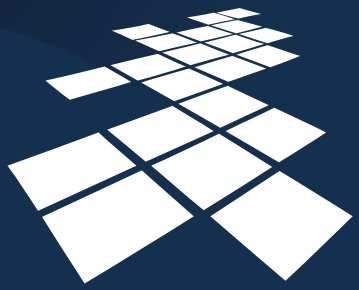
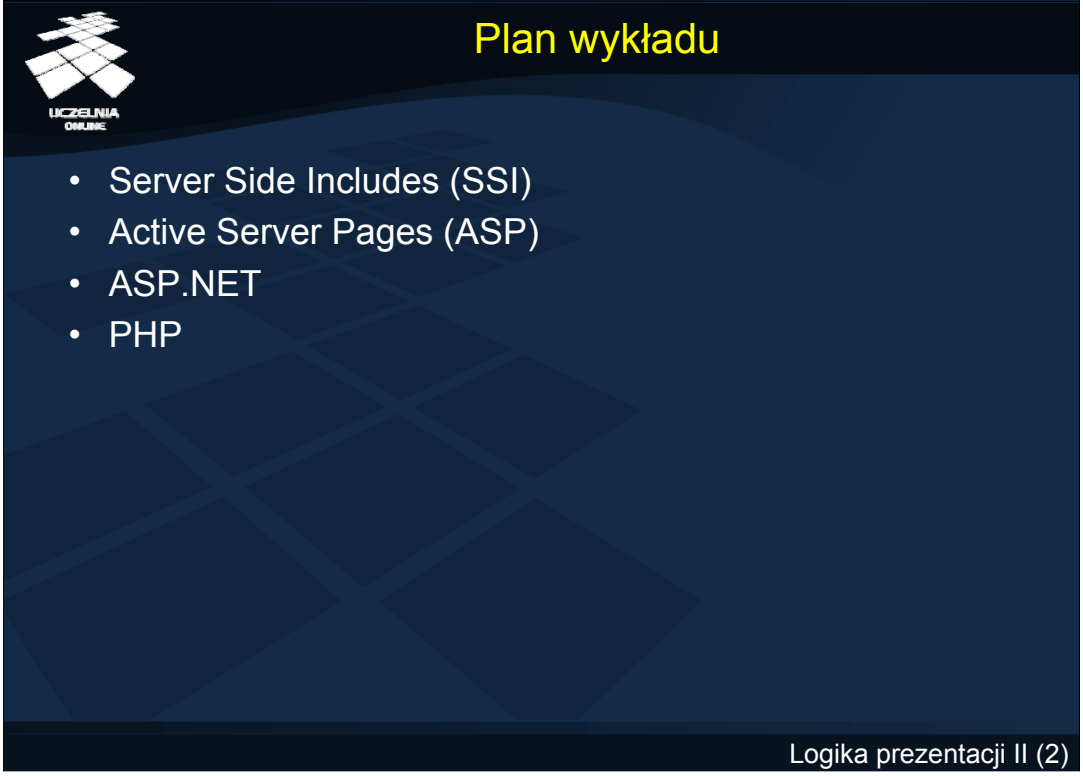
Logika prezentacji II

