Informační a řídicí systémy I.
OPC XML Data Access

Pavel Balda
ZČU v Plzni, FAV, KKY

Osnova přednášky

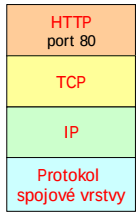
- Protokol HTTP
- XML
- XML Schema
- SOAP
- OPC XML Data Access

Doporučená literatura

- World Wide Web Consortium (W3C): http://www.w3.org
- Extensible Markup Language (XML) 1.1 http://www.w3.org/TR/2004/REC-xml11-20040204
- XML Schema Part 0: Primer Second Edition http://www.w3.org/TR/xmlschema-0
- XML Schema Part 1: Structures Second Edition http://www.w3.org/TR/xmlschema-1
- XML Schema Part 2: Datatypes Second Edition http://www.w3.org/TR/xmlschema-2
- SOAP Version 1.2 Part 0: Primer (Second Edition) http://www.w3.org/TR/2007/REC-soap12-part0-20070427
- SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) http://www.w3.org/TR/2007/REC-soap12-part1-20070427
- SOAP Version 1.2 Part 2: Adjuncts (Second Edition) http://www.w3.org/TR/2007/REC-soap12-part2-20070427
- OPC XML-DA Specification Version 1.01 http://www.opcfoundation.org
- Harold, E.R., Means, W.S.: XML v kostce, Computer Press Praha, 2002

Protokol HTTP

- Aplikační protokol založený na TCP/IP
  - Určen pro výměnu zpráv (dokumentů) s libovolným obsahem
  - Klient-server architektura
  - Každý požadavek (request) klienta je následován odpovědí (response) serveru
  - Bezestavový protokol, využívá port 80
- Základní protokol Webu
  - Web je množinou procesů (klientských, serverovských) běžících v Internetu, které komunikují pomocí HTTP

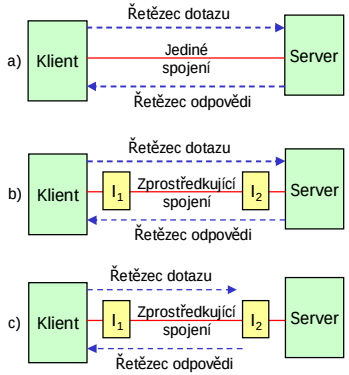


HTTP klient a server

- Klient
  - Internetový prohlížeč schopný zobrazovat HTML stránky
- Server
  - Pracuje se stránkami, které předává klientům
- Stránky jsou identifikovány pomocí URL (Uniform Resource Locator) ve tvaru:
  <protocol>://<host_name>/<file_name>
  - **<protocol>** – protokol pro komunikaci se serverem, např. **http**, **ftp**
  - **<host_name>** (jméno počítače) je přeloženo do IP adresy počítače, např. **www.kky.zcu.cz** má adresu **147.228.47.12**
  - **<file_name>** – jméno souboru na počítači **<host_name>**

Komunikace mezi klientem a serverem

- Obr. a): Přímá komunikace mezi klientem a serverem
- Obr. b): Komunikace přes zprostředkující uzly (proxy servery)
  - V tomto případě dojde požadavek až na server
- Obr. c): Komunikace přes zprostředkující uzly
  - V tomto případě druhý zprostředkující uzel má požadovanou informaci na svém disku nebo v paměti (cache) a obslouží požadavek klienta sám



Formát požadavku HTTP

- Požadavek HTTP má následující textový formát
  první řádek: **<method> <URL> <protocol_version> CRLF**
  následován: **<header> CRLF**
  následován: **<data>**
  - Hlavička může obsahovat více než jeden řádek
- **<method>** = **GET** | **HEAD** | **POST** | **PUT** | ...
- **<protocol_version> = HTTP/1.1** | ...
- **<header>** = **<field_name>:<value> CRLF**
- **<field_name>** =
  - **From** | - emailová adresa odesilatele
  - **Accept** | - přípustné formáty odpovědi
  - **User-Agent** | - identifikuje program klienta (název a verze prohlížeče)
  - **Referer** | - URL dokumentu obsahujícího odkaz (pro gener. zpětných odkazů)
  - **If-Modified-Since** | - pošli dokument byl-li modifikován od daného času (užíván s **GET**)
  - **Content-Type** | - typ data (pro SOAP a XML)
  - **Host** | - cílový počítač
  - ...
- **<data>** = text v ASCII kódu (default)

