XPath

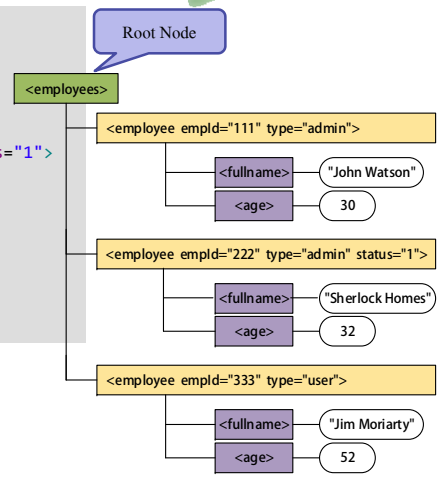
- XPath คือ ประโยคสัญลักษณ์ที่ใช้ในการดึงข้อมูลจากเอกสาร XML เพื่อนำไปประมวลผล หรือแสดงผล
- XPath กำหนดเส้นทางเพื่อไปยังโหนดต่างๆ บนเอกสาร XML
- หากเปรียบเทียบ XML เป็นฐานข้อมูล ภาษา XPath ก็เปรียบเสมือนภาษา SQL

สัญลักษณ์ของ XPath

รูปแบบ	ความหมาย
/	เลือก Node ที่ต้องการโดยเริ่มจาก Root Node
//	แทนเส้นทางของ Node ตั้งแต่ Root Node
.	เลือก Node ปัจจุบัน
..	เลือก Parent Node ของ Node ปัจจุบัน
@	เลือก attribute
nodename	เลือกทุก Node ที่มีชื่อเป็น "nodename"

ตัวอย่างเอกสาร XML

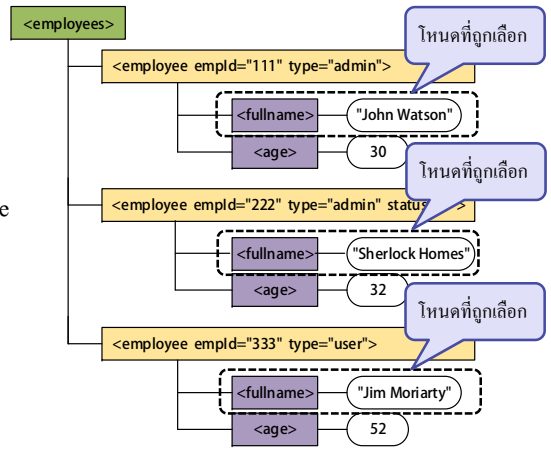
```xml
<?xml version="1.0" encoding="UTF-8"?>
<employees>
    <employee empId="111" type="admin">
        <fullname>John Watson</fullname>
        <age>30</age>
    </employee>
    <employee empId="222" type="admin" status="1">
        <fullname>Sherlock Homes</fullname>
        <age>32</age>
    </employee>
    <employee empId="333" type="user">
        <fullname>Jim Moriarty</fullname>
        <age>52</age>
    </employee>
</employees>
```



Absolute Path

- Absolute Path คือ การเลือกโหนดที่ต้องการ โดยระบุชื่อโหนดทั้งหมดตั้งแต่โหนด root
- ตัวอย่าง

