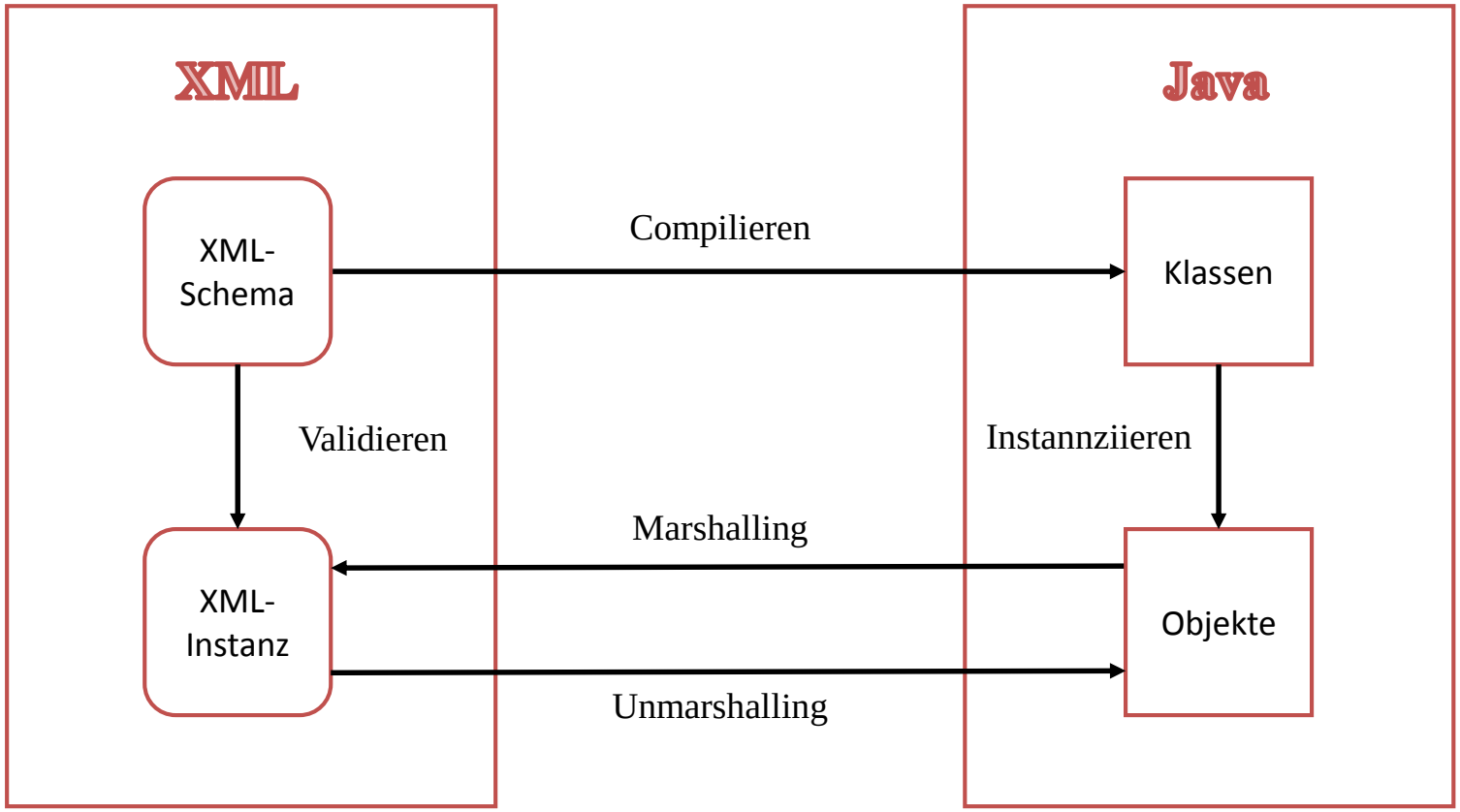
XML

SS 2019

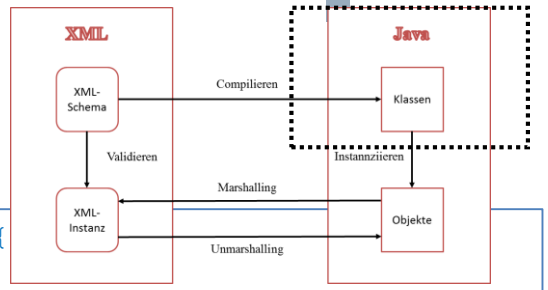
JAXB

- Java Architecture for XML Binding
  - „Binding“: bestimmte Eigenschaften der Java-Klassen werden an bestimmte Elemente oder Attribute im XML gebunden
- Realisierung über Annotationen:
  - Definition, welches Attribut einer Klasse auf welches Element bzw. Attribut im XML-Dokument abgebildet werden soll und umgekehrt
  - Java -> XML
    - automatische Umwandlung von Objektinstanzen mit Attributen in das entsprechende XML: Marshalling
  - XML -> Java
    - Es werden Objektinstanzen erzeugt und die Attribute der Java-Objekte werden mit den Werten in den Attributen und Elementen aus dem XML belegt: Unmarshalling

Java Architecture for XML Binding



Java - Beispiel*

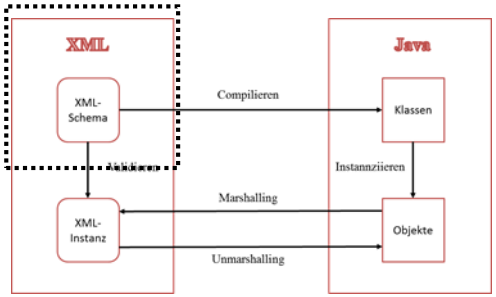



```
public class Terminkalender {

    private Besitzer besitzer;
    private Termine termine;

    public Besitzer getBesitzer() {
        return besitzer;
    }

    public void setBesitzer(Besitzer besitzer) {
        this.besitzer = besitzer;
    }

        public Termine getTermine() {
                return termine;
        }

        public void setTermine(Termine termine) {
                this.termine = termine;
        }
}
```

```
public class Termine {

    private String name;
    private List<Termin> termine;

    public String getName() {
                return name;
    }

    public void setName(String name) {
                this.name = name;
    }

    public List<Termin> getTermine() {
        if(termine == null) {
                termine = new ArrayList<>();
        }
        return termine;
    }

    public void setTermine(List<Termin> termine){
                this.termine = termine;
    }
}
```

* Dieses und folgende Beispiele aus [PA]