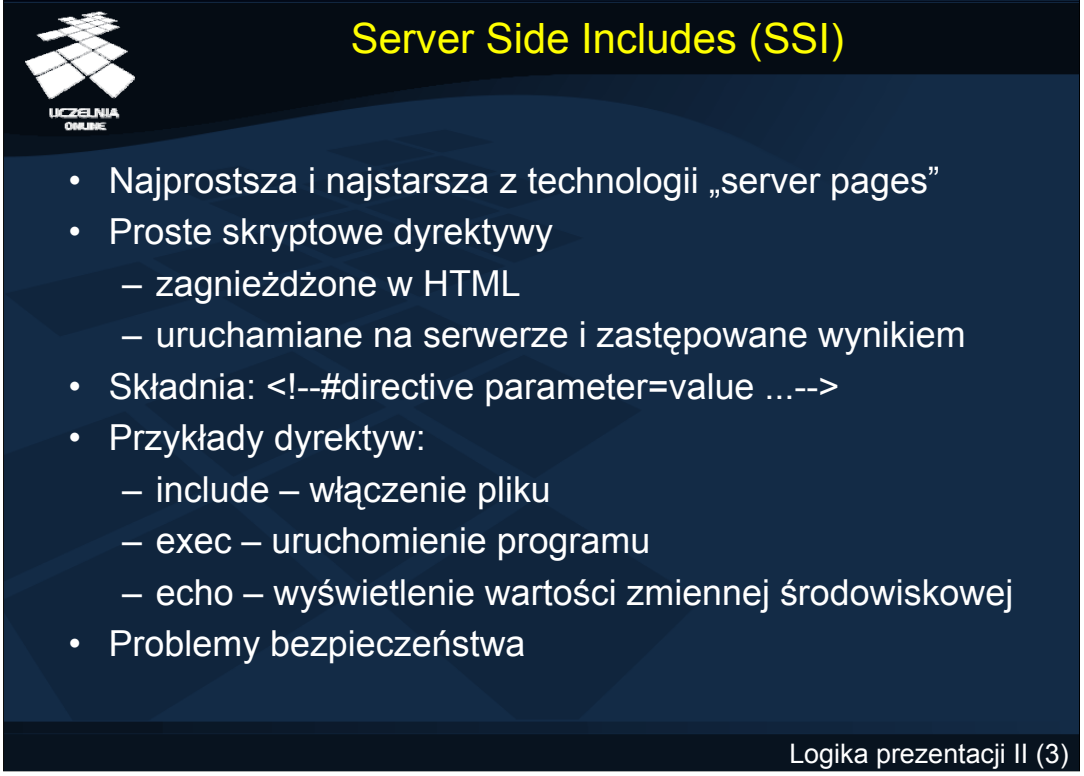
Wykład prowadzi:
Marek Wojciechowski



UCZELNIA
ONLINE



Celem wykładu jest przedstawienie najważniejszych technologii szablonów (server pages) wykorzystywanych w aplikacjach WWW. W ramach wykładu najpierw omówione będą krótko starsze technologie Server Side Includes (SSI) oraz Active Server Pages (ASP), a następnie bardziej szczegółowo następca ASP, czyli ASP.NET, oraz PHP. Technologie związane z językiem Java tj. JavaServer Pages (JSP) i Velocity będą stanowiły temat następnego wykładu.



Najprostszą i najstarszą technologią typu „server pages” tworzenia dynamicznych stron WWW jest Server Side Includes (SSI). Jak nazwa wskazuje, technologia ta pozwala na włączanie we wskazanych miejscach pliku źródłowego HTML informacji zewnętrznych. Włączane mogą być statyczne pliki z zawartością HTML, wyniki działania programów i wartości zmiennych środowiskowych serwera. Włączanie zawartości odbywa się po stronie serwera, dynamicznie dla każdego żądania. Dzięki temu, w odpowiedzi na kolejne żądania serwer wysyła dokumenty z różną włączoną zawartością, a do klienta trafia czysty kod HTML bez żadnych konstrukcji SSI.

W SSI zawartość jest włączana poprzez proste dyrektywy, stąd technologia ta nadaje się do prostych zastosowań. Składnia dyrektyw to:
<!--#directive parameter=value parameter=value-->,
gdzie „directive” to jedna z nazw dyrektyw.

Najpopularniejsze dyrektywy to:

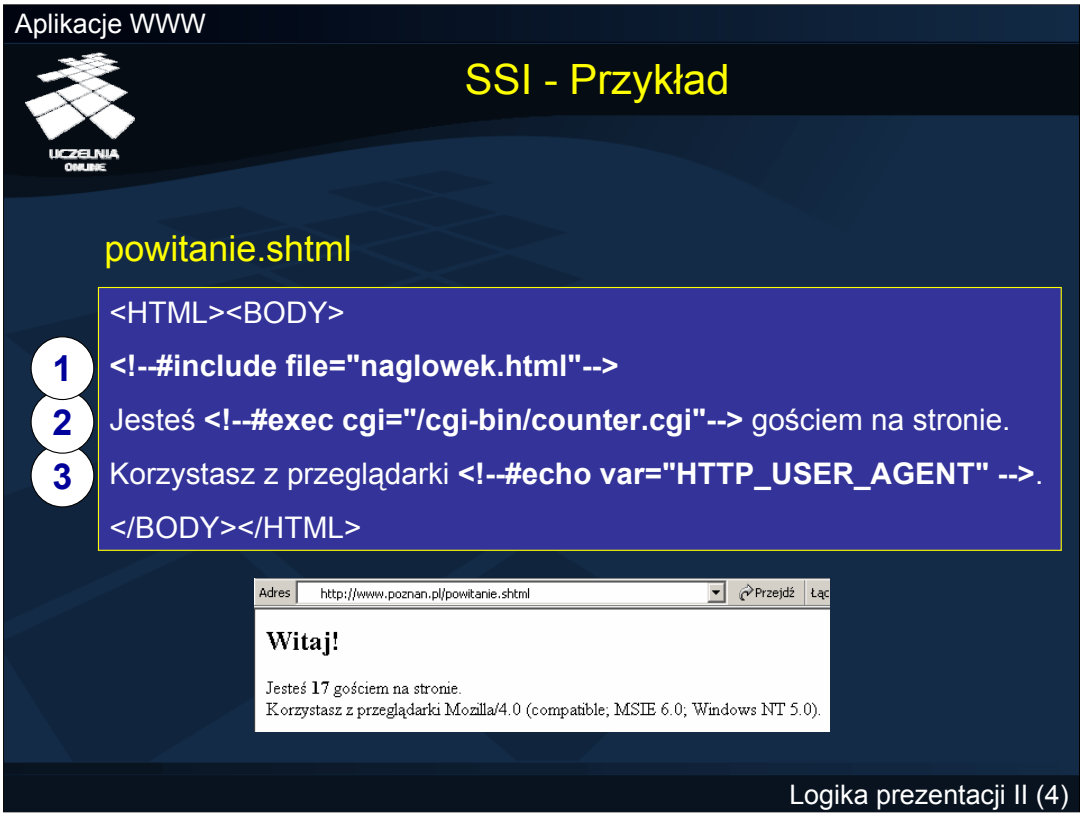
include – włączenie pliku z katalogu fizycznego lub wirtualnego,