/employees/employee/fullname

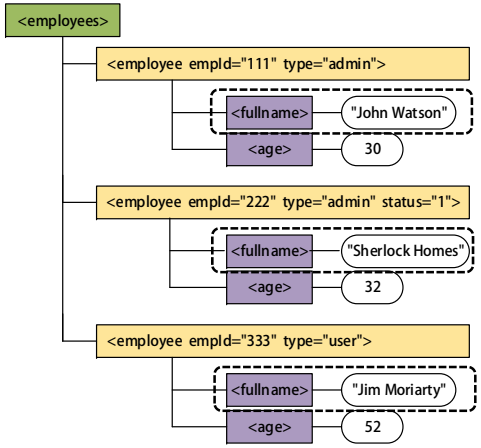


Relative Path

❖ Relative Path คือ การเลือกโหนดที่ต้องการโดยละชื่อโหนดทั้งหมดด้วย //

❖ ตัวอย่าง

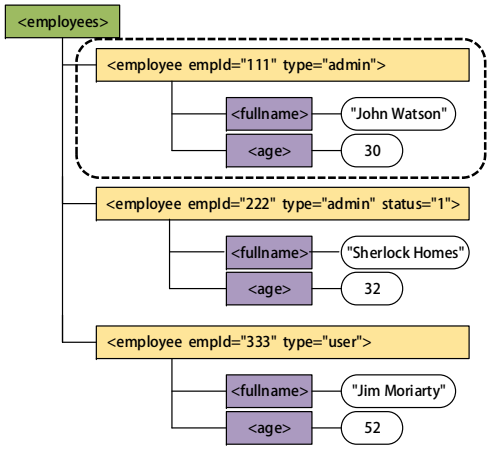
//fullname



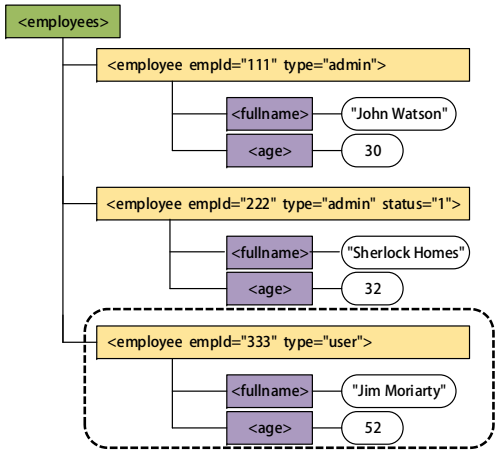
การเลือกแบบมีเงื่อนไข

รูปแบบ	ความหมาย
/employees/employee[1]	เลือกโหนด employee ซึ่งเป็นลูกของ employees เฉพาะโหนดแรก
/employees/employee[last()]	เลือกโหนด employee ซึ่งเป็นลูกของ employees เฉพาะโหนดสุดท้าย
/employees/employee[last()-1]	เลือกโหนด employee ซึ่งเป็นลูกของ employees เฉพาะโหนดรองสุดท้าย
//employee[@status]	เลือกโหนด employee ที่มี attribute ชื่อ status
//employee[@type='admin']	เลือกโหนด employee ที่มี attribute ชื่อ type และมีค่าเป็น "admin"
//employee[@type='admin']/fullname	เลือกโหนด fullname จากโหนด employee ที่มี attribute ชื่อ type และมีค่าเป็น "admin"
/employees/employee[age>40]/fullname	เลือกโหนด fullname จากโหนด employee ที่มีโหนดลูกชื่อ age มีค่ามากกว่า 40
/employees/employee[position() <= 2]/fullname	เลือกโหนด fullname จากโหนด employee ที่อยู่ตำแหน่ง 1 และ 2
//employee/@type	เลือกข้อมูลใน attribute ชื่อ type ของโหนด employee

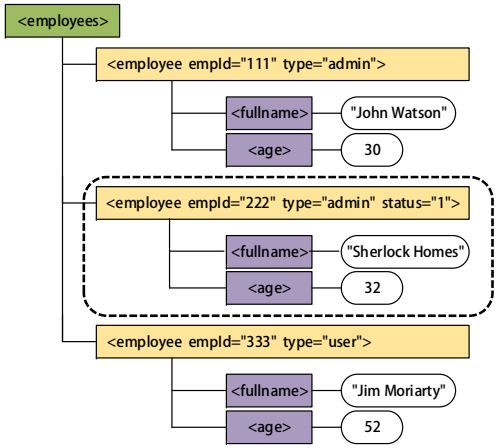
/employees/employee[1]



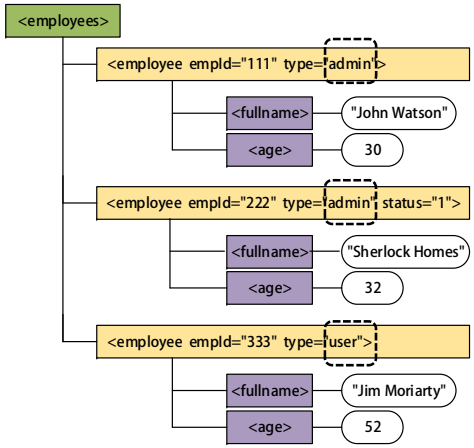
/employees/employee[last()]



