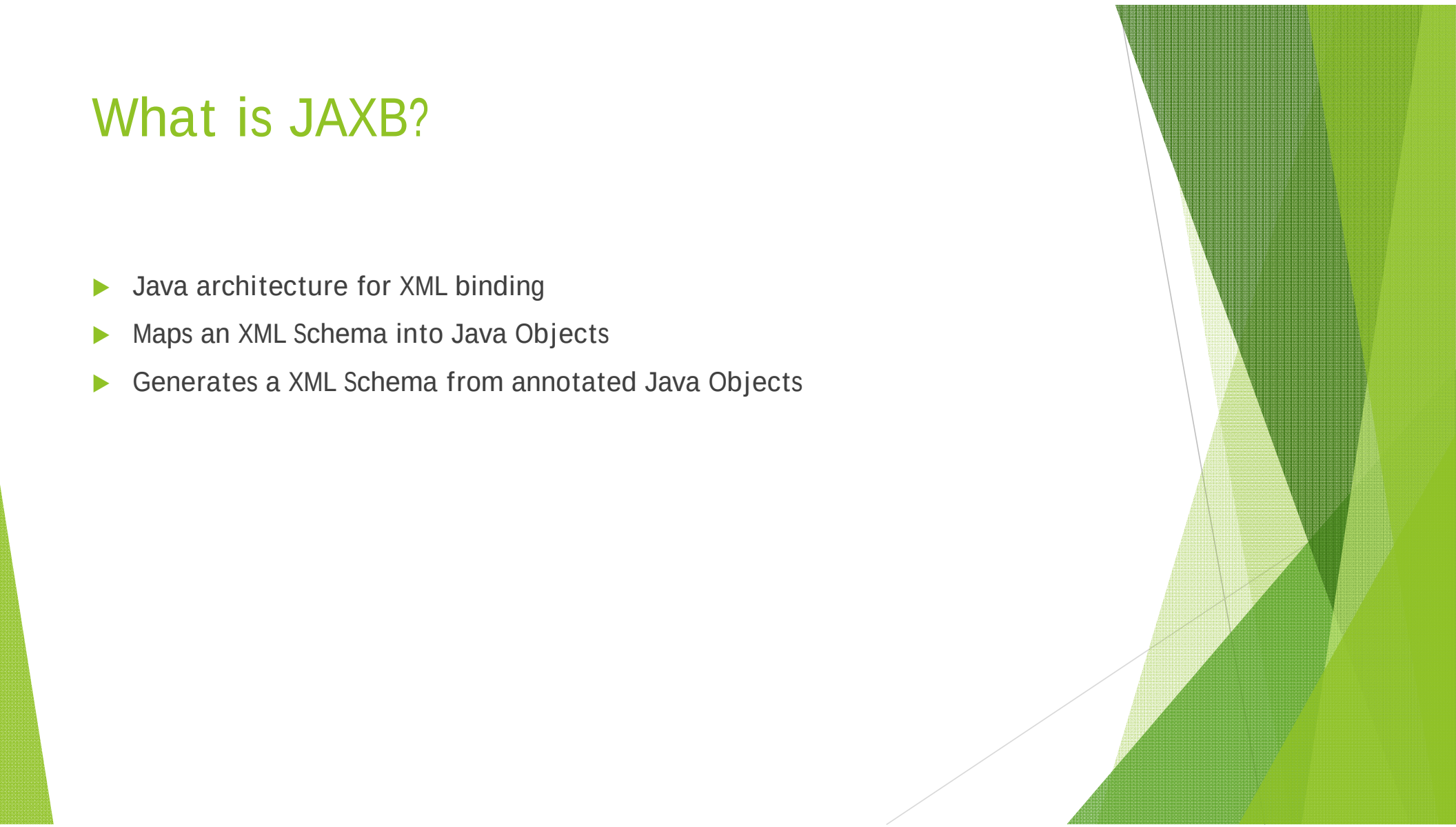
JAXB Tips and Tricks

By

Rob Ratcliff

What is JAXB?

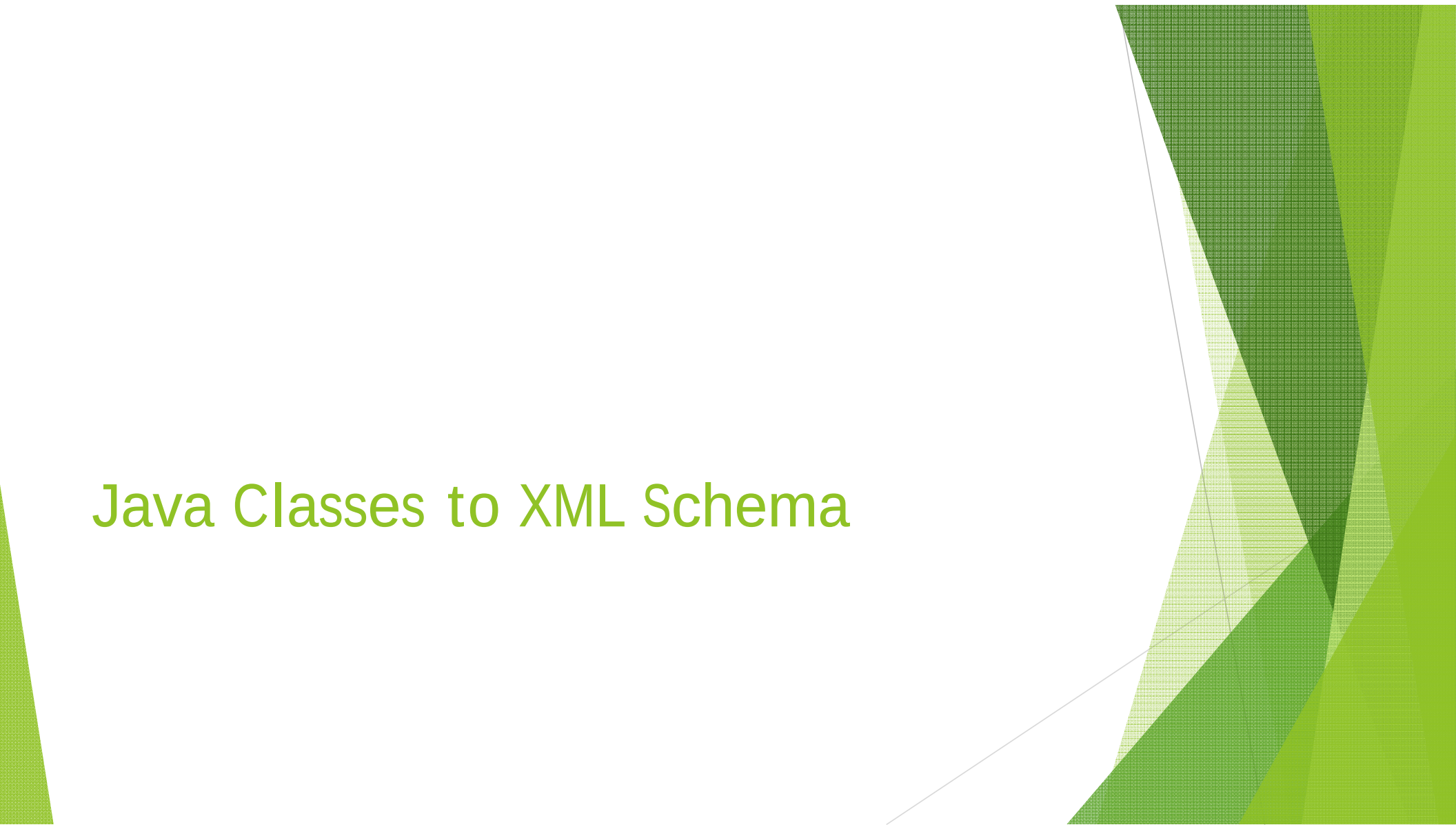
- Java architecture for XML binding
- Maps an XML Schema into Java Objects
- Generates a XML Schema from annotated Java Objects



Top 10 Reasons JAXB is a Fav!

1. Built into the JDK
2. Easy to use
3. Well documented
4. Logical API
5. Robust and Efficient implementation
6. Complete control of the fields that get serialized to XML
7. Handles mapping from XML Schema or Annotated Java classes
8. Community generated plugins to customize code generation
9. Good default mappings
10. Default mappings can be completely overridden

Java Classes to XML Schema



Converting Java to XML

```
JAXBContext context = JAXBContext.newInstance(new
Class[]{Shape.class, Circle.class, Rectangle.class
        Polygon.class, ShapeList.class});


StringWriter stringWriter = new StringWriter();
Marshaller marshaller = context.createMarshaller();
marshaller.setProperty("jaxb.formatted.output",
Boolean.TRUE);
marshaller.marshal(jaxbElement, stringWriter);
teturn stringWriter.toString();
```

Converting XML to Java

```
StringReader reader = new StringReader(xmlString);
JAXBContext context = getJaxBContext();
ShapeList shape = (ShapeList) context.createUnmarshaller().unmarshal(reader);
```