exec – uruchomienie programu (CGI lub komendy systemowej) i włączenie wyniku działania programu,

echo – wyświetlenie wartości zmiennej środowiskowej.

W celu uruchamiania SSI na serwerze HTTP, należy go tak skonfigurować by wiedział, które pliki HTML zawierają dyrektywy SSI i w związku z tym powinny być parsowane po stronie serwera. Zwyczajowo pliki HTML z dyrektywami SSI są wyróżnione rozszerzeniem *.shtml.

Często w domyślnej konfiguracji serwerów HTTP uruchamianie SSI jest wyłączone, ze względu na zagrożenia bezpieczeństwa związane z SSI, a szczególnie dyrektywą „exec”.



Na slajdzie zamieszczono kod źródłowy przykładowego dokumentu wykorzystującego SSI. Dokument znajduje się w pliku powitanie.shtml i odwołuje się dyrektywami SSI do pliku HTML naglowek.html zawierającego tekst powitalny i programu CGI counter.cgi zliczającego kolejne uruchomienia i generującego dokument zawierający liczbę uruchomień.

Oto opis trzech dyrektyw SSI wykorzystanych w przykładzie:

1. Dyrektywa włączająca nagłówek z pliku naglowek.html (<H2>Witaj!</H2>).

2. Dyrektywa uruchamiająca program CGI i włączająca tekst przez niego wygenerowany (<B>17</B>).

3. Dyrektywa wstawiająca do dokumentu wartość zmiennej środowiskowej HTTP_USER_AGENT, zawierającej informację o typie przeglądarki, z której przyszło żądanie użytkownika (Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0))

Active Server Pages (ASP)

- Technologia Microsoft dla serwera IIS
- Oparta o interpretowane po stronie serwera wstawki skryptowe dynamicznie generujące HTML
- Domyślnie skrypty w VBScript, alternatywą JScript
- Skrypty odwołują się do predefiniowanych obiektów, reprezentujących funkcjonalność dynamicznych stron WWW
- ASP wprowadziło składnię dla „server pages”:
  - <% ... %>, <%= ... %>, <%@ ... %>
- Następcą ASP jest ASP.NET

Active Server Pages (ASP) to technologia firmy Microsoft stanowiąca rozszerzenie serwera HTTP Internet Information Services (IIS). Technologia ASP jest oparta o interpretowane po stronie serwera wstawki skryptowe zagnieżdżone w statycznym kodzie HTML, dynamicznie generujące zmienny kod HTML. Domyślnie używanym językiem skryptowym jest VBScript (wersja skryptowa języka Visual Basic). Możliwe jest wykorzystanie innych języków skryptowych (w ramach technologii Active Scripting), z których najpoważniejszą alternatywę dla VBScript stanowi JScript. Wstawki programowe zagnieżdżone w kodzie odwołują się do predefiniowanych obiektów Application, ASPError, Request, Response, Server i Session, reprezentujących funkcjonalność przydatną w tworzeniu dynamicznych stron WWW.

