor 스택의 첫번째 Interceptor의 interrupt()를 호출
- Interceptor는 일을 수행 후 다시 ActionInvocation의 invoke() 메소드를 호출함으로써 연속적인 Interceptor의 호출이 일어남
- Interceptor는 validation 체크같은 곳에서 에러가 발생하여 더 이상 진행 할 수없는 상황이 되면 ActionInvocation의 invoke()를 호출하지 않고 결과값을 리턴하고 작업을 멈춤

Intercepts 행동

❑ Interceptors 동작 방법(2)

TimerInterceptor의 intercept() 메소드 사용 Code

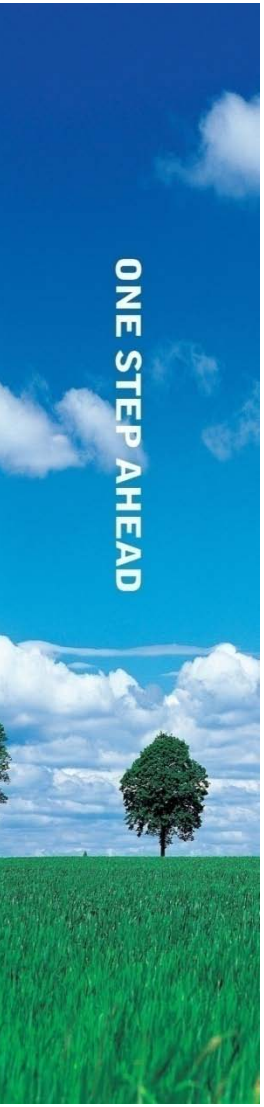
```
public String intercept(ActionInvocation invocation) throws Exception {

    long startTime = System.currentTimeMillis();
    String result = invocation.invoke();
    long executionTime = System.currentTimeMillis() - startTime;

    ... log the time ...
    return result;
}
```

```
(annotation) <#1 Do Some Pre-processing >
(annotation)<#2 Pass Control to the Rest of the Action Invocation Process>
(annotation) <#3 Do Some Post-processing >
```

스트럿츠2의 내장 Interceptor



- Utility Interceptor
- 데이터전송 Interceptor
- Workflow Interceptor
- 각종 Interceptor
- 내장 스택

스트럿츠2의 내장 Interceptor

❑ Utility Interceptor

- Timer
  - 실행시간을 기록
- Logger
  - 전처리와 후처리시에 단순한 logging 기능을 제공

Timer 실행결과

```
INFO: Executed action [/chapterFour/secure/ImageUpload!execute] took 123 ms.
```

Logger 실행결과

```
INFO: Starting execution stack for action/chapterFour/secure/ImageUpload
INFO: Finishing execution stack for action/chapterFour/secure/ImageUpload
```

스트럿츠2의 내장 Interceptor

❑ 데이터전송 Interceptor(1)

- Params(defaultStack)
  - request parameter를 ValueStack에 노출된 속성값을 전달
  - OGNL 에서 폼 필드명을 이용
- Static-params(defaultStack)
  - declarative architecture XML 파일에 선언된 정적인 파라미터 값을 ValueStack에 전달
  - params interceptors 호출 후 request에서 넘어온 파라미터 값을 덮어씀
  - Interceptor 호출 순서를 직접 변경하면 덮어 쓰는것을 방지함

Declarative architecture XML

```
<action name="exampleAction" class="example.ExampleAction">
    <param name="firstName">John</param>
    <param name="lastName">Doe</param>
</action>
```

스트럿츠2의 내장 Interceptor

❑ 데이터전송 Interceptor(2)

- Servlet-config(defaultStack)
  - Servlet API의 다양한 오브젝트들을 Action에서 사용 할 수 있게 함
  - org.apache.struts2.interceptor
- FileUploads(defaultStack)
  - multi-part 요청으로부터 파일과 관련된 메타 데이터를 일반적인 파라미터와 같이 Action에 셋팅될 수 있도록 변환

Servlet-config

ServletContextAware – sets the ServletContext
ServletRequestAware – sets the HttpServletRequest
ServletResponseAware – sets the HttpServletResponse
ParameterAware – sets a map of the request parameters
RequestAware – sets a map of the request attributes
SessionAware – sets a map of the session attributes
ApplicationAware – sets a map of application scope properties
PrincipalAware – sets the Principal object (security)

스트럿츠2의 내장 Interceptor

❑ Workflow Interceptors(1)