XSD - Venetian Blind Design




```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tk="http://meinnamespace.de"
  targetNamespace="http://meinnamespace.de"
  elementFormDefault="qualified">
  <element name="terminkalender">
    <complexType name="terminkalender">
      <sequence>
        <element name="besitzer" type="tk:besitzer" />
        <element name="termine" type="tk:termine" />
      </sequence>
    </complexType>
  </element>
  <complexType name="besitzer">
    <sequence>
      <element name="name" type="string" />
    </sequence>
  </complexType>
  <complexType name="termine">
    <sequence>
      <element name="termin" type="tk:termin" maxOccurs="unbounded" />
    </sequence>
    <attribute name="name" type="string" />
  </complexType>
. . . . .
</schema>
```

Attribut xmlns: Namensraum für XSD für Terminkalender

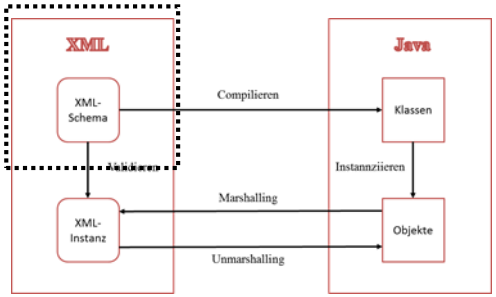
Namensraum für das XML, das mit der XSD beschrieben wird, inkl.Namensraum-Präfix tk

Wurzelelement Terminkalender

Elementdefinitionen

Typdefinitionen über den Namen mit dem sie von der jeweiligen Elementdefinition referenziert werden.

XSD

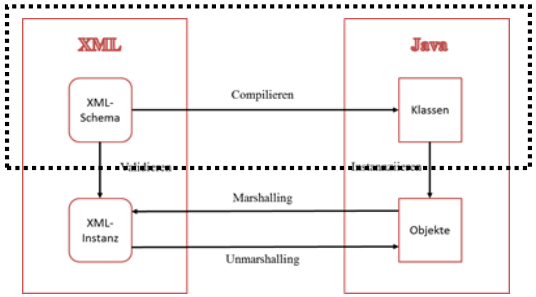



```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tk="http://meinnamespace.de"
  targetNamespace="http://meinnamespace.de"
  elementFormDefault="qualified">
  <element name="terminkalender">
    <complexType name="terminkalender">
    . . .
    </complexType>
  </element>
  <complexType name="besitzer">
  . . .
  </complexType>
  <complexType name="termine">
  . . .
  </complexType>
  <complexType name="termin">
    <sequence>
      <element name="was" type="string" />
      <element name="wann" type="string" />
      <element name="wo" type="string" />
    </sequence>
  </complexType>
</schema>
```