Controlling Serialization

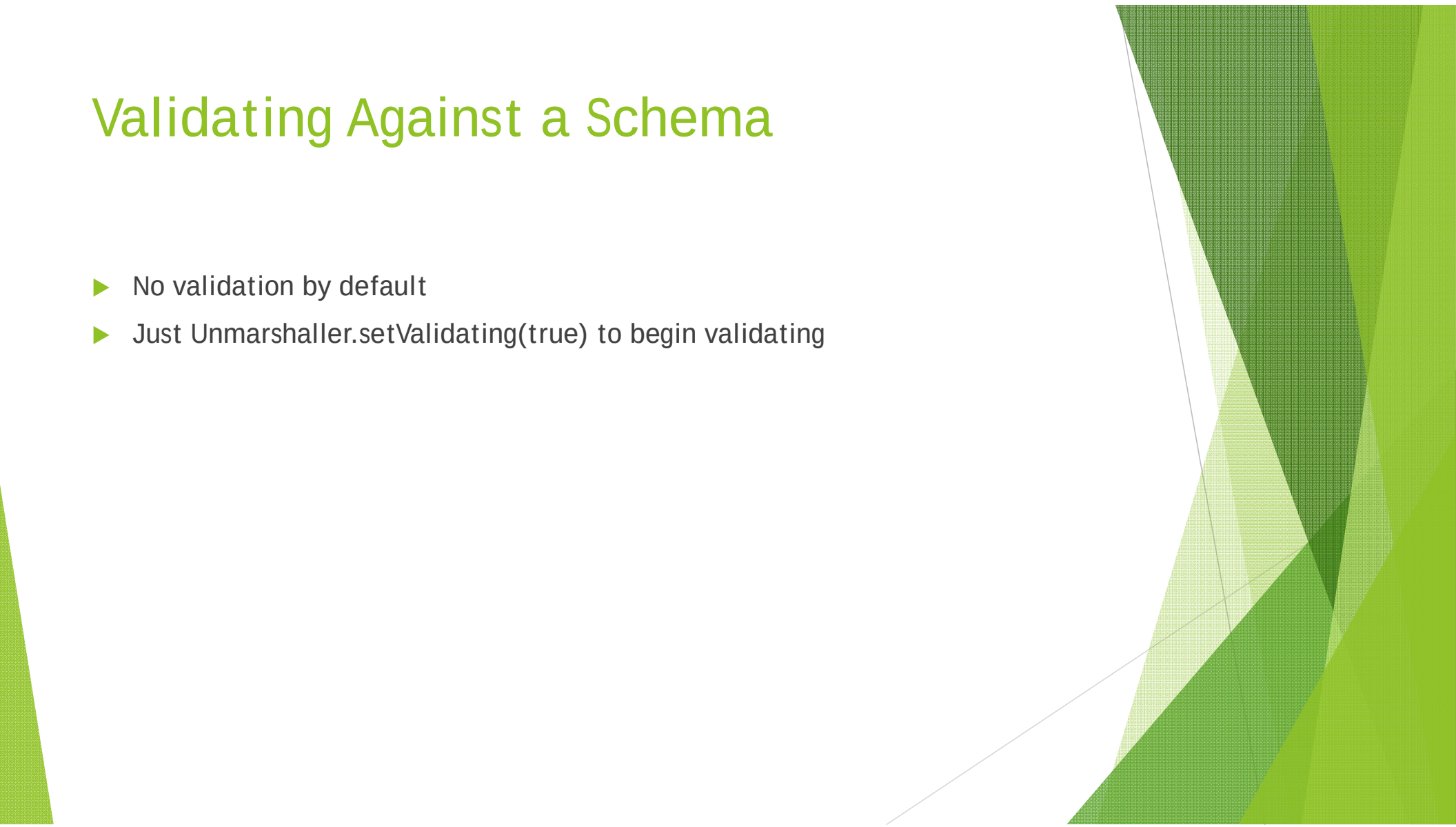
- Field - @XmlAccessorType(XmlAccessType.FIELD)
  - All fields are serialized
- Property - @XmlAccessorType(XmlAccessType.PROPERTY)
  - Any property with a getter/setter is seralized
- Public field - @XmlAccessorType(XmlAccessType.PUBLIC_MEMBER)
  - Public getter/setter
  - Public member
- NONE - @XmlAccessorType(XmlAccessType.NONE)
  - Only serialize the fields explicitly specified with @XmlElement()
- @XmlTransient – do not serialize this field

Custom Mapping

- Color
- Polygons
- DateTime
- Wrapper Elements

Validating Against a Schema

- No validation by default
- Just Unmarshaller.setValidating(true) to begin validating

