Workshop - ODFDOM

Lars Behrmann
Frank Meies
Svante Schubert
Sun Microsystems, Hamburg

Do you know ODF?

- The OASIS / ISO standard for office documents (2005/06)
- The document format of many office applications
- A zipped package of XML and related files (image, sound, user files)
- Origin from OpenOffice.org's default format

OASIS

Open Document Format for Office Applications (OpenDocument) v1.0

OASIS Standard, 1 May 2005

ISO International Organization for Standardization

ISO/IEC 26300

What to do with ODF documents?

- Usually store your Texts / Spreadsheets / Presentation
- Edited and viewing by an Office application

What to do with ODF documents?

- But how do I process 1.000.000 ODF documents?

We need an ODF API!

- API to automate ODF processing
  - Creating, manipulating ODF documents
  - Lightweight API
  - API close related to ODF
  - Opensource

The Idea of a new ODF API!

- We need a new lightweight ODF API!!
  - New API will focus on ODF
  - Taking over ideas from previous ODF APIs (OOo API, AODL, ODF4J, etc.)

New ODF API - ODFDOM

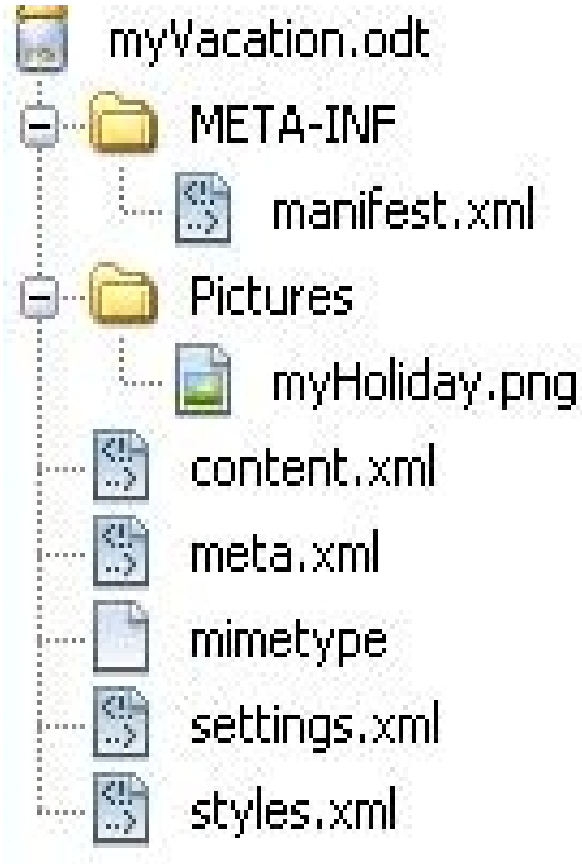
- Sun opensourced ODFDOM
  - Lightweight API
  - OpenDocument centric
  - Opensource (Apache 2)
  - Multi-layered
  - Java 5 reference implementation

ODF Basics - Package & Files

ISO standardized default content
(as shown, but NOT Picture folder)

Manifest as an Inventory /
'table of content'

Any user content..