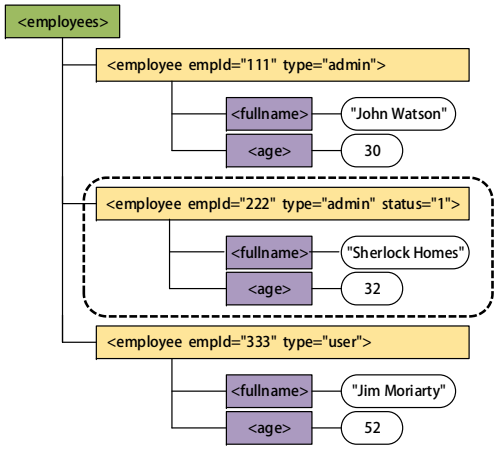
/employees/employee[last()-1]



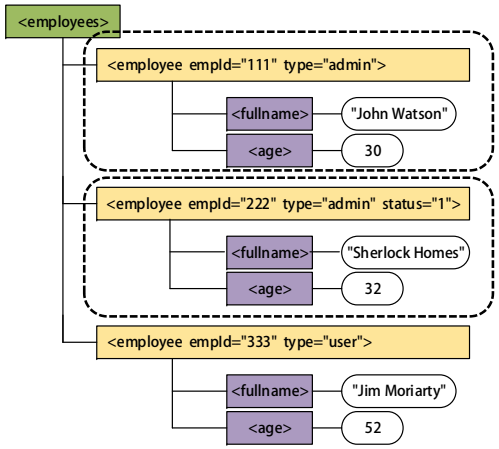
//employee/@type



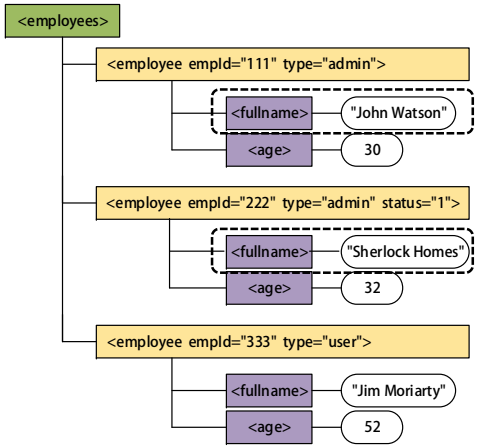
//employee[@status]



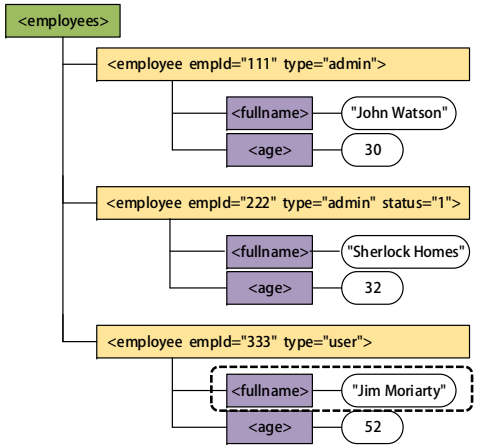
//employee[@type='admin']



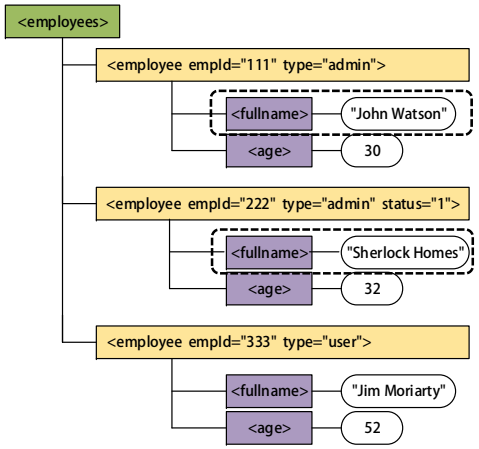
//employee[@type='admin']/fullname



/employees/employee[age>40]/fullname



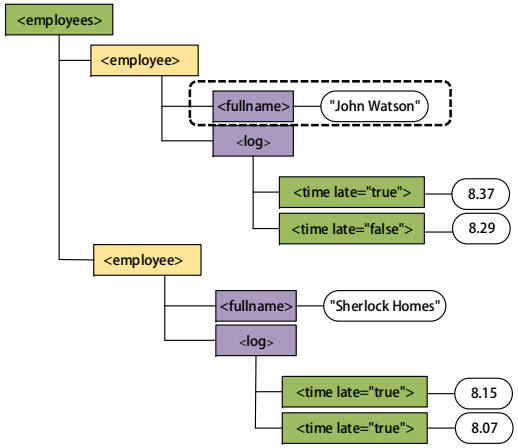
/employees/employee[position()<=2]/fullname



การใช้ ..