Metody protokolu HTTP

- **GET** – požadavek na poslání dat z daného URL
- **HEAD** – požadavek na poslání hlavičky dat z daného URL, na rozdíl od **GET** neposílá vlastní data
- **PUT** – tělo požadavku obsahuje data, která se mají uložit pod dané URL
- **DELETE** – ruší data na daném URL
- **POST** – tělo požadavku obsahuje nový objekt, který je přidružen k danému URL (např. soubor do adresáře daného URL, email, data formuláře, apod.)
- **TRACE** – testování serveru, který má vrátit kladnou odpověď bez dat
- **OPTIONS** – umožňuje klientovi určit omezení a možnosti serveru pro dané URL
- Server nemusí podporovat všechny metody. **GET**, **HEAD** a **POST** jsou podporovány všemi současnými servery

Formát odpovědi HTTP

- Odpověď HTTP má následující formát
  stavový řádek: **<HTTP_version> <status_code><reason_line>CRLF**
  následován: **<header> CRLF**
  následován: **<data>**
  - Hlavička může obsahovat více než jeden řádek
- **<status_code>** = 3 číslice
  - **2xx** - úspěch
  - **4xx** - špatný požadavek od klienta
  - **5xx** - chyba serveru při obsluze správného požadavku
- **<reason_line>** = vysvětlení v „lidsky čitelné“ podobě
- **<header>** = <field_name>:<value> CRLF
- **<field_name>** =
  - **Allowed** | - metody podporované daným URL
  - **Date** | - datum a čas vytvoření odpovědi
  - **Expires** | - doba vypršení platnosti dat
  - **Last-Modified** | - datum vytvoření objektu
  - **Content-Length** | - délka těla v bajtech (oktetech)
  - **Content-Type** | - typ dat
  - ...

Úvod do XML

- XML (eXtensible Markup Language) popisuje množinu objektů nazývaných XML dokumenty
  - XML částečně definuje chování programů, které tyto dokumenty zpracovávají
  - XML vychází z SGML (Standard Generalized Markup Language) standardizovaného normou ISO 8879
  - Standard XML je spravován konsorciem W3C (World Wide Web Consortium)
  - Standard XML 1.0 byl vydán jako doporučení W3C (W3C Recommendation) v roce 1998
  - Standard XML 1.1 byl vydán v roce 2004
- Dokumenty XML jsou textové soubory obsahující:
  - Prolog specifikující verzi XML (u verze 1.1 je povinná)
  - Značky k popisu dat
  - Data ve formě textových řetězců
- Příklad
  - ```
    <?xml version="1.1"?>
    <pozdrav>Nazdar, světe!</pozdrav>
    ```
  - Bude-li chybět první řádek, půjde o XML soubor verze 1.0

Co není XML

- XML je standard určující syntaxi pro vytváření vlastních značkovacích jazyků
- XML není programovací jazyk, neexistuje pro něj žádný překladač
  - Činnost vykonává program, nikoliv XML. Program může ukládat svá data nebo konfiguraci do souborů XML
- XML není „vylepšený“ HTML
  - XML používá sice podobnou syntaxi jako HTML, není však vázán na formátování textu v prohlížeči
  - XML je metajazyk, umožňuje definovat vlastní značkovací jazyky pro konkrétní aplikace
- XML není síťový přenosový protokol
  - Dokumenty XML mohou být posílány po síti standardními přenosovými protokoly, např. HTTP, FTP, apod.
- XML není databáze
  - Nenahrazuje SQL databáze
  - Data z databází lze však do XML konvertovat, silná podpora je v .NET Framework a v C#

Správně formované dokumenty

- Správně formované dokumenty (well-formed documents, WFD) vyhovují XML specifikaci
  - Není-li dokument správně formován, nelze jej dále zpracovávat (fatální chyba)
- WFD obsahuje zejména Prolog a kořenový element.
- Prolog obsahuje:
  - Verzi XML
  - Může obsahovat deklaraci typu dokumentu (Document Type Declaration, DTD)
- Kořenový element
  - Může obsahovat vnořené elementy
  - Elementy musí být správně vnořeny, např. `<a><b></a></b>` je špatně
- Příklad
  - ```
    <?xml version="1.1" encoding="UTF-8" ?>
    <!DOCTYPE pozdrav [
    <!ELEMENT pozdrav (#PCDATA)>
    ]>
    <pozdrav>Nazdar, světe!</pozdrav>
    ```

Značkování (Markup) v XML

- Značkování je vždy uvozeno speciálními znaky. Uvedeme jednotlivé typy značek s příklady:
  - Počáteční značka s případnými atributy: `<předmět hodin_týdně='3'>`
  - Koncová značka: `</předmět>`
  - Prázdný element – nemá žádný obsah. Buď se užívá syntaxe složená z počáteční značky následované koncovou značkou nebo speciální syntaxe: `<obrázek soubor="http://www.exampleweb.com/images/demo" />`
  - „Entitní reference“ – náhradní symboly pro znaky nepřípustné uvnitř dat, např.: `<` místo znaku `<`
  - Znakové reference: `&#65;` (dekadický kód znaku) nebo `&#x41;` (hexadecimální kód)
  - Komentáře: `<!-- toto je komentář -->`. Explicitně je zakázáno ukončení pomocí `--->`
  - Sekce `CDATA`
  - Deklarace typu dokumentu:
    - `<!DOCTYPE pozdrav SYSTEM "pozdr.dtd">`
    - `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
  - Instrukce pro zpracování – uvnitř závorek `<?` a `?>`

Názvy, entitní reference, CDATA v XML

- Názvy jsou v XML tvořeny posloupností znaků:
  - Prvním znakem mohou být: „:“ [A-Z] „_“ [a-z] a většina znaků se sady Unicode
  - Druhým a dalším znakem mohou být znaky jako první znak a navíc: „-“ „.“ [0-9] a několik znaků ze sady Unicode
  - Znaky, vyskytující se v názvech byly upřesněny ve specifikaci XML 1.1
- Entitní reference – náhrada znaků, které se nesmějí vyskytnout v datech
  - Znak & je nahrazen `&`
  - Znak < je nahrazen `<`
  - Znak > lze nahradit `>`
  - Znak ' lze nahradit `&apos;`
  - Znak " lze nahradit `&quot;`
- Sekce `CDATA` slouží pro ohraničení textu (např. zdrojového kódu XML), kde není třeba nahrazovat znaky `&` a `<` entitními referencemi, např.:

```
<![CDATA[
  <předmět>Programové prostředky řízení</předmět>
]]>
```


Definice typu dokumentů (DTD)

- DTD přesně popisují, jaké elementy a entity se mohou v dokumentu vyskytovat
  - Mohou být v samostatném souboru nebo přímo v dokumentu
  - Samostatný soubor umožňuje využít dané DTD ve více XML dokumentech
- Validace – proces ověření platnosti dokumentu (při zpracování XML je nepovinný), tj. porovnání dokumentů s jejich DTD.
  - Chyby ve validaci nejsou na rozdíl od chyb ve správném formování fatální
- Příklad DTD pro element osoba

```
<!ELEMENT osoba (jméno, zaměstnání*)>
<!ELEMENT jméno (křestní_jméno, příjmení)>
<!ELEMENT křestní_jméno (#PCDATA)>
<!ELEMENT příjmení (#PCDATA)>
<!ELEMENT zaměstnání (#PCDATA)>
```

  - `#PCDATA` je klíčové slovo, říkající, že jde o text, který nesmí obsahovat značky ani vložené elementy
  - `*` značí „žádný, jeden nebo více“, `+` znamená „jeden nebo více“, `?` značí „žádný nebo jeden“ element
- DTD je pro výměnu typových dat koncepčně zastaralé, proto byl hledán způsob, jak jej nahradit. Zdá se, že řešením jsou tzv. XML Schémata

Jmenné prostory

- Jmenné prostory slouží ke dvěma hlavním účelům
  - Rozlišit elementy a atributy z různých aplikací XML se stejným názvem
  - Sdružení všech elementů a atributů z jedné aplikace XML, aby se s nimi dalo snadno pracovat
  - Detaily viz http://www.w3.org/TR/REC-xml-names/
- Kvalifikovaný název
  - Název doplněný o prefix identifikující jmenný prostor oddělený znakem ':'
  - Prefix se používá pro zkrácení URI (Uniform Resource Identifier, jednotný identifikátor zdrojů). URI je nadmnožinou URL (Uniform Resource Locator).
  - Při porovnávání URI jmenných prostorů se rozlišují malá a velká písmena!
  - Deklarace jmenného prostoru má oblast platnosti (scope) od začátku počáteční značky ve které je uvedena do koncové značky
- Příklady napojení prefixů na URI:
  - `<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">`
  - `<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">`

XML Schémata

- XML Schema je doporučení W3C popisující tzv. definiční jazyk XML schémat (XML Schema definition Language)
- XML schémata výrazně rozšiřují možnosti DTD, umožňují specifikovat typy elementů a atributů, apod. Výhody:
  - Integrace se jmennými prostory
  - Mnoho vestavěných typů, uživatelsky definované typy
- Přesná specifikace je značně rozsáhlá, podrobnosti viz W3C, nebo http://www.w3schools.com/schema/
- Příklad definice typu **ČeskáAdresa**

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="českáAdresa">
    <xsd:sequence>
      <xsd:element name="jméno" type="xsd:string"/>
      <xsd:element name="ulice" type="xsd:string"/>
      <xsd:element name="město" type="xsd:string"/>
      <xsd:element name="PSČ"   type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="stát" type="xsd:NMTOKEN"
      fixed="Česká Republika"/>
  </xsd:complexType>
  <!-- Další definice ... -->
</xsd:schema>
```


Komunikace mezi aplikacemi v síti

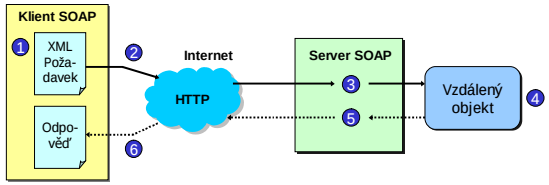
- COM (Component Object Model)
  - Vhodný pro komunikaci mezi aplikacemi na jednom počítači
  - Odstranil závislost na programovacím jazyku, v němž jsou komponenty vytvářeny
- DCOM (Distributed COM)
  - umožňuje komunikovat data mezi aplikacemi na různých počítačích
  - DCOM funguje dobře v síti počítačů s OS Windows
  - Neřeší však nekompatibilitu s jinými komponentovými technologiemi z prostředí OS Unix/Linux (např. Enterprise Java Beans nebo CORBA).
  - Je třeba odstranit vazbu mezi daty a aplikacemi jež je zobrazují.
- Řešením je SOAP (Simple Object Access Protocol)
  - Odstraňuje problémy přístupu k datům plynoucí z neslučitelnosti různých operačních systémů
  - Umožňuje vytvářet Webové služby (Web Services) silně podpořené v .NET Frameworku a C#

SOAP – Simple Object Access Protocol

- SOAP definuje mechanismus výměny strukturovaných a typových informací pomocí XML mezi počítači v decentralizovaném distribuovaném prostředí
  - SOAP definuje jednoduchá pravidla pro zabalování dat do modulů,
  - Nedefinuje však programový model aplikací ani implementaci
  - SOAP může být používán v široké oblasti aplikací od systémů zpráv (messaging systems) až po RPC (Remote Procedure Calls)
- SOAP se skládá ze tří částí:
  - SOAP envelope (obálka) – definuje co je ve zprávě, kdo by se zprávou měl zabývat a zda je to volitelné nebo povinné
  - SOAP encoding rules (kódovací pravidla) – definují mechanismus serializace (ukládání a načítání dat) využitelný pro výměnu instancí datových typů definovaných v dané aplikaci
  - SOAP RPC representation – definuje konvenci pro vzdálené volání procedur a vracení návratových hodnot
- Vztahy mezi SOAP a XML
  - SOAP by měl používat správné jmenné prostory u všech elementů a atributů ve všech zprávách, které generuje
  - Nesmí používat DTD (Document Type Declaration) a nesmí používat instrukce pro zpracování (processing instructions)
  - SOAP používá lokální nekvalifikovaný atribut „**href**“ ve shodě se specifikacemi XML, XML Schema a XML Linking Language.

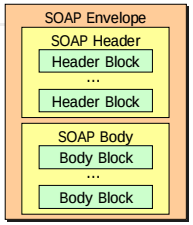
Volání vzdálených objektů pomocí SOAP

- Volání vzdálených metod objektů pomocí SOAP a HTTP připojení:
  - 1. Klient SOAP vytvoří XML dokument obsahující údaje potřebné pro vzdálené volání metody objektu a zabalí jej do obálky SOAP
  - 2. Celý balíček odešle pomocí protokolu HTTP
  - 3. Server naslouchá, balíček přijme, zanalyzuje a s doštými parametry zavolá příslušný vzdálený objekt
  - 4. Objekt provede požadovanou operaci a výsledek vrátí serveru SOAP
  - 5. Obálka s dokumentem SOAP je odeslána zpět na počítač, odkud byl poslán požadavek
  - 6. Klient přijme odpověď a z obálky vyjme výsledný XML dokument



Struktura zprávy SOAP

- Dokument SOAP je „obálkou“ (envelope), ve které je umístěna datová část. Obálka se skládá ze dvou částí:
  - Hlavička (Header) – obsahuje rozšiřující informace o zprávě, např. o transakci, autentikaci, apod.
  - Tělo (Body) – obsahuje povinné informace pro konečného příjemce zprávy. V požadavku obsahuje tagy definované metodou, která se volá; v odpovědi obsahuje vytvořená data
- Příklad