- Workflow(defaultStack)
  - Data validation과 검증에러 발생 후 Workflow를 제공

Interceptor 내부에서의 Workflow 변경

```
public String intercept(ActionInvocation invocation) throws Exception {

    Action action = invocation.getAction();
    if (action instanceof Validateable) {
        Validateable validateable = (Validateable) action;
        validateable.validate();
    }

    if (action instanceof ValidationAware) {
        ValidationAware validationAwareAction = (ValidationAware) action;
        if (validationAwareAction.hasErrors()) return Action.INPUT;
    }
     return invocation.invoke();
}
```

스트럿츠2의 내장 Interceptor

❑ Workflow Interceptors(2)

- Workflow(defaultStack)
  - Workflow Interceptor와 협업하기 위한 2가지 인터페이스
    - ➢ Validateable 인터페이스
    - ➢ ValidationAware 인터페이스
  - 3가지 설정 파라미터
    - ➢ alwaysInvokeValidate : true or false
      - validate() 호출여부
    - ➢ inputResultName
      - default : Action.INPUT
      - validation 실패시 result 이름을 반환
    - ➢ excludeMethods
      - workflow interceptor에 의해 실행되어서는 안되는 메소드명들

struts-default.xml

```
<interceptor-ref name="workflow">
   <param name="excludeMethods">input,back,cancel</param>
</interceptor-ref>
```

스트럿츠2의 내장 Interceptor

❑ Workflow Interceptors(3)

- Validation(defaultStack)
  - Validateable 인터페이스가 프로그램적으로 구현된 validation을 제공한다면 validation은 이미 구현되어 제공되는 validation을 위한 Interceptor
  - Workflow Interceptor는 이전에 동작
- Prepare(defaultStack)
  - Action에 임의의 workflow를 진행시키기 위한 진입점을 제공
  - Prepare Interceptor가 동작하면 Action의 prepare() 메소드를 찾아 실행
  - Preparable 인터페이스 구현은 선택
  - 설정 파라미터
    - alwaysInvokePrepare
      - Default : true
      - prepare() 호출할지 여부
  - 단일 액션에서 다른 실행 메서드를 가지는 경우, 실행메서드에 따라 prepare 메서드를 설정 가능(명명 규칙 **prepare**XXX(), **prepareD**oXXXX())

Action 메서드 명	Prepare 메서드 1	Prepare 메서드 2
input()	prepareInput()	prepareDoInput()
update()	prepareUpdate()	prepareDoUpdate()

스트럿츠2의 내장 Interceptor

❑ Workflow Interceptors(4)

- Model-driven(defaultStack)
  - ModelDriven 인터페이스를 구현한 액션 클래스의 getModel() 메서드를 호출하여 반환되는 객체의 프로퍼티를 ValueStack에 저장
  - ModelDriven 인터셉터는 static-params 인터셉터와 params 인터셉터 전에 위치해야함

```
<action name="RegistrationMD" class="example.actions.RegistrationMD">
    <interceptor-ref name="modelDriven"/>
    <interceptor-ref name="basicStack" />
    <result>/example/RegistrationMD.jsp</result>
</action>
```

스트럿츠2의 내장 Interceptor

❑ 각종 Interceptors(1)

- Exception(defaultStack)
  - defaultStack의 첫번째 등록되어 있으므로 액션과 인터셉터에서 발생한 모든 Exception을 담당
  - result로 제어를 넘길때 ValueStack의 최상단에 ExceptionHolder 객체를 배치

예외처리를 위한 XML 설정

```
<global-results>
        <result name="error">/common/error.jsp</result>
</global-results>

<global-exception-mappings>
        <exception-mapping exception="java.lang.Exception" result="error"/>
</global-exception-mappings>
```

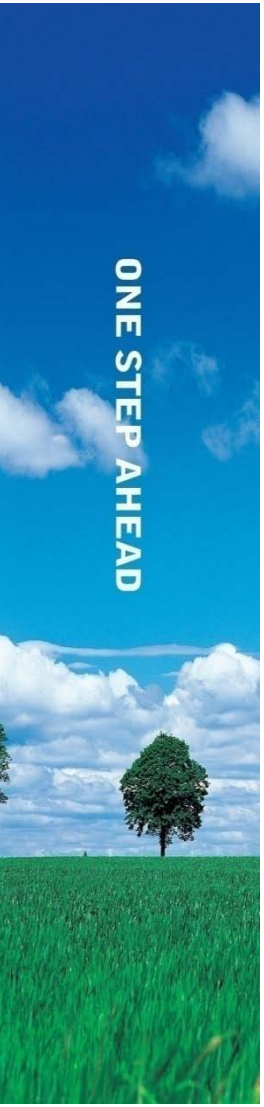
Error.jsp

```
<h4>Exception Name: </h4><s:property value="exception" /><br/>
<h4>What you did wrong:</h4> <s:property value="exceptionStack"/>
```

