## Adresy logiczne - numery IP

**Adres IP** to logiczny adres numeryczny nadawany dla interfejsu sieciowego, grupie interfejsów bądź całej sieci komputerowej w protokole IP, służący identyfikacji elementów w warstwie trzeciej modelu OSI – w obrębie sieci lokalnej oraz poza nią – tzw. Adres publiczny.

- **IPv4** to 32 bitowa liczba zapisana sekwencyjnie w notacji kropkowo-dziesiętnej. Służy do precyzyjnej lokalizacji węzła w sieci. Jest on podzielony na cztery ośmiobitowe bloki – oktety. Dostępna pula adresowa w przypadku IPv4 to 2^32 (4 294 967 296).

- **IPv6** (ang. Internet Protocol version 6) – jest to kolejna wersja protokołu IP. Adres IPv6 to 128 bitowa liczba zapisana zwykle jako osiem 16-bitowych bloków w systemie heksadecymalnym oddzielonych dwukropkiem. Dostępna pula adresowa IPv6 to 2^128.

Adres IP zapewnia jednolity sposób identyfikacji maszyn w sieci Internetowej. Stanowi on adres logiczny, którego celem jest identyfikowanie adresata musi być on odwzorowany na adres sprzętowy w danej sieci fizycznej. Jeśli indentyfikuje on maszynę w bieżącej sieci, można dokonać odwzorowania. W przeciwnym razie datagram trafi do maszyny zwanej routerem działającej na granicy danej sieci fizycznej, która na podstawie adresu IP określi kolejnego pośrednika, któremu należy go przekazać.

Adres IP w stosowanym aktualnie powszechnie protokole IPv4 składa się z 4 oktetów (zapisywanych zwykle jako 4 liczby dziesiętne z przedziału 0-255 - łącznie z 32 bitów). Adres ten musi zawierać rozróżnienie adresu sieci i adresu maszyny w aktualnej sieci. W zależności od wielkości sieci - podział w sensie liczby bitów identyfikujących sieć i maszynę w tej sieci może być różny.

## Wyróżniamy 5 klas adresów:

- **Klasa A** - część sieci składa się z 8 bitów (I oktet), a część hosta z pozostałych 24 bitów (II, III, IV oktet). Pierwszy bit adresu zawsze jest ustawiony na 0. Maksymalnie można zaadresować:

  **$2^{24}$- 2 hosty = 16777214 hostów**

Przeznaczona dla największych sieci. Wartości pierwszego oktetu od 1 do 126. Pozostałe oktety identyfikują maszyny w sieci.

- **Klasa B** – część sieci składa się z 16 bitów (I, II oktet), a część hosta z pozostałych 16 bitów (III, IV oktet). Pierwsze 2 bity adresu zawsze są ustawione na 10. Maksymalnie można zaadresować:

  **$2^{16}$- 2 hosty = 65534 hostów**

  Przeznaczona dla średnich sieci. Wartości pierwszego oktetu od 128 do 191. Dwa pierwsze oktety przeznaczone na identyfikację sieci; dwa pozostałe - dla komputerów w tej sieci.

- **Klasa C** - część sieci składa się z 24 bitów (I, II, III oktet), a część hosta z pozostałych 8 bitów (IV oktet). Pierwsze 3 bity adresu zawsze są ustawione na 110. Maksymalnie można zaadresować:

  **$2^{8}$- 2 hosty = 254 hosty**

  przeznaczona dla małych sieci. Wartości pierwszego oktetu od 192 do 223. Tylko ostatni oktet identyfikuje maszyny w sieci.

- **Klasa D** - pierwsze 4 bity adresu zawsze są ustawione na 1110, przeznaczona do identyfikacji grup komputerów przy rozgłaszaniu (multicast). Wartości pierwszego oktetu od 224 do 239. 28 ostatnich bitów identyfikuje grupę.

- **Klasa E** - pierwszy 4 bity adresu zawsze są ustawione na 1111. zarezerwowana do celów eksperymentalnych. Wartości pierwszego oktetu od 240 do 255.

**Adresy prywatne**, to adresy wydzielone w każdej klasie adresów IP, które nie są przydzielane hostom w Internecie. Adresy takie najczęściej wykorzystuje się do adresowania w sieciach lokalnych. Ruch kierowany do sieci prywatnej nie jest przepuszczany przez routery. Aby sieci prywatne (budowane w oparciu o adresy prywatne) mogły łączyć się z sieciami publicznymi (np. Internetem), muszą wykorzystywać translację NAT lub Proxy Server.