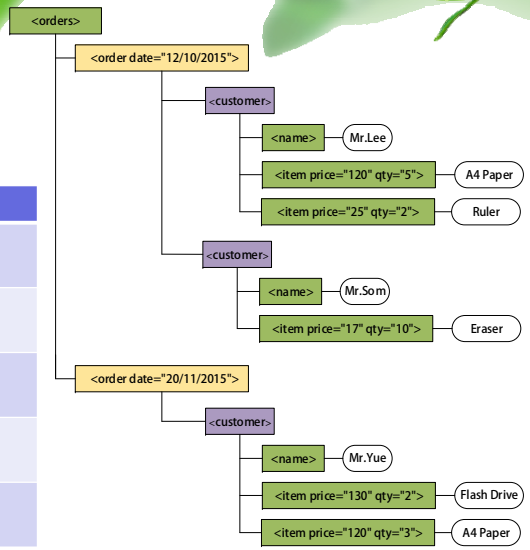
ต้องการชื่อพนักงานที่มี attribute late เป็น true

//time[@late='true']/../../fullname



กิจกรรม

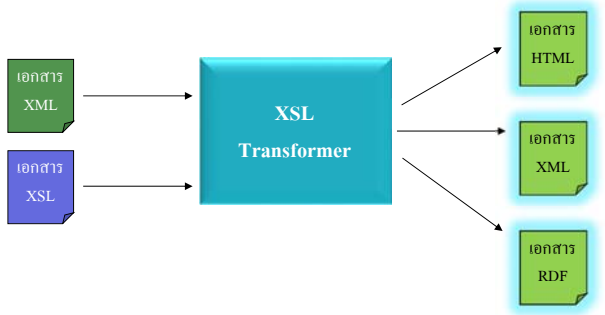
❖ จงเขียน XPath สำหรับดึงข้อมูลจากเอกสาร XML ซึ่งมีโครงสร้างและข้อมูลดังภาพ



เงื่อนไข	XPath
แสดงชื่อลูกค้าทั้งหมด	
แสดงชื่อสินค้าที่มีราคามากกว่า 100 บาท	
แสดงชื่อลูกค้าที่ซื้อของในวันที่ 12/10/2015	
แสดงชื่อลูกค้าที่ซื้อ 'A4 Paper'	
แสดงชื่อลูกค้าที่ซื้อสินค้าที่มีราคามากกว่า 125 บาท	

XSL

❖ XSL (Extensible Stylesheet Language) คือ ภาษาที่ใช้ในการแปลงเอกสาร xml ให้อยู่ในรูปแบบที่ต้องการ เช่น HTML, RDF, หรือ XML ในรูปแบบใหม่



รูปแบบของเอกสาร XSL

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">


   <xsl:output method="html"/>

   <xsl:template match="nodeset">
      <!-- เพิ่มรูปแบบและคำสั่ง XSL ที่นี่ -->
   </xsl:template>


</xsl:stylesheet>
```

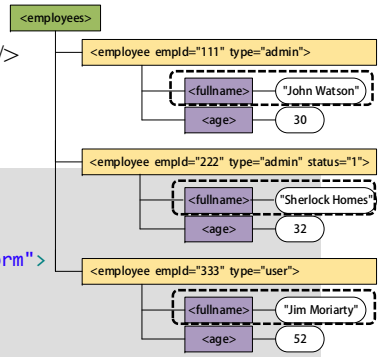
ระบุชนิดรูปแบบปลายทาง

ระบุ Node เริ่มต้นด้วย XPath

การแสดงค่าของ Element หรือ Attribute

❖ รูปแบบคำสั่ง

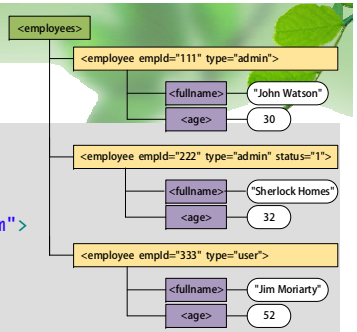
<xsl:value-of select="*XPath Expression*" />




```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html"/>

   <xsl:template match="/">
      Name: <xsl:value-of select="employees/employee/fullname"/><br/>
   </xsl:template>