스트럿츠2의 내장 Interceptor

- **각종 Interceptors(2)**
  - Token 과 token-session
    - 사용자의 form submit상황에서 자주 일어나는 중복처리 방지
  - Scoped-Model-Driven(defaultStack)
    - model-driven interceptor에서 model을 HttpSession에 저장 하는 기능이 추가
  - ExecAndWait
    - 요청 처리가 길어지는 경우 사용자에게 피드백을 주기 위한 interceptor
- **내장 스택**
  - 스트럿츠2 프레임워크는 내장 Interceptor들로 이루어진 내장 스택을 제공
  - struts-default.xml에서 struts-default 패키지를 정의함으로써 다른 모든 내장 스택을 상속
  - defaultStack 사용 권장

Interceptor 선언



- Interceptor와 Interceptor 스택 선언
- Action과 Interceptor 맵핑
- 설정과 파라미터 오버라이딩

Interceptor 선언

❑ Interceptor와 Interceptor 스택 선언(1)

- Interceptor를 선언(key와 class를 매핑)
- 선언된 Interceptor를 개별적으로 액션 선언에서 사용가능
- Interceptor의 집합을 InterceptorStack으로 선언하여 사용가능

```
<package name="struts-default">
    ....
    <interceptors>
        <interceptor name="execAndWait" class="ExecuteAndWaitInterceptor"/>
        <interceptor name="exception" class="ExceptionMappingInterceptor"/>
        <interceptor name="fileUpload" class="FileUploadInterceptor"/>
        <interceptor name="i18n" class="I18nInterceptor"/>
        <interceptor name="logger" class="LoggingInterceptor"/>
        <interceptor name="model-driven" class="ModelDrivenInterceptor"/>
        <interceptor name="scoped-model-driven" class="ScopedModelDrivenInterceptor"/>
        <interceptor name="params" class="ParametersInterceptor"/>
        <interceptor name="prepare" class="PrepareInterceptor"/>
        ....
```

Interceptor 선언

❑ Interceptor와 Interceptor 스택 선언(2)

```
....
<interceptor-stack name="defaultStack">
    <interceptor-ref name="exception"/>
    <interceptor-ref name="alias"/>
    <interceptor-ref name="servlet-config"/>
    <interceptor-ref name="prepare"/>
    <interceptor-ref name="i18n"/>
    <interceptor-ref name="chain"/>
    <interceptor-ref name="debugging"/>
    <interceptor-ref name="profiling"/>
    <interceptor-ref name="scoped-model-driven"/>
    <interceptor-ref name="model-driven"/>
    <interceptor-ref name="fileUpload"/>
    <interceptor-ref name="checkbox"/>
    <interceptor-ref name="static-params"/>
```

Interceptor 선언

❑ Interceptor와 Interceptor 스택 선언(3)

```
            <interceptor-ref name="params">
                <param name="excludeParams">dojo₩..*</param>
            </interceptor-ref>
            <interceptor-ref name="conversionError"/>
            <interceptor-ref name="validation">
                <param  name="excludeMethods">input,back,cancel,browse</param>
            </interceptor-ref>
            <interceptor-ref name="workflow">
                <param  name="excludeMethods">input,back,cancel,browse</param>
            </interceptor-ref>
        </interceptor-stack>
    </interceptors>

    <default-interceptor-ref name="defaultStack"/>

</package>
```

Interceptor 선언

❑ XML Document 구조(http://struts.apache.org/dtds/struts-2.0.dtd)

- struts.xml 설정시 XML 엘리먼트들의 순서에 주의

struts.xml 의 dtd 선언 참조

```
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
        "http://struts.apache.org/dtds/struts-2.0.dtd">
```

struts-2.0.dtd

```
<!ELEMENT struts (package|include|bean|constant)*>
<!ELEMENT package (result-types?, interceptors?, default-interceptor-ref?, default-action-ref?,
        global-results?, global-exception-mappings?, action*)>

<!ATTLIST package
    name CDATA #REQUIRED
    extends CDATA #IMPLIED
    namespace CDATA #IMPLIED
    abstract CDATA #IMPLIED
    externalReferenceResolver NMTOKEN #IMPLIED>
<!ELEMENT result-types (result-type+)>
.....
```