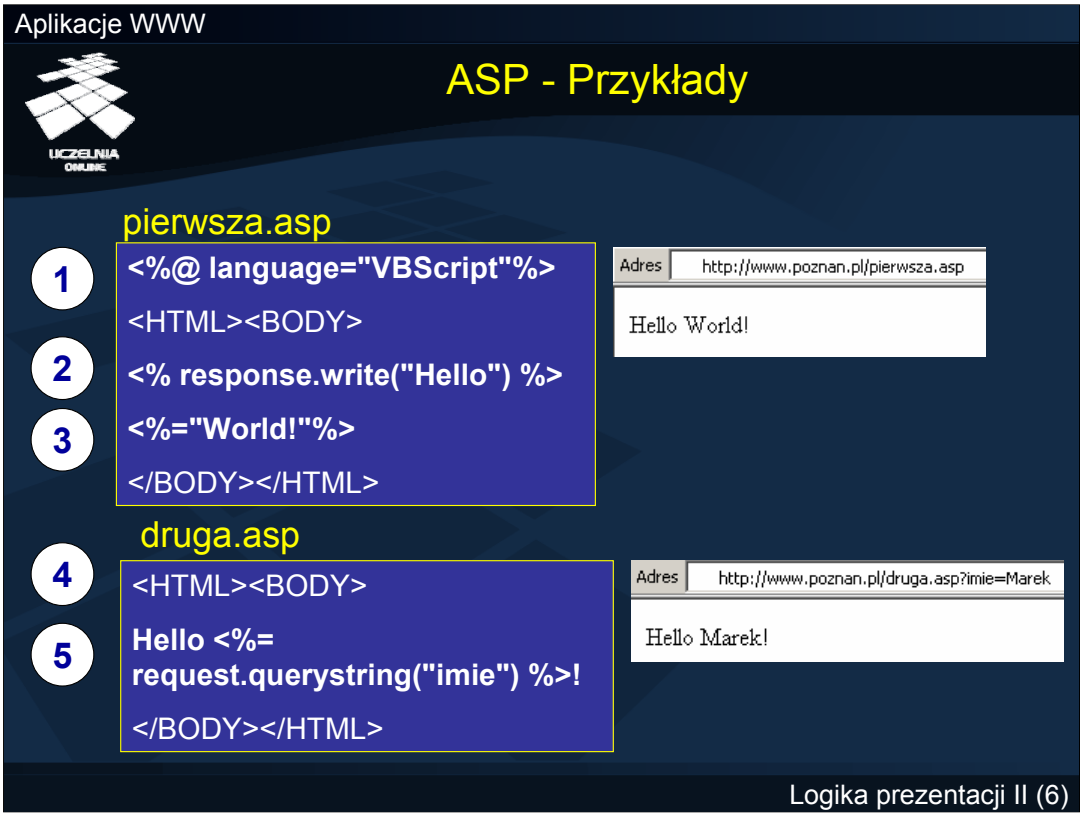
Technologia ASP wprowadziła znaczniki do zagnieżdżania fragmentów kodu programu, które później zostały wykorzystane w innych technologiach „server pages”, wzorujących się w pewnym stopniu na ASP. Te znaczniki to:

<% ... %> - do zagnieżdżania dowolnego kodu w języku programowania,

<%= ... %> - do zwrócenia wartości wyrażenia i wstawienia wyniku w dokumencie,

<%@ ... %> - dla różnych dyrektyw, np. do specyfikowania globalnych ustawień dla strony.

Wraz z pojawieniem się platformy Microsoft .NET, technologia ASP została od podstaw przepisana na tę platformę i jest obecnie dostępna jako ASP.NET, a „klasyczne” ASP nie jest dalej rozwijane. ASP.NET posiada szereg zalet w porównaniu z „klasycznym” ASP i powinno być wykorzystywane w nowych projektach.



Slajd przedstawia dwie proste przykładowe strony w „klasycznym” ASP (kod i efekt działania w przeglądarce). Zwyczajowe rozszerzenie nazwy pliku dla stron ASP to „.asp”. Wyróżnione konstrukcje ASP mają następujące znaczenie:

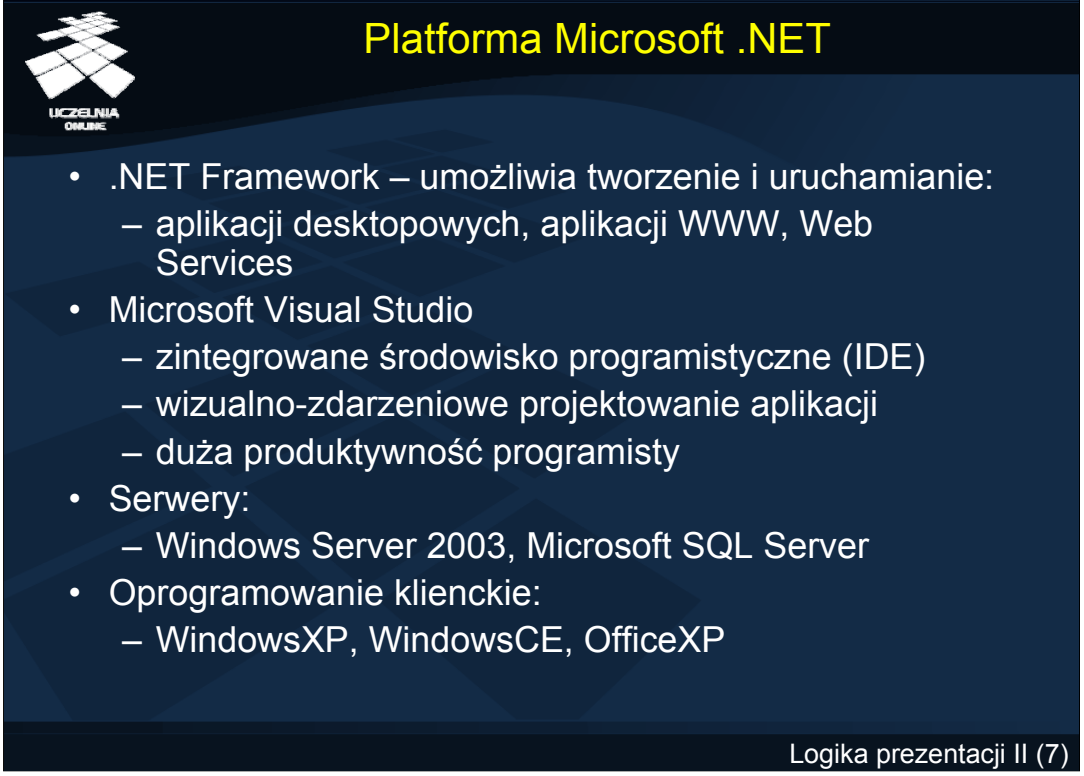
1. Dyrektywa informująca, że językiem skryptowym wykorzystywanym w zagnieżdżonych fragmentach kodu do przetworzenia po stronie serwera jest VBScript.