ODFDOM - Layered Model

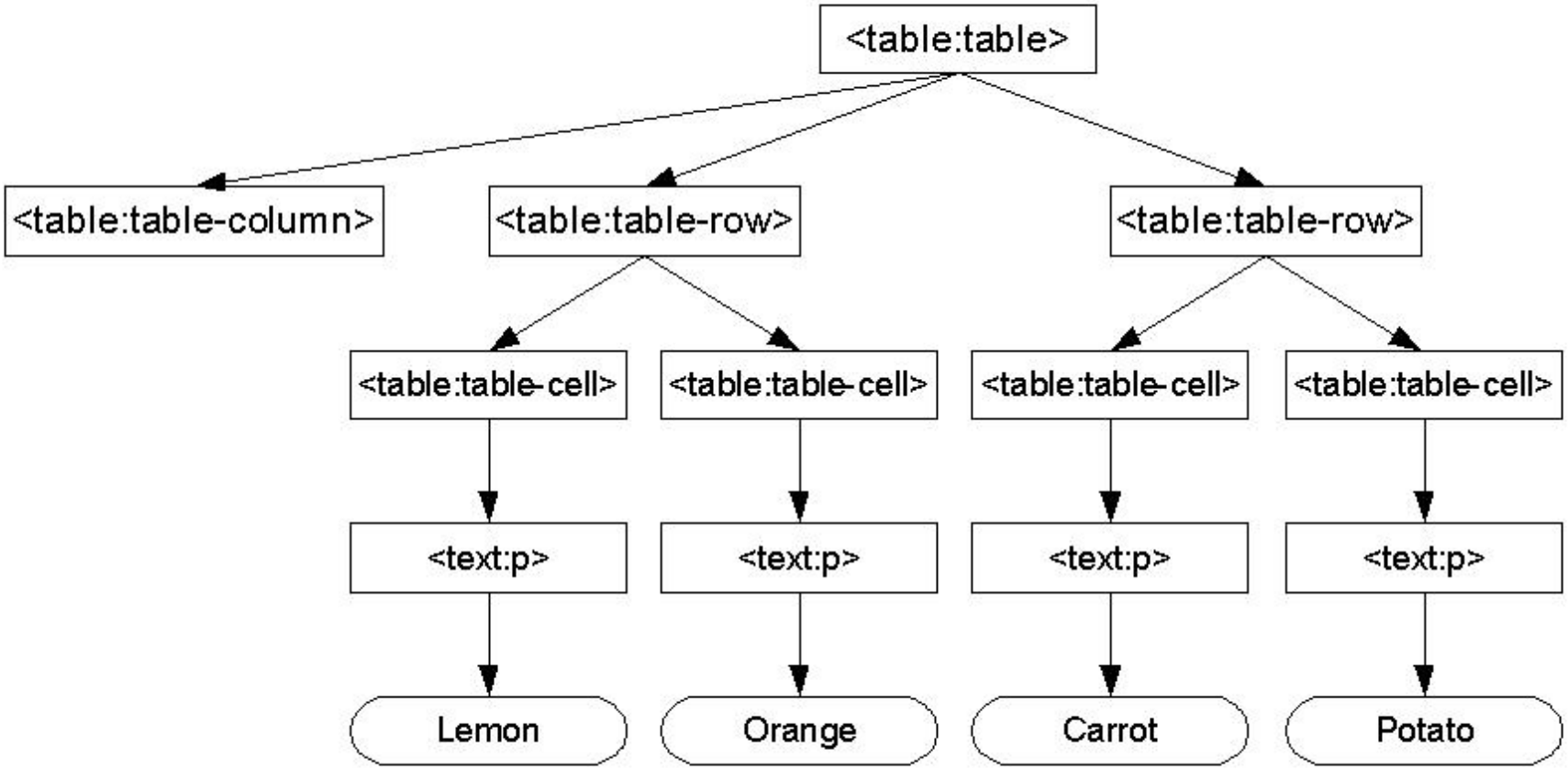
- ODFDOM featuring:
  - Adding / removing file streams from the ODF package (ZIP)

ODF Package / Physical Layer
(managing package file streams)

ODF Basics – XML Table Example

```
<table:table table:name="Table - fruits vs. vegies">.
    <table:table-column table:number-columns-repeated="2"/>.
    <table:table-row>.
        <table:table-cell>.
            <text:p>Lemon</text:p>.
        </table:table-cell>.
        <table:table-cell>.
            <text:p>Orange</text:p>.
        </table:table-cell>.
    </table:table-row>.
    <table:table-row>.
        <table:table-cell>.
            <text:p>Carrot</text:p>.
        </table:table-cell>.
        <table:table-cell>.
            <text:p>Potato</text:p>.
        </table:table-cell>.
    </table:table-row>.
</table:table>.
```

Design Idea - DOM API for ODF



ODFDOM - Layered Model

- ODFDOM featuring:
  - Processing ODF documents on ODF XML element level

ODF Typed DOM / XML Layer
(DOM classes generated from ODF RelaxNG)

ODF Package / Physical Layer
(managing package file streams)

ODFDOM - Layered Model

- ODFDOM featuring:
  - Common high-level convenience functionality (e.g. add table, add table row, etc.)

ODF Document / Convenient Functionality Layer
(frequently used functionality)

ODF Typed DOM / XML Layer
(DOM classes generated from ODF RelaxNG)

ODF Package / Physical Layer
(managing package file streams)

ODFDOM - Layered Model

Customized ODF Document / Extendable Layer
(optional layer not part of ODFDOM)

ODF Document / Convenient Functionality Layer
(frequently used functionality)

ODF Typed DOM / XML Layer
(DOM classes generated from ODF RelaxNG)

ODF Package / Physical Layer
(managing package file streams)

ODFDOM - Layered Model

Customized ODF Document / Extendable Layer
(optional layer not part of ODFDOM)

ODF Document / Convenient Functionality Layer
(frequently used functionality)

ODF Typed DOM / XML Layer
(DOM classes generated from ODF RelaxNG)