Interceptor 선언

❑ Action과 Interceptor 맵핑

- Action에 interceptor를 선언한 경우, 선언된 interceptor만 동작
- Interceptor를 선언하지 않으면 default-interceptor-ref 에 정의된 interceptor stack 사용
- package가 struts-default를 상속하였으나, default-interceptor-ref를 선언하지 않은 경우는 defaultStack을 사용

```
<action name="MyAction" class="org.actions.myactions.MyAction">
        <interceptor-ref name="timer"/>
        <interceptor-ref name="logger"/>
        <result>Success.jsp</result>
</action>
```

action에 2개의 Interceptor가 연관되어 있고 순서대로 동작. params interceptor가 없기 때문에 어떤 request data에도 접근 할 수 없음.

```
<action name="MyAction" class="org.actions.myactions.MyAction">
        <interceptor-ref name="timer"/>
        <interceptor-ref name="logger"/>
        <interceptor-ref name="defaultStack"/>
        <result>Success.jsp</result>
</action>
```

timer, logger Interceptor를 수행하고 defaultStack에 정의된 Interceptor들을 수행

Interceptor 선언

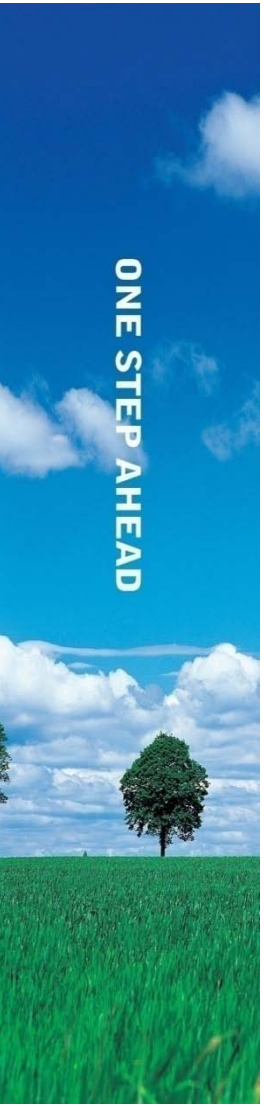
❑ 설정과 파라미터 오버라이딩

- workflow interceptor는 defaultStack의 요청을 무시하고 excludeMethods의 정의된데로 변경
- interceptor-ref 가 생성될때 파라미터를 전달
- defaultStack을 참조 할때 이미 excludeMethods 파라미터 값으로 변경

```
<interceptor-ref name="workflow">
    <param name="excludeMethods">input,back,cancel,browse</param>
</interceptor-ref>
```

```
<action name="YourAction" class="org.actions.youractions.YourAction">
    <interceptor-ref name="defaultStack">
        <param name="workflow.excludeMethods">doSomething</param>
    </interceptor-ref>
    <result>Success.jsp</result>
</action>
```

사용자 Interceptor 작성



- Interceptor 인터페이스 구현
- AuthenticationInterceptor 작성

사용자 Interceptor 작성

❑ Interceptor 인터페이스 구현

- Interceptor 구현 인터페이스
  - com.opensymphony.xwork2.interceptor.Interceptor
  - destory(), init() 메소드는 생명주기에 따른 자원을 회수 하거나 초기화 시킴
  - 실제 비즈니스는 intercept 메소드에서 실행
  - intercept 메소드는 ActionInvocation의 invoke() 메소드에서 재귀적으로 호출

```
public interface Interceptor extends Serializable {
    void destroy();
    void init();
    String intercept(ActionInvocation invocation) throws Exception;
}
```

사용자 Interceptor 작성

❑ AuthenticationInterceptor 작성(1)

- 보안을 요하는 행위는 허가된 사용자의 요구인지 확인
  - 허가된 사용자일 경우 : ActionInvocation의 invoke() 메소드 호출
    - ➢ interceptor stack에 정의된 순서대로 다음 interceptor를 호출하기 위해 제어를 ActionInvocation으로 이전
  - 허가된 사용자가 아닐 경우 : 더 이상 실행 되지 않음
    - ➢ 제어 문자열(Action.INPUT)을 반환하여 workflow를 변경

사용자 Interceptor 작성

❑ AuthenticationInterceptor 작성(2)

LoginAction.java

```
public class LoginAction extends ActionSupport implements SessionAware {

    public String execute(){
        User user = getUserService().login(userId, password);
        if ( user == null ) {
            return INPUT;
        }else{
            session.put(NexschoolConstants.USERID, user);
        }
        return SUCCESS;
    }

    . . .
    private Map session;
    public void setSession(Map session) {
        this.session = session;
    }
}
```

사용자 Interceptor 작성

❑ AuthenticationInterceptor 작성(3)

AuthenticationInterceptor.java

```
public class AuthenticationInterceptor implements Interceptor {
    ....
    public String intercept( ActionInvocation actionInvocation ) throws Exception{
        Map session = actionInvocation.getInvocationContext().getSession();
        User user = (User) session.get(NexschoolConstants.USER);
        if (user == null) {
           return Action.LOGIN;
        }

        return actionInvocation.invoke();

    }
}
```