2. Wyświetlenie tekstu za pomocą metody write() predefiniowanego obiektu Response, reprezentującego odpowiedź HTTP.

3. Wyświetlenie tekstu za pomocą znacznika ASP, zwracającego wartość wyrażenia. W tym wypadku wyrażeniem jest literał tekstowy.

4. Kod drugiej przykładowej strony nie rozpoczyna się od dyrektywy wskazującej wybrany język skryptowy. Oznacza to, że podobnie jak dla pierwszej przykładowej strony jest to VBScript, gdyż jest on językiem domyślnym.

5. W środku statycznego tekstu wklejana jest wartość wyrażenia, które odczytuje wartość parametru „imie” przekazanego w adresie wywołania (parametr mógłby być w ten sposób przekazany z formularza HTML wywołującego stronę ASP metodą GET).



Jak wspomniano wcześniej, technologia ASP jest obecnie rozwijana pod nazwą ASP.NET jako element platformy Microsoft .NET. Platforma Microsoft .NET to szerokie pojęcie obejmujące oprogramowanie dla serwerów i stacji roboczych oraz narzędzia do tworzenia aplikacji. Elementy Microsoft .NET to:

.NET Framework – integralny komponent systemu Windows, umożliwiający tworzenie i uruchamianie aplikacji nowej generacji, w tym aplikacji desktopowych, aplikacji WWW i komponentów Web Services.

Narzędzia programistyczne – w tym przede wszystkim zintegrowane środowisko programistyczne (IDE) Microsoft Visual Studio, zorientowane na wizualno-zdarzeniowe projektowanie aplikacji, pozwalające uzyskać bardzo dużą produktywność programisty.

Serwery – Microsoft Windows Server 2003, Microsoft SQL Server i Microsoft BizTalk Server, do uruchamiania i integrowania aplikacji.

Oprogramowanie klienckie – WindowsXP, WindowsCE, OfficeXP, dla użytkowników różnych urządzeń końcowych.

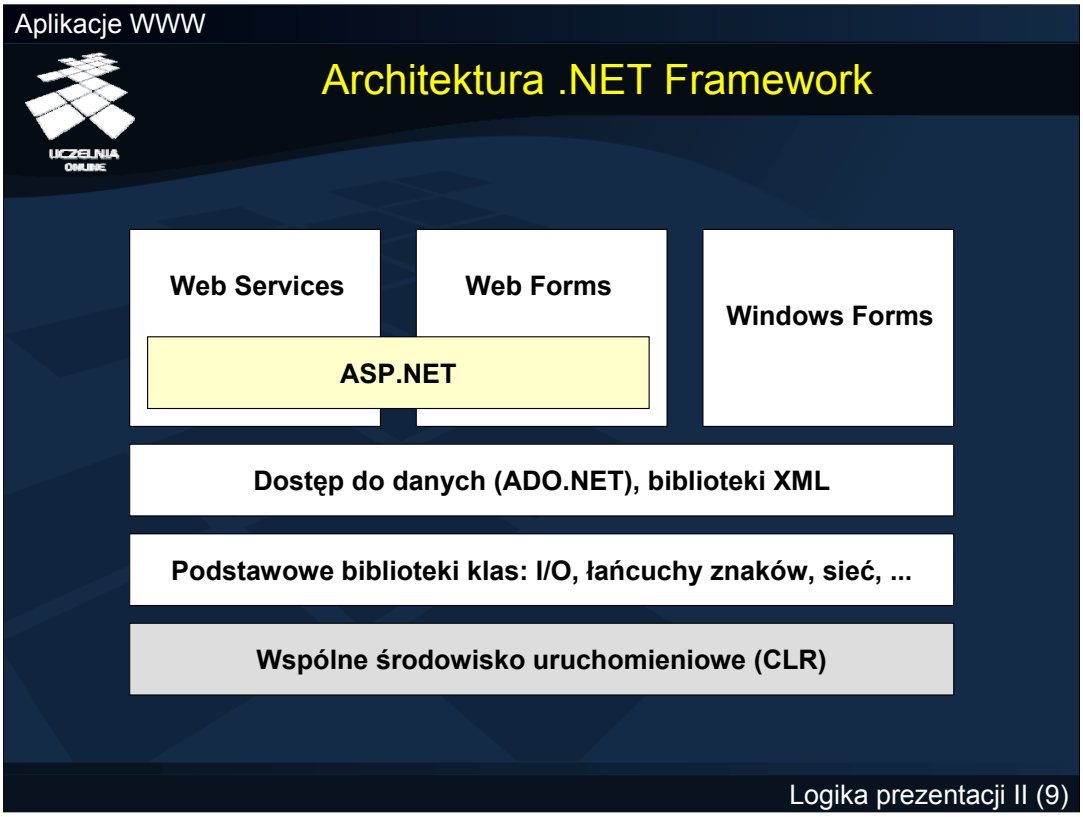
.NET Framework

- Umożliwia tworzenie i uruchamianie nowoczesnych aplikacji i Web Services
- Wspiera ponad 20 języków programowania
  - najważniejsze to: Visual Basic .NET, C#, C++, J#
- Podstawowe cele .NET Framework:
  - zwiększenie produktywności twórców aplikacji
  - łatwość tworzenia bezpiecznych i wydajnych aplikacji
  - łatwość instalacji aplikacji i administrowania nimi
- Składniki .NET Framework:
  - Common Language Runtime (CLR)
  - biblioteki klas

.NET Framework to kluczowy element platformy .NET, umożliwiający tworzenie i uruchamianie aplikacji nowej generacji, w tym aplikacji desktopowych, aplikacji WWW i komponentów Web Services. .NET Framework daje programistom dużą swobodę w zakresie wyboru języka programowania, wspierając ponad 20 języków, z których najważniejsze to Visual Basic .NET i C#, a w następnej kolejności C++ i J#.

.NET Framework to integralny komponent systemu Windows, opracowany z myślą o zwiększeniu produktywności twórców aplikacji, stanowiący gotową infrastrukturę rozwiązującą typowe problemy w implementacji aplikacji internetowych. .NET Framework ma na celu ułatwienie tworzenia bezpiecznych i wydajnych aplikacji, łatwych w instalacji i administrowaniu.

Składniki .NET Framework to wspólne środowisko uruchomieniowe – Common Language Runtime (CLR) i biblioteki klas spójne dla wszystkich języków.



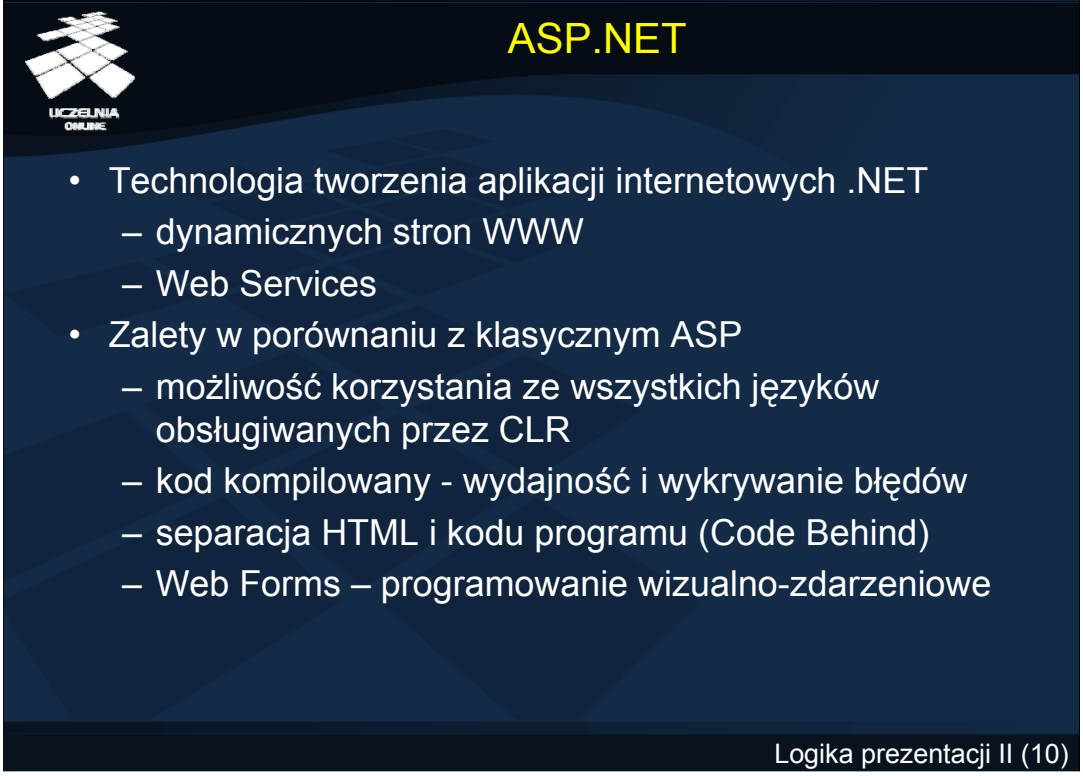
Wspólne środowisko uruchomieniowe (Common Language Runtime - CLR) to środowisko uruchomieniowe dla kodu tworzonego we wszystkich językach .NET. Kod źródłowy aplikacji, niezależnie od wybranego języka programowania i typu aplikacji, jest kompilowany do tego samego języka pośredniego o nazwie Microsoft Intermediate Language (MSIL). Dzięki takiemu rozwiązaniu możliwe jest wykorzystanie w ramach jednej aplikacji klas, których źródła zostały napisane w różnych językach. Przykładowo, klasa implementowana w języku C# może dziedziczyć z klasy zaimplementowanej w Visual Basic i odwrotnie.

Przy pierwszym uruchomieniu aplikacji, kod pośredni MSIL jest kompilowany do kodu maszynowego w trybie Just-In-Time (JIT) przez CLR. CLR zarządza wykonaniem kodu aplikacji .NET, odpowiadając za tworzenie obiektów, przydział pamięci i zwalnianie pamięci (garbage collection).

Klasy biblioteczne .NET Framework, które twórcy aplikacji mogą wykorzystywać w swoich programach dla platformy .NET można podzielić na kilka kategorii. Klasy podstawowe dostarczają standardową funkcjonalność, taką jak obsługa wejścia/wyjścia, operacje na łańcuchach znaków, zarządzanie bezpieczeństwem, obsługa komunikacji sieciowej, zarządzanie wątkami, zarządzanie tekstem, elementy interfejsu użytkownika. Klasy ADO.NET służą do komunikacji z bazami danych w aplikacjach .NET. Klasy XML oferują wsparcie dla przetwarzania danych XML w aplikacjach .NET.

Klasy biblioteczne .NET Framework zorganizowane są w przestrzenie nazw (ang. namespaces), które pozwalają na istnienie dwóch i więcej klas o tej samej nazwie, pod warunkiem, że są one przypisane do różnych przestrzeni nazw.

Typy aplikacji dla platformy .NET to Web Services, aplikacje WWW tworzone wg modelu Web Forms i aplikacje desktopowe tworzone wg modelu Windows Forms. Technologia ASP.NET wykorzystywana jest w Web Services i aplikacjach WWW. W ramach tego wykładu skupimy się na wykorzystaniu ASP.NET do implementacji aplikacji WWW.

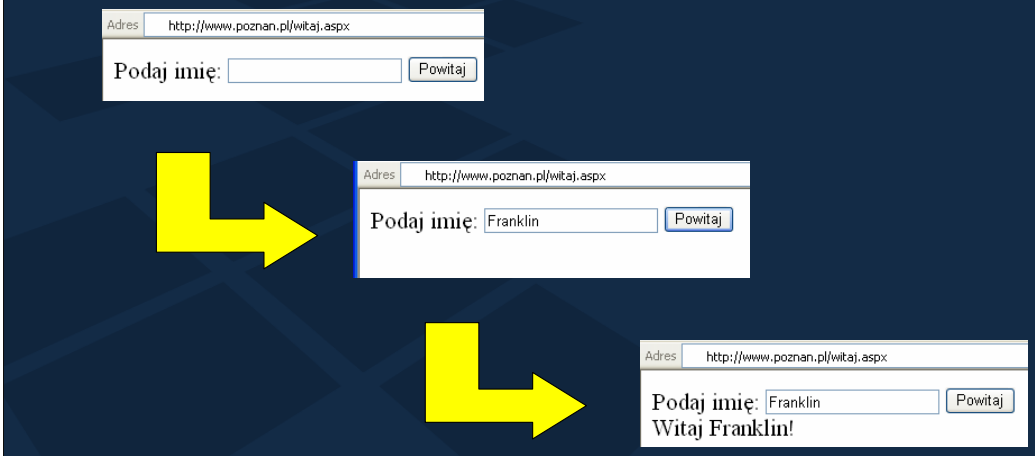


ASP.NET to technologia tworzenia dynamicznych stron internetowych dla platformy .NET i środowiska uruchomieniowego CLR. ASP.NET jest następcą klasycznego ASP w obszarze tworzenia dynamicznych stron WWW, ale umożliwia też tworzenie komponentów Web Services. ASP.NET w dużym stopniu, choć nie w pełni, zachowuje zgodność wstecz z klasycznym ASP. Niemniej, od strony technicznej technologia ASP.NET została od podstaw przebudowana i przewyższa klasyczne ASP pod wieloma względami.