</xsl:stylesheet>
```

ระบุจุดเริ่มต้น

ตัวอย่าง



เอกสาร XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html"/>

   <xsl:template match="/">
      <h1>Employee List</h1>
      ID: <xsl:value-of select="employees/employee/@empId"/><br/>
      Name: <xsl:value-of select="employees/employee/fullname"/><br/><hr/>
   </xsl:template>

</xsl:stylesheet>
```

ผลลัพธ์

```
<h1>Employee List</h1>
ID: 111<br/>
Name: John Watson <br/><hr/>
```

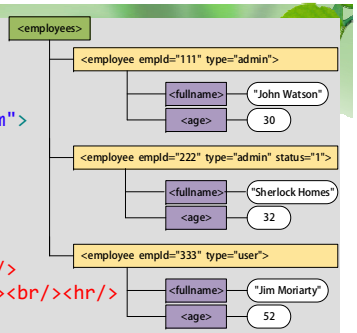

การวนลูปแสดงค่าของ Element

❖ รูปแบบ

<xsl:for-each select="*XPath Expression*">

styles

</xsl:for-each>



เอกสาร XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html"/>

   <xsl:template match="/">
      <h1>Employee List</h1>
      <xsl:for-each select="employees/employee">
          ID: <xsl:value-of select="@empId"/><br/>
          Name: <xsl:value-of select="fullname"/><br/><hr/>
      </xsl:for-each>
   </xsl:template>

</xsl:stylesheet>
```

ผลลัพธ์

```
<h1>Employee List</h1>
ID: 111<br/>
Name: John Watson<br/><hr/>
ID: 222<br/>
Name: Sherlock Homes<br/><hr/>
ID: 333<br/>
Name: Jim Moriarty<br/><hr/>
```


การสร้าง Template ให้แต่ละ Element

❖ รูปแบบ

- กำหนดตำแหน่งที่จะแสดงผล

  <xsl:apply-templates select="*XPath Expression*" />

- สร้าง template ให้ Element

  <xsl:template match="ชื่อ Element">

  styles

  </xsl:template>

เอกสาร XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html"/>

   <xsl:template match="/">
      <h1>Employee List</h1>
      <xsl:apply-templates select="employees/employee" />
   </xsl:template>

   <xsl:template match="employee">
       ID: <xsl:value-of select="@empId"/><br/>
       Name: <xsl:value-of select="fullname"/><br/><hr/>
   </xsl:template>
</xsl:stylesheet>
```

ผลลัพธ์

```
<h1>Employee List</h1>
ID: 111<br/>
Name: John Watson<br/><hr/>
ID: 222<br/>
Name: Sherlock Homes<br/><hr/>
ID: 333<br/>
Name: Jim Moriarty<br/><hr/>
```


การเรียงลำดับข้อมูล

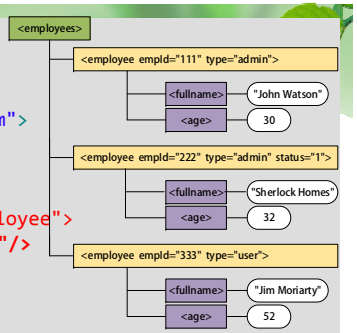
❖ รูปแบบ

```
<xsl:sort  select = "XPath Expression"
        data-type = { "text" | "number" | QName }
        order = { "ascending" | "descending" }
        case-order = { "upper-first" | "lower-first" } />
```

- select ระบุชื่อ Element ที่ต้องการเรียงลำดับ
- data-type ระบุชนิดของข้อมูล
- order ระบุวิธีการเรียงข้อมูล จากน้อยไปมาก (ascending) หรือ จากมากไปน้อย (descending)
- case-order ระบุว่าเรียงตามตัวพิมพ์ใหญ่ก่อน (upper-first) หรือตัวพิมพ์เล็กก่อน (lower-first)

เอกสาร XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html"/>
   <xsl:template match="/">
      <h1>Employee List</h1>
      <xsl:apply-templates select="employees/employee">
        <xsl:sort select="age" order="descending"/>
      </xsl:apply-templates>
   </xsl:template>
   <xsl:template match="employee">
       ID: <xsl:value-of select="@empId"/><br/>
       Name: <xsl:value-of select="fullname"/><br/><hr/>
   </xsl:template>
</xsl:stylesheet>
```



ผลลัพธ์

```
<h1>Employee List</h1>
ID: 333<br/>
Name: Jim Moriarty<br/><hr/>
ID: 222<br/>
Name: Sherlock Homes<br/><hr/>
ID: 111<br/>
Name: John Watson<br/><hr/>
```