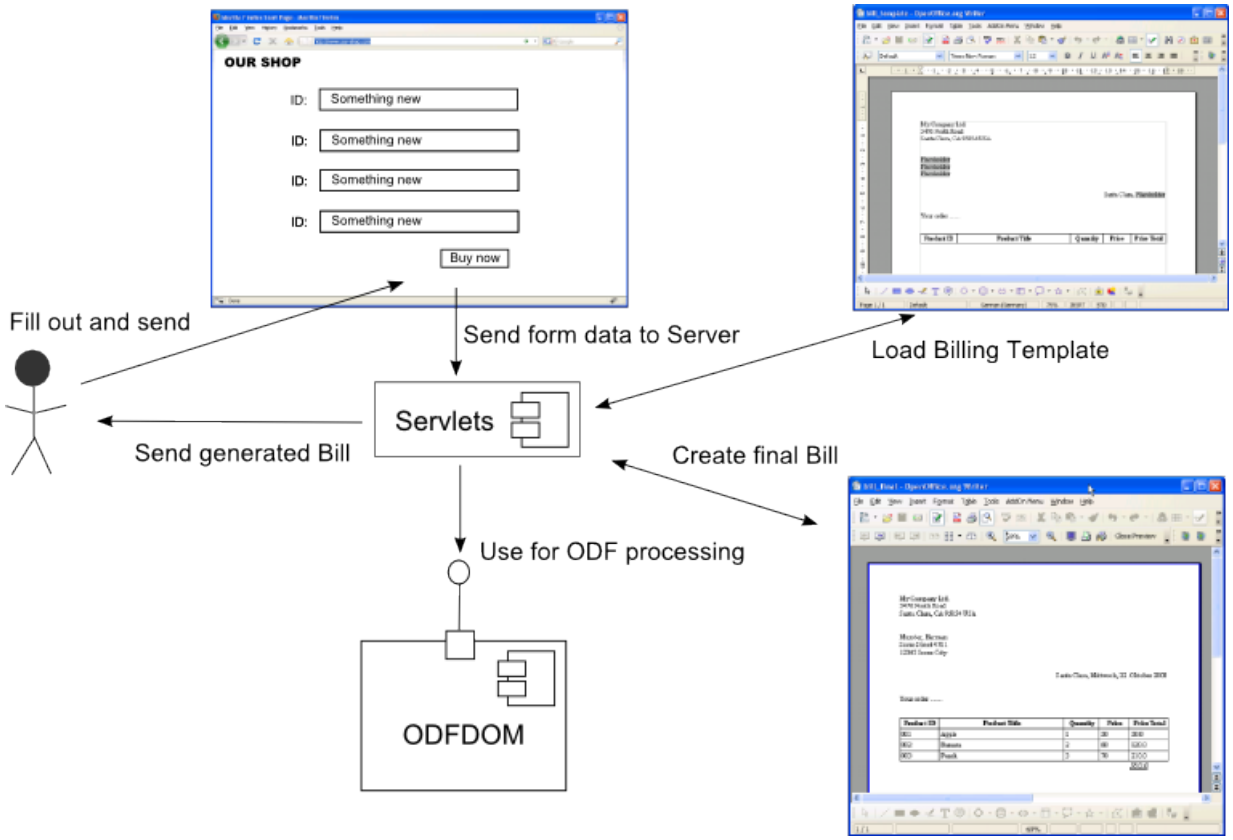
ODF Package / Physical Layer
(managing package file streams)

ODFDOM - Resources

- Quick Look
  - Project of ODF Toolkit - http://odftoolkit.org
  - Wiki - http://odftoolkit.org/projects/odftoolkit/pages/ODFDOM
- Deep Look (Packages)
  - The zipped JavaDoc API
  - The JAR of the reference Java 5 implementation
  - The zipped NetBeans package containing the sources of ODFDOM

Exercise: Text Document (1)

- Goal: Use ODFDOM API to create Company Bills in ODF from HTML Forms based on Templates.