## Zakresy adresów prywatnych IP:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255

Maszyny dysponujące takimi adresami mogą korzystać z zasobów Internetu dzięki zastosowaniu odpowiednich technik NAT (Network Address Translation). Dla podniesienia efektywności wykorzystania puli adresów w IPv4 wprowadzono specyfikację bez-klasowego (classless) podejścia do adresowania, w którym liczba bitów określających tzw. maskę sieci jest określana explicite – np. 192.9.205.22 /18

## Język JavaScript-Geneza

W 1996 r. organizacja ECMA rozpoczęła pracę nad specyfikacją języka JavaScript pozbawioną odniesień do środowiska interpretującego kod. Język opisany w standardzie ECMA-262 został nazwany ECMAScript. Od tego momentu nazwa JavaScript oznacza jedynie jeden z nadzbiorów języka ECMAScript. Pierwsza wersja standardu była bliska JavaScriptowi w wersji 1.1. Trzecia jako pierwsza rozszerzyła istniejące implementacje. Wersja ES4 została całkowicie porzucona przez przeglądarki internetowe, korzystał z niej jedynie język ActionScript dla platformy Flash.

Od 2015 roku znacząco zmienił się proces tworzenia i wydawania nowych wersji ECMAScript. Po wersji ES5 wydanej w 2009 pojawiała się wersja ES6 wydana w 2015 roku, a następna już w 2016 (ES7, a właściwie ES2016). Zmieniło się również oficjalne nazewnictwo – wersja początkowo nazywana ES6 to teraz ES2015, potem jest ES2016, ES2017 itd.

JavaScript zaprojektowano jako język skryptowy o luźnej kontroli typów. Oryginalnie powstał w roku 1995 dla przeglądarki Netscape, pierwotnie pod nazwą LiveScript, którą później zmieniono m.in. ze względów na nieuregulowane prawa do oryginalnej nazwy . Dostarczył on, niedostępnej wcześniej, podstawowej funkcjonalności skryptowej do ówczesnych przeglądarek. Należy zaznaczyć, że od początku był on planowany jako

rozszerzenie możliwości stron WWW, przystosowane do użycia przez nieprofesjonalistów. Toteż zrezygnowano z szeregu drugoplanowych konstrukcji języka, które skomplikowałyby jego opanowanie przez twórców.

Należy podkreślić, że wówczas nie istniał żaden generyczny interfejs programistyczny pozwalający językom programowania manipulować strukturą bieżącego dokumentu WWW oraz środowiskiem, w którym został on otwarty. Jedynie JavaScript został wyposażony w dostęp do tych zasobów - dlatego też, dla innych języków mających działać w środowisku dokumentu WWW musiał pełnić rolę pośrednika. Stąd też, ważnym pierwotnie zakładanym zastosowaniem miała być interakcja z apletami Javy. Ten fakt, oraz znaczne podobieństwo składniowe do języka Java należy uznać jako motywy wyboru nazwy JavaScript.

Następnie, został przyjęty jako otwarty standard i zyskał implementacje w przeglądarkach różnych producentów. Obecnie zaimplementowana (np. Firefox 3), wersja 1.7. Obok oryginalnej linii specyfikacji JavaScript mamy do czynienia z dialektem nazwanym JScript, implementowanym przez przeglądarki firmy Microsoft. Cechuje go inna numeracja wersji, choć funkcjonalność języka jest zasadniczo zbieżna. Np. wersji 1.5 JavaScript odpowiadają wersje 5.0 i 5.5 JScript).

Prace standaryzacyjne przeprowadzono pod patronatem European Computer Manufacturers Association (ECMA), stąd jeszcze jedna nazwa, pod którą występuje omawiany język: ECMAScript. Specyfikację opublikowano w roku 1997, zaś rok później stała się ona standardem ISO, którego aktualną obecnie wersją jest ISO/IEC 16262:2002 Information technology - ECMAScript language specification. Dojrzałość języka oraz fakt, że jest wspierany przez wszystkie nowoczesne przeglądarki, czynią zeń jedno z najpopularniejszych narzędzi zapewniania interakcyjności stron WWW.