XML-Namensraum

- Namensräume in XML sind vergleichbar mit Packages in Java.
- Packages verwendet man,
  - damit Klassen eindeutig identifiziert werden und
  - Konflikte mit gleichnamigen Klassen vermieden werden.
- So auch in XML:
  - Über den Namensraum werden die Elemente unterhalb des Elements, welches das xmlns-Attribut hat (und das Element selbst) eindeutig identifiziert.
  - Eventuelle Namenskonflikte mit anderen Elementen (die zum Beispiel aus anderen XML-Dokumenten importiert werden können) werden so vermieden.

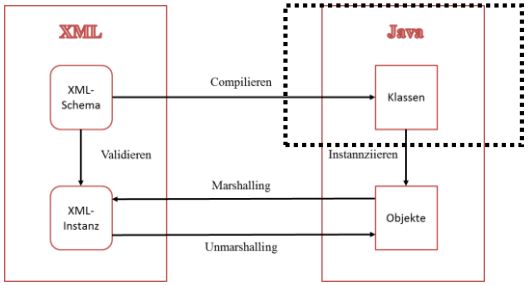
XSD <-> Java



- XSD -> Java
  - Aus der XSD kann man die Java-Klassen inklusive Annotationen auch generieren.
  - xjc -d src -p de.<meinpackagename> terminkalender.xsd
  - Beim Java-JDK wird das Programm xjc.exe mitgeliefert (bin-Verzeichnis)

- XSD <- Java
  - Aus Java-Klassen kann man Schema-XSD-Dateien generieren
  - Beim Java-JDK wird das Programm schemagen.exe mitgeliefert
  - Anwendung oft schwierig, da schemagen.exe Exceptions wirft, ohne die Ursache zu verdeutlichen.

Annotation der Java-Klassen




```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "besitzer",
    "termine"
})
@XmlRootElement(name = "terminkalender")
public class Terminkalender {
    @XmlElement(required = true)
    private Besitzer besitzer;
    @XmlElement(required = true)
    private Termine termine;

    public Besitzer getBesitzer() {
        return besitzer;
    }
    public void setBesitzer(Besitzer value) {
        this.besitzer = value;
    }
          public Termine getTermine() {
                    return termine;
          }
          public void setTermine(Termine termine) {
                    this.termine = termine;
          }
}
```

XML-Typ dieser Klasse, Reihenfolge der XML-Eigenschaften der Klasse

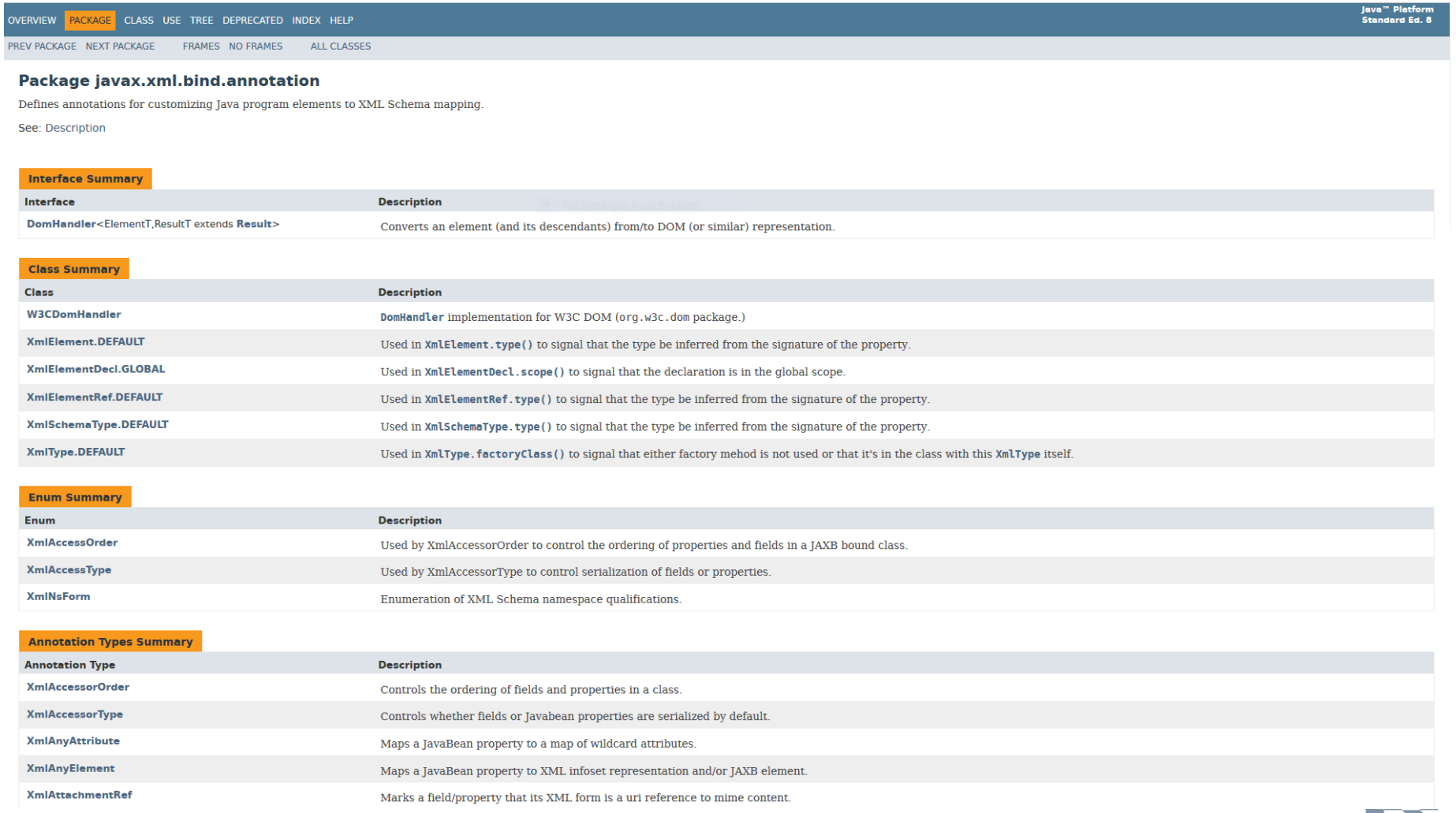
@XMLElement kann man sowohl an Gettern als auch direkt an Objekt-variablen verwenden.
Mit @XmlAccessorType legt man fest, welche Variante genommen werden soll, XmlAccessType.PROPERTY für Getter und XmlAccessType.FIELD für Objektvariablen.

@XmlRootElement heißt, dass diese Klasse die Rolle des Wurzelelements im XML-Baum hat

@XmlElement definiert, welche Attribute auf ein XML-Element abgebildet werden.

Annotationen

- http://docs.oracle.com/javase/8/docs/api/javax/xml/bind/annotation/package-summary.html



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP — Java™ Platform Standard Ed. 8

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

Package javax.xml.bind.annotation

Defines annotations for customizing Java program elements to XML Schema mapping.

See: Description

Interface Summary

Interface	Description
DomHandler<ElementT,ResultT extends **Result**>	Converts an element (and its descendants) from/to DOM (or similar) representation.

Class Summary

Class	Description
W3CDomHandler	**DomHandler** implementation for W3C DOM (org.w3c.dom package.)
XmlElement.DEFAULT	Used in **XmlElement.type()** to signal that the type be inferred from the signature of the property.
XmlElementDecl.GLOBAL	Used in **XmlElementDecl.scope()** to signal that the declaration is in the global scope.
XmlElementRef.DEFAULT	Used in **XmlElementRef.type()** to signal that the type be inferred from the signature of the property.
XmlSchemaType.DEFAULT	Used in **XmlSchemaType.type()** to signal that the type be inferred from the signature of the property.
XmlType.DEFAULT	Used in **XmlType.factoryClass()** to signal that either factory mehod is not used or that it's in the class with this **XmlType** itself.

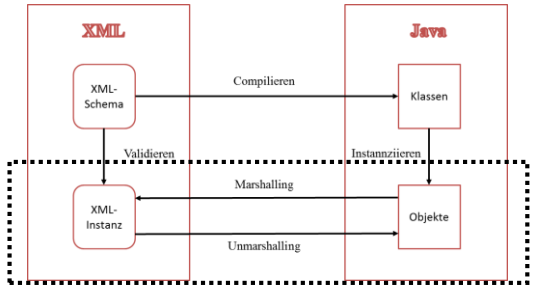
Enum Summary

Enum	Description
XmlAccessOrder	Used by XmlAccessorOrder to control the ordering of properties and fields in a JAXB bound class.
XmlAccessType	Used by XmlAccessorType to control serialization of fields or properties.
XmlNsForm	Enumeration of XML Schema namespace qualifications.

Annotation Types Summary

Annotation Type	Description
XmlAccessorOrder	Controls the ordering of fields and properties in a class.
XmlAccessorType	Controls whether fields or Javabean properties are serialized by default.
XmlAnyAttribute	Maps a JavaBean property to a map of wildcard attributes.
XmlAnyElement	Maps a JavaBean property to XML infoset representation and/or JAXB element.
XmlAttachmentRef	Marks a field/property that its XML form is a uri reference to mime content.

Java -> XML (Instanziieren + Marshalling)




```
Terminkalender terminkalender = new Terminkalender();
Besitzer besitzer = new Besitzer();
besitzer.setName(„ich");
Termine termine = new Termine();
termine.setName("Neue Termine");
Termin termin = new Termin();
termin.setWann(„8:15");
termin.setWas(„Java");
termin.setWo(„Labor");
termine.getTermine().add(termin);
terminkalender.setTermine(termine);
terminkalender.setBesitzer(besitzer);

JAXBContext jaxbContext = JAXBContext.newInstance(Terminkalender.class,
  Termine.class, Termin.class, Besitzer.class);

Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
jaxbMarshaller.marshal(terminkalender, ausgabe);
```

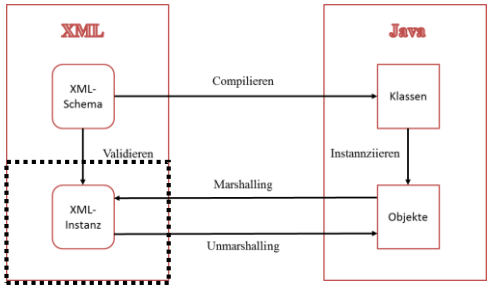
JAXBContext , über den die Klassen bekannt gemacht werden

die Klassen werden vom Marshaller verwendet, um aus den Java-Objekten das XML-Dokument zu generieren

-generiertes XML schön formatiert

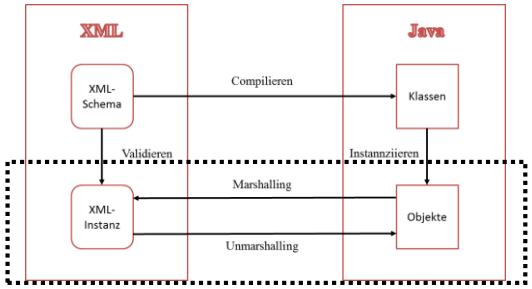
- das XML soll in ausgabe gespeichert werden

XML - Instanz




```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<terminkalender xmlns="http://meinnamespace.de">
    <besitzer>
        <name>ich</name>
    </besitzer>
    <termine name="Neue Termine">
        <termine>
            <was>Java</was>
            <wann>8:15</wann>
            <wo>Labor</wo>
        </termine>
    </termine>
</terminkalender>
```

XML -> Java (Unmarshalling)



die Klassen, auf die das XML gemappt werden soll, dem Kontext JAXBContext bekanntmachen.

```
JAXBContext jaxbContext = JAXBContext.newInstance(Terminkalender.class,
  Termine.class, Termin.class, Besitzer.class);
Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
Terminkalender terminkalender =
(Terminkalender) jaxbUnmarshaller.unmarshal(eingabe);
System.out.println(terminkalender.getBesitzer().getName());
```

Ein Unmarshaller für das Umwandeln von XML zu Java-Objekten

eingabe kann vom Typ File, InputStream, Reader u.a. sein, aus dem das XML gelesen werden soll

Literatur

- Christian Ullenboom: Java ist auch eine Insel
- http://docs.oracle.com/javase/8/docs/api/javax/xml/bind/annotation/package-summary.html
- http://www.torsten-horn.de/techdocs/java-xml-jaxb.htm#JAXB
- Philipp Ackermann: Schrödinger programmiert Java