Spring - examples

Web.xml (Spring configuration)

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      /WEB-INF/context/applicationContext.xml
      /WEB-INF/context/applicationContext-data.xml
      /WEB-INF/context/applicationContext-beans.xml
      /WEB-INF/context/applicationContext-security.xml
    </param-value>
  </context-param>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
```

Web.xml (enabling web scopes)

```
<listener>
    <listener-class>
        org.springframework.web.context.request.RequestContextListener
    </listener-class>
</listener>
```

Web.xml (log4j configuration)

```
<listener>
	<listener-class>
		org.springframework.web.util.Log4jConfigListener
	</listener-class>
</listener>
```

Spring annotations

- *@Component*
- *@Repository*
- *@Service*

Very similar : Life cycle of such class instances controlled by the Spring container.

Spring annotations

- *@Configurable*

Life cycle of such class instances **is not** controlled by the Spring container.

Still, Spring manages dependency injection in such classes.

As Spring does not create an instance of such a class, it must be notified that the instance has been created/ destroyed => code instrumentation necessary:

  - Class loader agent (load-time weaving)
  - Aspect programming (aspect compiler)

JSF validator

```
@Component
@Scope("request")
@FacesValidator("UsernameValidator")
public class UsernameValidator implements Validator {

        @Autowired
        UserService userService;

        @Override
        public void validate(FacesContext context, UIComponent component,
                Object value) throws ValidatorException {

          String username = value.toString();

          if(userService.isUsernameDuplicated(username)){
                FacesMessage facesMsg = new FacesMessage("Username is duplicated");
                facesMsg.setSeverity(FacesMessage.SEVERITY_ERROR);
                throw new ValidatorException(facesMsg);
          }

        }

}
```

http://www.mkyong.com/jsf2/spring-autowired-into-jsf-custom-validator/

JSF validator

import org.springframework.stereotype.Component;

import org.springframework.context.annotation.Scope;

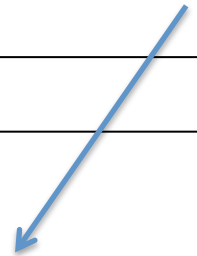
import javax.faces.validator.FacesValidator;

```
@Component
@Scope("request")
@FacesValidator("[illegible]")
public class UsernameValidator implements Validator {

        @Autowired
        UserService userService;

        @Override
        public void validate(FacesContext context, UIComponent component,
                Object value) throws ValidatorException {

          String username = value.toString();

          if(userService.isUsernameDuplicated(username)){
                FacesMessage facesMsg = new FacesMessage("Username is duplicated");
                facesMsg.setSeverity(FacesMessage.SEVERITY_ERROR);
                throw new ValidatorException(facesMsg);
          }

        }

}
```

http://www.mkyong.com/jsf2/spring-autowired-into-jsf-custom-validator/

JSF validator

```
<h:inputText id="username">
      <f:validator validatorId="UsernameValidator" />
</h:inputText>
```



```
@Component
@Scope("request")
@FacesValidator("UsernameValidator")
public class UsernameValidator implements Validator {
```

Problem:

- NullPointer exception thrown when accessing the validator. Why?

Spring creates an instance of UsernameValidator and injects to userService a (request-scoped) instance of UserService.

- JSF create their own instance of UsernameValidator, that has userService null (non-injected) => NullPointerException

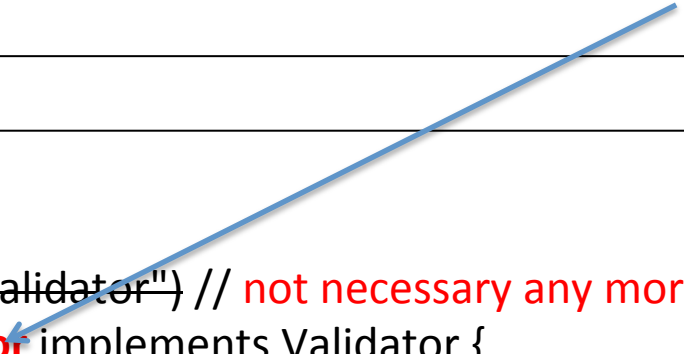
http://www.mkyong.com/jsf2/spring-autowired-into-jsf-custom-validator/

JSF validator

Solution:
JSF and Spring resolvers not collaborating correctly here, let us use Spring resolver only.

```
<h:inputText id="username">
      <f:validator validatorId="UsernameValidator" binding="#{usernameValidator}"/>
</h:inputText>
```

```
@Component
@Scope("request")
@FacesValidator("UsernameValidator") // not necessary any more
public class UsernameValidator implements Validator {
```



http://www.mkyong.com/jsf2/spring-autowired-into-jsf-custom-validator/