## Wzmianka o wielowątkowości

***Serwlet*** *– klasa Javy działająca po stronie serwera WWW w modelu żądanie-odpowiedź, rozszerzająca jego możliwości. Uruchamiane są w bezpiecznym środowisku serwera aplikacji (np. GlassFish) albo kontenera webowego (np.*

*Apache Tomcat). Jako część platformy JEE, serwlety mają dostęp do całego API Javy.*

*Nazwa Serwlet powstała na wzór nazwy aplet, przez zastąpienie sylaby ap- sylabą serw-, wskazującą na wykonywanie programu po stronie serwera.*

*Rozszerza możliwości serwera o*

- *CGI w Javie, wzbogacone o biblioteki ułatwiające życie programiście (np. utrzymywanie sesji, wspólne zasoby serwletów, ciasteczka, obsługa GET/POST/PUT/HEAD/DELETE);*
- *serwlety mogą korzystać ze standardowych klas Javy (z VM), klas wchodzących w skład Servlet API: javax.servlet.* i javax.servlet.http.* oraz ewentualnie z dodatkowych bibliotek zainstalowanych na serwerze;*
- *serwlety nie mają interfejsu użytkownika, komunikują się z przeglądarką (za pośrednictwem serwera WWW wspierającego serwlety) za pomocą protokołu HTTP.*

Serwlety wykonują się w środowisku wielowątkowym, więc może się zdarzyć, że wiele kopii serwletu będzie działać naraz. Może być tak, że będzie jedna instancja serwletu, ale wiele wątków wykonujących metodę service. Jeżeli w serwlecie jest odwołanie do zasobu, który wymaga wyłączności to:

- trzeba zapewnić wykluczanie ręcznie, np. używając zmiennych/metod synchronized, albo
- zadeklarować, że serwlet implementuje interfejs SingleThreadModel, wtedy serwer uruchomi jednocześnie tylko jedną instancję metody service, np.

## XML i DTD; eXtensible Markup Language

XML to są pliki tekstowe, które tak samo wyglądają w każdym systemie, dzięki czemu możliwości opartego na XML współdziałania pomiędzy różnymi

platformami. Pozwala to komunikować się w wysoko niejednorodnych (heterogenicznych) środowiskach.

## Uzywane skróty:

- **PDL** (Page Description Language) - język opisu strony, profesjonalnych systemów składu (TeX, PostScript).
- **GML** (Generalized Markup Language). Wprowadzono atrybuty znaczników.
- **SGML** (Standard Generalized Markup Languege) - międzynarodowa norma (ISO 8879), dotycząca strukturalizacji dokumentów elektronicznych. Nie jest to język, ale sposób formalny definiowania języków znakowania, metajęzyk. Elementy dokumentu, z punktu widzenia SGML, pozostają w pewnych relacjach: następują po sobie lub jedne zawierają w sobie inne. Można wskazać elementy konieczne lub opcjonalne, unikatowe lub przetwarzalne. Elementy można także sparametryzować, przypisując im pewne atrybuty, w tym także referencje. Zastosowania: duże przedsięwzięcia wydawnicze, publikacje (dokumentacje techniczne, zbiory przepisów prawa, publikacje naukowe, encyklopedie).

## XML - charakterystyka

XML stanowi zbiór reguł składniowych umożliwiających stworzenie języków specjalnego zastosowania dla dowolnych dziedzin. Ma umożliwiać wymianę danych / dokumentów pomiędzy heterogenicznymi systemami komputerowymi, zwłaszcza w środowisku Internetu. W szerszym znaczeniu termin XML może oznaczać również:

- Technologie związane z przetwarzaniem dokumentów w składni XML,
- Specjalizowane języki oparte na XML.

W przeciwieństwie do HTML (i podobnie jak SGML), XML nie określa "słownictwa", tj. predefiniowanych nazw elementów z ich dozwoloną zawartością i atrybutami. Jest otwarty na definiowanie tego rodzaju reguł. Z

tego względu bywa nazywany metajęzykiem. Stanowi uproszczoną odmianę języka SGML.

W metajęzykach takich jak SGML czy XML, używamy w odniesieniu do zdefiniowanych na ich podstawie języków dziedzinowych pojęcia aplikacji:

- HTML = język dziedzinowy oparty na SGML (aplikacja SGML)
- XSLT,
- XMI,
- SVG i wiele innych = języki dziedzinowe oparte na XML (aplikacje XML)

## Przykładowy dokument XML

```
<?xml version="1.0"?>
<store>
  <name>SuperSklep</name>
  <location city="New York" state="NY"
country="USA"/>
  <products>
    <product id="1" category="books">
      <name>Władca Pierścieni</name>
      <author>J.R.R. Tolkien</author>
      <year>1954</year>
      <genre>Fantasy</genre>
      <price>29.99</price>
    </product>
    <product id="2" category="movies">
      <name>Ojciec Chrzestny</name>
      <director>Francis Ford
Coppola</director>
      <year>1972</year>
      <genre>Gangsterski</genre>
      <price>19.99</price>
    </product>
    <product id="3" category="music">
      <name>Thriller</name>
      <artist>Michael Jackson</artist>
      <year>1982</year>
      <genre>Pop</genre>
      <price>9.99</price>
    </product>
  </products>
  <employees>
    <employee id="1">
      <name>John Smith</name>
      <position>Manager</position>
      <phone>555-555-5555</phone>
    </employee>
    <employee id="2">
      <name>Jane Doe</name>
      <position>Assistant Manager</position>
      <phone>555-555-5556</phone>
    </employee>
  </employees>
  <sales>
    <sale date="2022-01-01">
      <product id="1">
        <quantity>10</quantity>
        <total>299.90</total>
      </product>
      <product id="3">
        <quantity>5</quantity>
        <total>49.95</total>
      </product>
    </sale>
    <sale date="2022-01-02">
      <product id="2">
        <quantity>3</quantity>
        <total>59.97</total>
      </product>
      <product id="3">
        <quantity>8</quantity>
        <total>79.92</total>
      </product>
    </sale>
  </sales>
</store>
```

**Strukturę logiczną** dokumentu XML wyznaczają znaczniki, określając hierarchię elementów. Oprócz tego dokument XML posiada **strukturę fizyczną**. Jej przetworzenie i odwzorowanie dokumentu w pamięci pozwoli odtworzyć strukturę logiczną. Rozróżnienie to jest istotne, gdyż logicznie pojedynczy dokument XML może się składać z oddzielnych fizycznie elementów. W strukturze fizycznej dokument stanowi złożenie jednostek zwanych **encjami** (*XML entities*).

## Rodzaje zawartości elementowej

W zależności od budowy, wyróżniamy następujące rodzaje elementów XML:

- Elementy puste (pojedynczy znacznik zakończony "/").
- Elementy zawierające tylko tekst.
- Elementy złożone, tj. Posiadające pod elementy.
- Elementy o zawartości mieszanej (tj. Zarówno tekst jak i pod elementy - mogą być na rożne sposoby przeplecione).

**DTD** (*document type definition*) – rodzaj dokumentu definiujący formalną strukturę dokumentów XML, HTML, XHTML lub innych pochodzących z rodziny SGML lub XML. Definicje DTD mogą być zawarte w pliku dokumentu, którego strukturę definiują, przeważnie jednak zapisane są w osobnym pliku tekstowym, co pozwala na zastosowanie tego samego DTD dla wielu dokumentów. DTD określa składnię konkretnej aplikacji XML lub SGML, np. XHTML, EAD, TEI lub innej, zdefiniowanej dla potrzeb użytkownika. Zazwyczaj DTD definiuje każdy dopuszczalny element dokumentu, jego zbiór atrybutów i dopuszczalne wartości. DTD określa także zagnieżdżanie i wymagalność poszczególnych elementów w dokumencie. W praktyce DTD przeważnie składa się z definicji ELEMENT i definicji ATTLIST.  W praktyce ze względu na małe możliwości obecnie DTD jest wypierane przez nowocześniejsze XML Schema, które posiada znacznie większe możliwości oraz nie wymaga stosowania dodatkowej, nie-XML-owej składni.