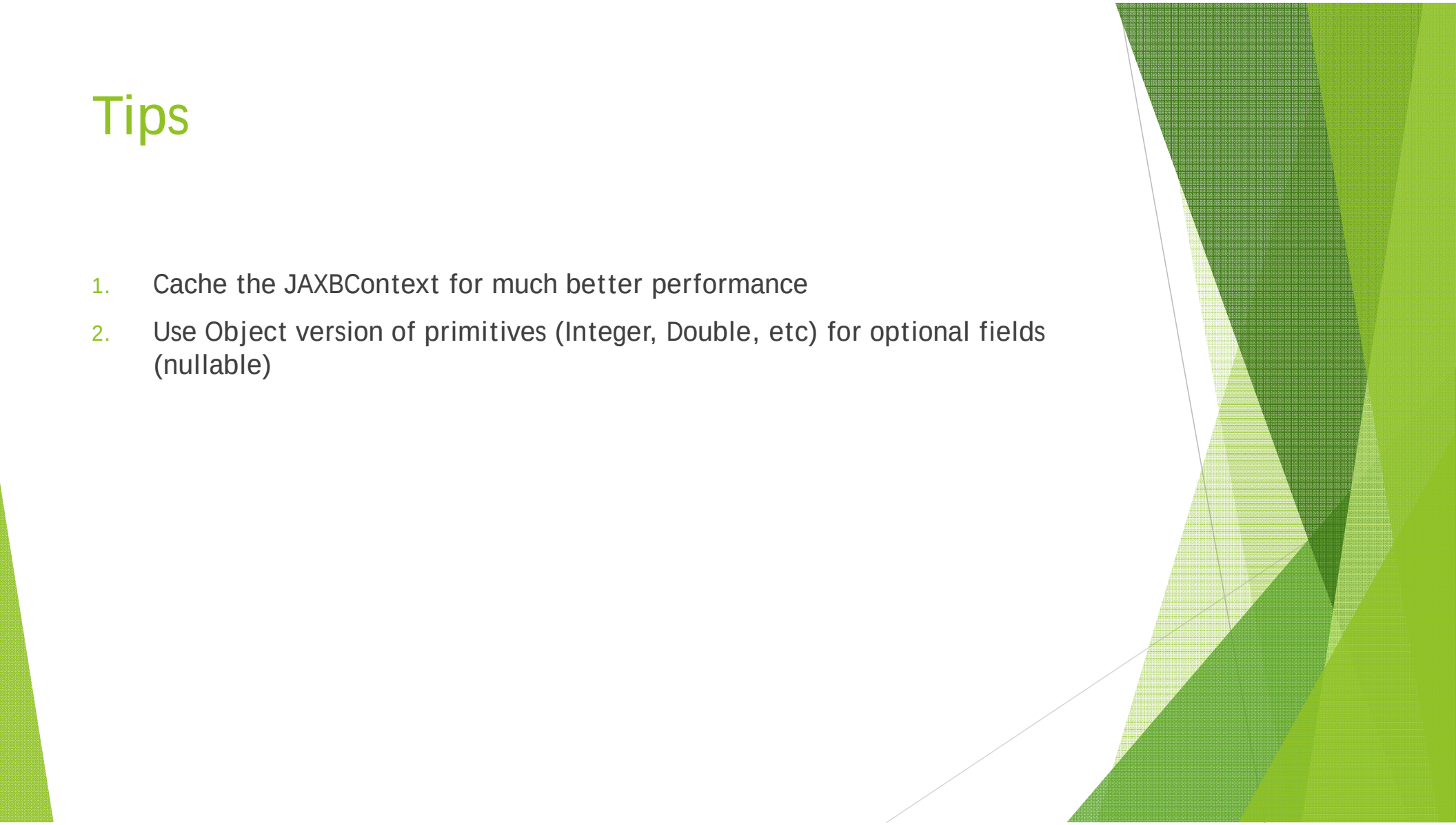


Controlling the Formatting of Generated XML

```
public String toXML(Object jaxbElement) throws Exception {
    StringWriter stringWriter = new StringWriter();
    JAXBContext context = getJaxBContext();
    Marshaller marshaller = context.createMarshaller();
    marshaller.setProperty("jaxb.formatted.output", Boolean.TRUE);
    DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
    Document document = docBuilderFactory.newDocumentBuilder().newDocument();
    marshaller.marshal(jaxbElement, document);
    Transformer transformer = this.getTransformer();
    transformer.transform(new DOMSource(document), new StreamResult(stringWriter));
    return stringWriter.toString();
}
Transformer getTransformer() throws Exception {
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    transformer.setOutputProperty(OutputKeys.CDATA_SECTION_ELEMENTS, "Notes");
    return transformer;
```

Tips

1. Cache the JAXBContext for much better performance
2. Use Object version of primitives (Integer, Double, etc) for optional fields (nullable)



Current Deficits

1. JavaDoc gets lost in translation
2. No standard way to specify data restrictions with annotations (Bean Validation may help…better if they were built into JAXB…See EclipseLink Moxy)

Questions?