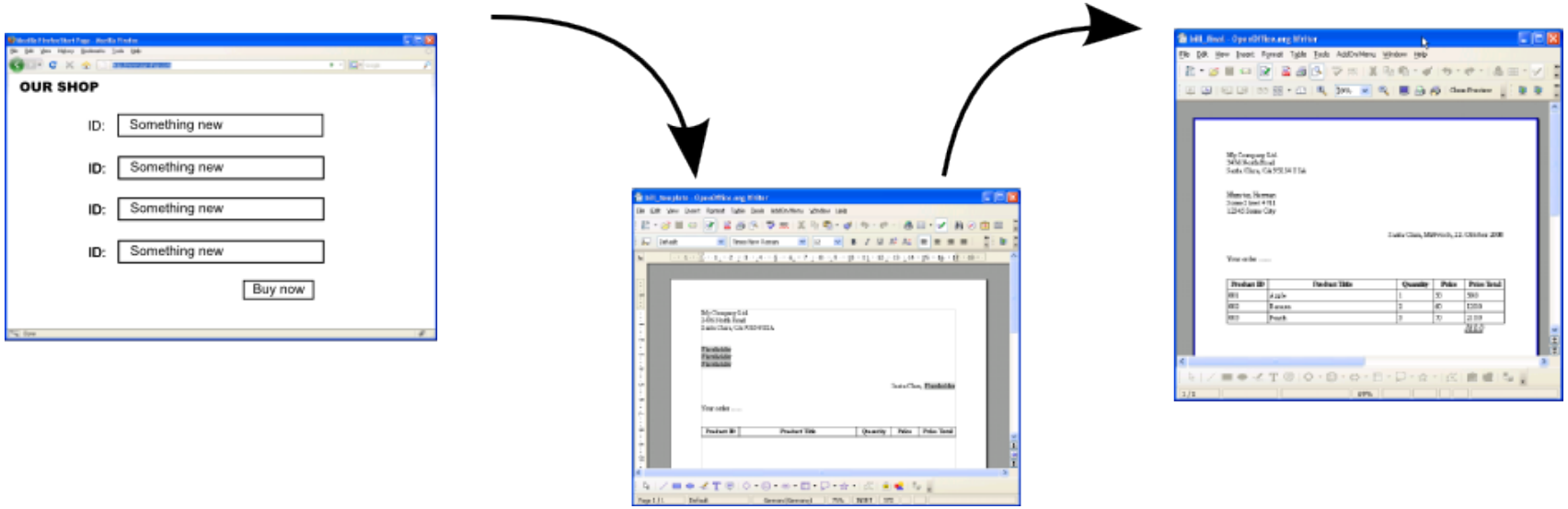


Exercise: Text Document (2)

- Required steps:
- Set up a Web Application Netbeans Project
- Add a JSP File with a HTML Form
- Add required libraries to projects and imports to JSP file
- Add a JSP File which receives the HTML Form Data
- Add Java code that
  - Receives the HTML Form Data and calculate the bought items
  - Load the Billing Template
  - Finds the required place holder
  - Applies the calculated data to the place holder
  - Finally saves the generated Bill

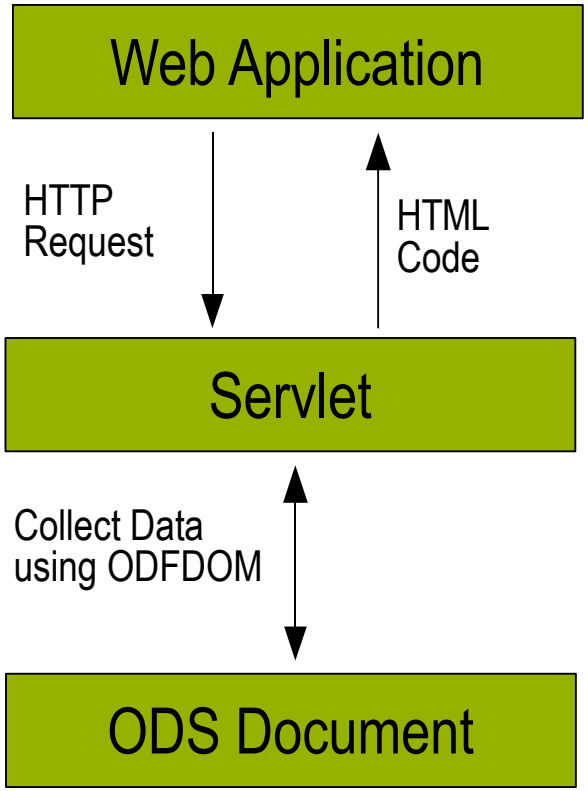
Exercise: Text Document (3)

- After all steps are finished we have our generated Bill:



Exercise: Spreadsheet Document (1)

- Goal: Use ODFDOM API to read data from a spreadsheetdocument and present the data as HTML page:



Exercise: Spreadsheet Document (2)

- Required steps:
- Set up a Web Application Netbeans Project
- Add required libraries to projects
- Add required imports to JSP file
- Add Java code that
  - Finds the table rows in the ODS document
  - Finds the table cells for each table row
  - Gets the text content of each table cell
  - Writes the respective HTML tags for the rows/cells/text content

Exercise: Spreadsheet Document (3)

- The final result should look like this: