weblogic 中 hibernate Validator 解决 jar 冲突

在tomcat环境中hibernate Validator会默认在classpath中找PersistenceProvider的实现（eclipse.persistence.PersistenceProviderImpl）；

而在weblogic中亦然，但实现找的确是（javax.persistence.spi.PersistenceProvider） 所以会发生转换异常.

解决办法:

自定义 TraversableResolver, 基于@Configuration @Bean 配置 Validator

代码如下:

```
import java.lang.annotation.ElementType;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.validation.ConstraintViolation;
import javax.validation.ConstraintViolationException;
import javax.validation.Path;
import javax.validation.Path.Node;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import javax.validation.TraversableResolver;
import javax.validation.Validation;
import javax.validation.Validator;
import javax.validation.ValidatorFactory;

import com.google.common.collect.Lists;
import com.google.common.collect.Maps;


@Configuration
public class BeanValidators implements TraversableResolver{

@Override
public boolean isReachable(Object traversableObject, Node traversableProperty, Class<?> rootBeanType,
        Path pathToTraversableObject, ElementType elementType) {
    return true;
}

@Override
public boolean isCascadable(Object traversableObject, Node traversableProperty, Class<?> rotBeanType,
        Path pathToTraversableObject, ElementType elementType) {
    return true;
}

/**
 * 配置 Validator
 * @return
 */
@Bean
public Validator validator(){
    ValidatorFactory factory = Validation.byDefaultProvider()
            .configure().traversableResolver(this)
            .buildValidatorFactory();
            return  factory.getValidator();
}



/**
 * 调用JSR303的validate方法, 验证失败时抛出ConstraintViolationException.
 */
@SuppressWarnings({ "unchecked", "rawtypes" })
public static void validateWithException(Validator validator, Object object, Class<?>... groups)
        throws ConstraintViolationException {
    Set constraintViolations = validator.validate(object, groups);
    if (!constraintViolations.isEmpty()) {
        throw new ConstraintViolationException(constraintViolations);
    }
}

/**
 * 辅助方法, 转换ConstraintViolationException中的Set<ConstraintViolations>中为List<messag
>.
 */
public static List<String> extractMessage(ConstraintViolationException e) {
    return extractMessage(e.getConstraintViolations());
}

/**
 * 辅助方法, 转换Set<ConstraintViolation>为List<message>
 */
@SuppressWarnings("rawtypes")
public static List<String> extractMessage(Set<? extends ConstraintViolation> constraintViolat
ons) {
    List<String> errorMessages = Lists.newArrayList();
    for (ConstraintViolation violation : constraintViolations) {
        errorMessages.add(violation.getMessage());
    }
    return errorMessages;
}

/**
 * 辅助方法, 转换ConstraintViolationException中的Set<ConstraintViolations>为Map<property,
message>.
 */
public static Map<String, String> extractPropertyAndMessage(ConstraintViolationException e
 {
    return extractPropertyAndMessage(e.getConstraintViolations());
}
/**
 * 辅助方法, 转换Set<ConstraintViolation>为Map<property, message>.
 */
@SuppressWarnings("rawtypes")
public static Map<String, String> extractPropertyAndMessage(Set<? extends ConstraintViolat
on> constraintViolations) {
    Map<String, String> errorMessages = Maps.newHashMap();
    for (ConstraintViolation violation : constraintViolations) {
        errorMessages.put(violation.getPropertyPath().toString(), violation.getMessage());
    }
    return errorMessages;
}

/**
 * 辅助方法, 转换ConstraintViolationException中的Set<ConstraintViolations>为List<propertyPa
h message>.
 */
public static List<String> extractPropertyAndMessageAsList(ConstraintViolationException e) {
    return extractPropertyAndMessageAsList(e.getConstraintViolations(), " ");
}

/**
 * 辅助方法, 转换Set<ConstraintViolations>为List<propertyPath message>.
 */
@SuppressWarnings("rawtypes")
public static List<String> extractPropertyAndMessageAsList(Set<? extends ConstraintViolatio
> constraintViolations) {
    return extractPropertyAndMessageAsList(constraintViolations, " ");
}

/**
 * 辅助方法, 转换ConstraintViolationException中的Set<ConstraintViolations>为List<propertyPa
h +separator+ message>.
 */
public static List<String> extractPropertyAndMessageAsList(ConstraintViolationException e, S
ring separator) {
    return extractPropertyAndMessageAsList(e.getConstraintViolations(), separator);
}

/**
 * 辅助方法, 转换Set<ConstraintViolation>为List<propertyPath +separator+ message>.
 */
@SuppressWarnings("rawtypes")
public static List<String> extractPropertyAndMessageAsList(Set<? extends ConstraintViolatio
> constraintViolations,
        String separator) {
    List<String> errorMessages = Lists.newArrayList();
    for (ConstraintViolation violation : constraintViolations) {
        errorMessages.add(violation.getPropertyPath() + separator + violation.getMessage());
    }
    return errorMessages;
}

}
```

原文链接：weblogic 中 hibernate Validator 解决 jar 冲突