```
<?xml version="1.0" encoding="UTF-8" ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope">
   <env:Header>
      <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
         <n:priority>1</n:priority>
         <n:expires>2001-06-22T16:30:00+01:00</n:expires>
      </n:alertcontrol>
   </env:Header>
   <env:Body>
      <m:alert xmlns:m="http://example.org/alert">
         <m:msg>Vyzvedni Martina ve škole v 16:30</m:msg>
      </m:alert>
   </env:Body>
</env:Envelope>
```


Kódování v SOAP

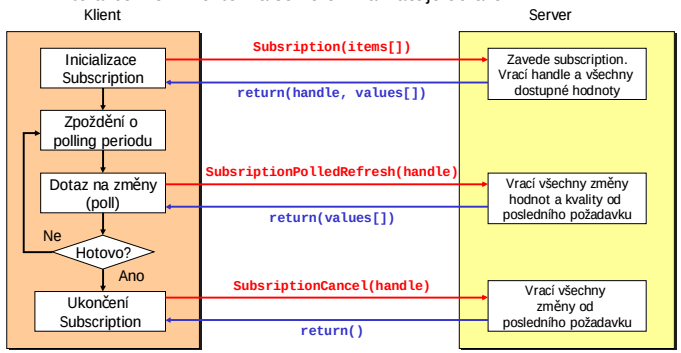
- Kódování je v SOAP založeno na jednoduchém systému typů, které zobecňují rysy typových systémů programovacích jazyků a databází
- Daný typ může být
  - Jednoduchý typ (skalární)
  - Složený typ (compound) – sestává s několika částí, z nichž každá je nějakého typu
- Jednoduché typy – většinou převzaty ze specifikace „XML Schema Part 2: Datatypes“
  - Vestavěné typy (built-in datatypes)
  - Výčtové typy (Enumerations)
- Složené typy – jsou definovány typy odpovídající typům z programovacích jazyků:
  - Struktura
  - Pole

OPC XML Data Access (OPC XML-DA)

- OPC XML-DA je přizpůsobení technologií XML pro výměnu dat z procesů přes internet a do oblasti podnikových aplikací
- Specifikace OPC XML-DA navazuje na specifikaci OPC DA 3.0 (a 2.0)
- OPC XML-DA využívá koncepci SOAP 1.1
  - Data jsou přenášena prostřednictvím SOAP Body
- OPC XML-DA ve verzi 1.0 nedetekuje servery. Klient musí znát URL serveru
- Podporuje lokalizaci pro textové informace o chybách a proměnné typu string
- Podporuje asynchronní přenos informace – tzv. Subscription architecture
  - Klient inicializuje subscription
  - Ve smyčce čeká po určitou dobu (tzv. polling period), pak se dotáže na změny (poll for changes). Tento postup opakuje dokud je třeba
  - Nakonec ukončí subscription
- OPC XML-DA umožňuje klientovi předávat bufferovaná data (více než jednu hodnotu pro jednu položku).
- Každá položka má časovou značku a kvalitu

OPC XML-DA: Basic Polled Subscription

- Interakce mezi klientem a serverem naznačuje obrázek



Příklad: Čtení položek s indikací chyby

```
<soap:Body>
  <ReadResponse
    xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
    <ReadResult RcvTime="2003-05-26T15:55:14.0250000-07:00„
      ReplyTime="2003-05-26T15:55:14.1250000-07:00"
      ServerState="running" />
    <RItemList>
      <Items ItemName="Simple Types/UInt1"
        ResultID="E_UNKNOWNITEMNAME">
        <Quality QualityField="bad" />
      </Items>
      <Items ItemName="Simple Types/UInt2"
        ResultID="E_UNKNOWNITEMNAME">
        <Quality QualityField="bad" />
      </Items>
    </RItemList>
    <Errors ID="E_UNKNOWNITEMNAME">
      <Text>The item name is no longer available in the server
address space.</Text>
    </Errors>
  </ReadResponse>
</soap:Body>
```


Jednoduché typy v OPC XML-DA

Datový typ	Typ VARIANT	Datový typ	Typ VARIANT
string	VT_BSTR	unsignedInt	VT_UI4
boolean	VT_BOOL	unsignedShort	VT_UI2
float	VT_R4	unsignedByte	VT_UI1
double	VT_R8	base64Binary	VT_UI1 | VT_ARRAY
decimal	VT_CY	dateTime	VT_DATE
long	VT_I8	time [2)]	VT_DATE
int	VT_I4	date [2)]	VT_DATE
short	VT_I2	duration [2)]	VT_BSTR
byte [1)]	VT_I1	QName [3)]	---
unsignedLong	VT_UI8	anyType	VT_VARIANT

- [1)] Na rozdíl od většiny jazyků je typ `byte` se znaménkem
- [2)] Viz W3C "XML Schema Part 2: Datatypes"
- [3)] Plně kvalifikované jméno skládající se ze jména a jmenného prostoru. Nemá ekvivalent ve struktuře **VARIANT**.

Pole v OPC XML-DA

Typ pole: ArrayOf…	Typ prvku pole
… Byte	byte
… Short	short
… UnsignedShort	unsignedShort
… Int	int
… UnsignedInt	unsignedInt
… Long	long
… UnsignedLong	unsignedLong
… Float	float
… Decimal	decimal
… Double	double
… Boolean	boolean
… String	string
… DateTime	dateTime
… AnyType	anyType

Příklad **ArrayOfAnyType** v XML dokumentu

```
<Value xsi:type="ArrayOfAnyType">
  <anyType xsi:type="xsd:byte">127</anyType>
  <anyType
    xsi:type="xsd:unsignedByte">255</anyType>
  <anyType
    xsi:type="xsd:string">Hello<>World</anyType>
  <anyType xsi:type="ArrayOfInt">
    <int>-2147483648</int>
    <int>0</int>
    <int>2147483647</int>
  </anyType>
  <anyType xsi:type="ArrayOfAnyType">
    <anyType xsi:type="xsd:byte">127</anyType>
    <anyType
      xsi:type="xsd:unsignedByte">255</anyType>
    <anyType
      xsi:type="xsd:string">Hello World</anyType>
    <anyType xsi:type="ArrayOfInt">
      <int>-2147483648</int>
      <int>0</int>
      <int>2147483647</int>
    </anyType>
  </anyType>
</Value>
```

- `xsi` je asociováno s URI: "http://www.w3.org/2001/XMLSchema-instance"
- `xsd` je asociováno s URI: "http://www.w3.org/2001/XMLSchema"

Definice ItemValue ve WSDL (1/3)

```
<s:complexType name="ItemValue">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="DiagnosticInfo“
      type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Value /> "
    <s:element minOccurs="0" maxOccurs="1" name="Quality“
      type="s0:OPCQuality" />
  </s:sequence>
  <s:attribute name="ValueTypeQualifier" type="s:QName" use="optional" />
  <s:attribute name="ItemPath" type="s:string" />
  <s:attribute name="ItemName" type="s:string" />
  <s:attribute name="ClientItemHandle" type="s:string" />
  <s:attribute name="Timestamp" type="s:dateTime" />
  <s:attribute name="ResultID" type="s:Qname" />
</s:complexType>

<s:complexType name="OPCQuality">
  <s:attribute default="good" name="QualityField" type="s0:qualityBits" />
  <s:attribute default="none" name="LimitField" type="s0:limitBits" />
  <s:attribute default="0" name="VendorField" type="s:unsignedByte" />
</s:complexType>
```

Definice ItemValue ve WSDL (2/3)

```
<s:simpleType name="qualityBits">
  <s:restriction base="s:string">
    <s:enumeration value="bad" />
    <s:enumeration value="badConfigurationError" />
    <s:enumeration value="badNotConnected" />
    <s:enumeration value="badDeviceFailure" />
    <s:enumeration value="badSensorFailure" />
    <s:enumeration value="badLastKnownValue" />
    <s:enumeration value="badCommFailure" />
    <s:enumeration value="badOutOfService" />
    <s:enumeration value="badWaitingForInitialData" />
    <s:enumeration value="uncertain" />
    <s:enumeration value="uncertainLastUsableValue" />
    <s:enumeration value="uncertainSensorNotAccurate" />
    <s:enumeration value="uncertainEUExceeded" />
    <s:enumeration value="uncertainSubNormal" />
    <s:enumeration value="good" />
    <s:enumeration value="goodLocalOverride" />
  </s:restriction>
</s:simpleType>
```

29

Definice ItemValue ve WSDL (3/3)

```
<s:simpleType name="limitBits">
  <s:restriction base="s:string">
    <s:enumeration value="none" />
    <s:enumeration value="low" />
    <s:enumeration value="high" />
    <s:enumeration value="constant" />
  </s:restriction>
</s:simpleType>
```

- Element Value (5. řádek definice na první straně) může mít jednoduchý typ nebo být typem pole
- Z uvedené definice **`ItemValue`** je patrná (záměrná) podobnost mezi položkou v této specifikaci a ve specifikaci OPC DA 3.0 (viz přednášku o standardech OPC)

30