## Przykład bardzo prostego użycia DTD osadzonego wewnątrz dokumentu XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE osoba [
  <!ELEMENT osoba (imie, drugieImie, nazwisko)>
  <!ELEMENT imie (#PCDATA)>
  <!ELEMENT drugieImie (#PCDATA)>
  <!ELEMENT nazwisko (#PCDATA)>
]>
<osoba>
  <imie>Amadeusz</imie>
  <drugieImie>Zenon</drugieImie>
  <nazwisko>Kowalski</nazwisko>
</osoba>
```

Oprócz metody umieszczania DTD bezpośrednio w dokumencie XML można także stosować odwołanie zewnętrzne na dwa sposoby:

- Identyfikator systemowy.
- Identyfikator publiczny.

**eXtensible Markup Language** - Uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. To język znaczników i format pliku do przechowywania, przesyłania i rekonstrukcji dowolnych danych. Jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi (różnymi) systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu. XML jest standardem rekomendowanym oraz specyfikowanym przez organizację W3C. Jest najpopularniejszym obecnie uniwersalnym językiem przeznaczonym do reprezentowania danych.

## Scenariusz użycia JAXB

Budowę i wykorzystanie aplikacji opartej na JAXB można podsumować w postaci następujących kroków:

1. Projektujemy schemat dokumentu i specyfikujemy go w XML.
2. Generujemy kod źródłowy Javy z utworzonej definicji schematu za pomocą kompilatora wiązania.
3. Kompilujemy wygenerowany kod.
4. Implementujemy aplikację korzystającą z wygenerowanych klas zawartości. Może ona m.in. realizować następującą funkcjonalność:
    - Utworzenie struktury dokumentu XML zgodnej ze źródłowym schematem poprzez jej wczytanie w postaci kodu XML lub poprzez instancjonowanie wygenerowanych klas.
    - Odczyt i ewentualna manipulacja danych.
5. Ewentualna walidacja dokonanych modyfikacji.
    - Zapisanie struktury do dokumentu XML.
    - Ewentualna walidacja dokonanych modyfikacji.
    - Zapisanie struktury do dokumentu XML.

## Składniki technologi JAXB:

JAXB (Java Architecture for XML Binding) to technologia pozwalająca na mapowanie struktury dokumentów XML na obiekty Javy. Składa się ona z kilku elementów, które są niezbędne do przetwarzania danych XML:

- **Dokument XML** - jest to źródło danych dla kompilatora wiązania, który określa budowę przetwarzanego dokumentu XML.
- **Deklaracje wiązania** - to elementy, które precyzują sposób konstruowania nazw w obiektach Javy. Mogą one być umieszczone jako adnotacje w schemacie źródłowym lub w osobnym pliku.

- **Kompilator wiązania** - jest to narzędzie, które tworzy kod klas Javy na podstawie źródłowego XML Schema.
- **Implementacja Binding Framework** - dostarcza funkcjonalności odczytu, zapisu oraz walidacji danych XML.
- **Klasy zawartości** - generowane przez kompilator wiązania ze schematu, reprezentują zawartość dokumentu.
- **Aplikacja Javy** - korzysta z binding framework i z "klas zawartości" celem przetwarzania określonego XML.
- **Wejściowe i wyjściowe dokumenty XML** - są to dane przetwarzane przez aplikację.

Innymi słowy, JAXB pozwala na automatyczne mapowanie danych z dokumentów XML na obiekty Javy i odwrotnie. Dzięki temu, programiści mogą uniknąć ręcznego parsowania dokumentów XML i przekształcania ich na obiekty Javy. Zamiast tego, mogą skorzystać z klas generowanych przez kompilator wiązania i korzystać z nich w swoim kodzie.

Kompilator wiązania generuje klasy na podstawie schematu XML, który definiuje strukturę dokumentu. Klasy te reprezentują elementy i atrybuty z dokumentu XML i zawierają metody do odczytu i zapisu danych. Implementacja Binding Framework udostępnia interfejsy i klasy, które pozwalają na odczyt i zapis danych za pomocą klas generowanych przez kompilator wiązania.

Aplikacja Javy może korzystać z tych klas i implementacji frameworku, aby przetwarzać dane z wejściowych i wyjściowych dokumentów XML. Dzięki temu, programiści mogą skupić się na logicznej części swojego kodu, a nie na ręcznym parsowaniu danych XML.

JAXB jest bardzo przydatny w sytuacjach, gdzie aplikacja musi przetwarzać duże ilości danych z formatu XML. Może być również używany do tworzenia web service'ów, ponieważ pozwala na automatyczne mapowanie danych między formatem XML a obiektami Javy, co znacznie ułatwia proces tworzenia i konsumowania usług sieciowych. JAXB jest również elastyczny, ponieważ pozwala na ręczne dostosowywanie generowanych klas do specyficznych

potrzeb projektu. Programiści mogą edytować schemat XML, aby dostosować go do specyficznych wymagań, a następnie ponownie skompilować go, aby uzyskać nowe klasy Javy. Ogólnie rzecz biorąc, JAXB to potężne narzędzie, które pozwala na łatwe przetwarzanie danych XML w aplikacjach Javy. Pozwala na automatyzację procesu parsowania i mapowania danych, co zwiększa efektywność programistów i pozwala im skupić się na innych aspektach swojego kodu.