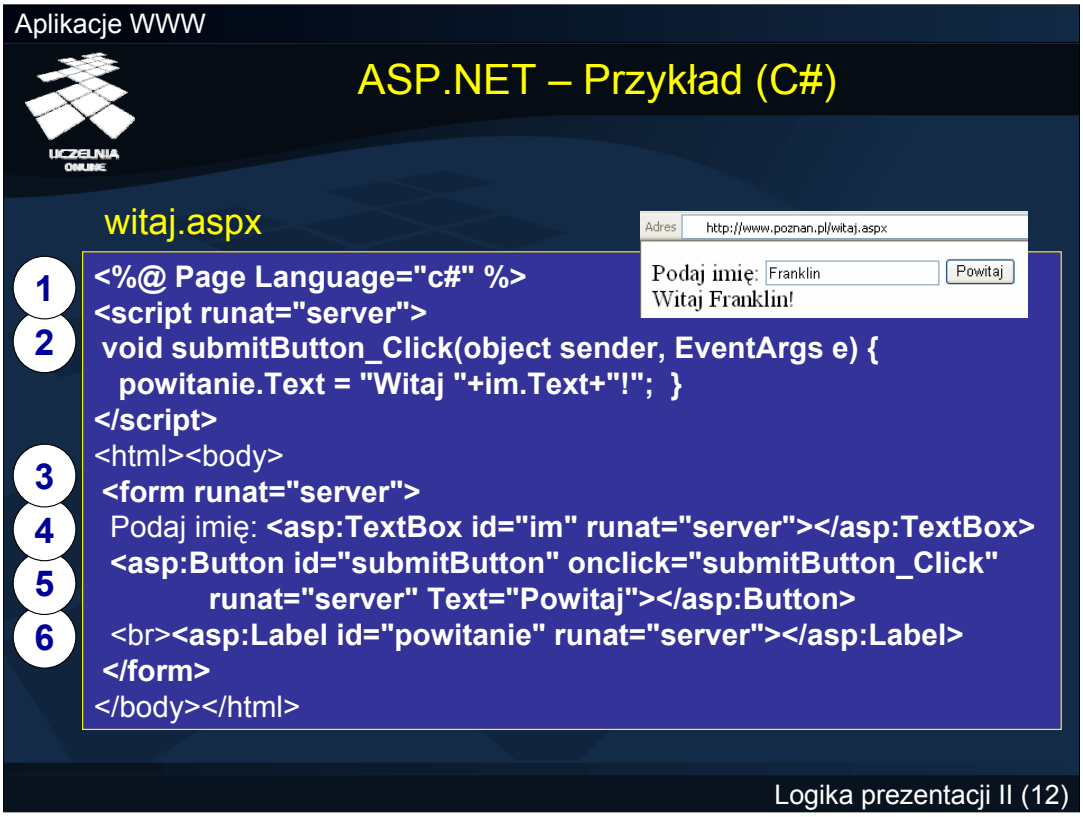
Po pierwsze, ASP.NET daje większy wybór języków programowania i co ważne są wśród nich pełnoprawne języki programowania jak Visual Basic i C#, a nie języki skryptowe jak VBScript i JScript w ASP. Po drugie, niezależnie od wyboru języka, kod aplikacji ASP.NET jest kompilowany (do postaci kodu pośredniego MSIL) co zwiększa wydajność aplikacji i ułatwia wykrywanie błędów. Po trzecie, ASP.NET umożliwia pełną separację kodu logiki aplikacji od kodu HTML dzięki architekturze Code Behind, polegającej na umieszczeniu kodu wykonywalnego w odrębnym pliku. Po czwarte, tworzenie aplikacji ASP.NET jest o wiele łatwiejsze niż w przypadku ASP, a także zdaniem wielu łatwiejsze niż np. na platformie Java EE, dzięki architekturze Web Forms i środowisku Visual Studio. Dzięki architekturze Web Forms tworzenie dynamicznych stron WWW bardzo przypomina tworzenie aplikacji desktopowych. Twórca aplikacji tworzy stronę ASP.NET poprzez dodawanie „kontrolek” do formularzy i wprowadzanie kodu do obsługi zdarzeń związanych z kontrolkami.

Szczególnie ciekawa na platformie .NET jest właśnie koncepcja Web Forms, która wyznaczyła nowy kierunek w tworzeniu interfejsu użytkownika w aplikacjach WWW. Dzięki Web Forms platforma .NET zyskała przewagę nad swym głównym konkurentem Java EE, pod względem łatwości tworzenia aplikacji. Obecnie, podobne rozwiązania w ramach platformy Java EE próbuje oferować JavaServer Faces.

ASP.NET – Przykład aplikacji



Tworzenie aplikacji w ASP.NET zostanie zilustrowane przykładem prostej strony. Na slajdzie pokazano planowany sposób działania aplikacji. Strona zawiera prostą formatkę umożliwiającą wprowadzenie imienia. Po zatwierdzeniu formularza przyciskiem „Powitaj”, w następnym wierszu pojawia się tekst powitalny zawierający wprowadzone imię.



Slajd przedstawia kod strony ASP.NET realizującej funkcjonalność opisaną na poprzednim slajdzie. W tym wypadku logika aplikacji została zaimplementowana w języku C#. Kod strony został zawarty w pliku witaj.aspx („.aspx" to zwyczajowe rozszerzenie dla stron ASP.NET). Ponieważ przykład jest bardzo prosty, nie została w nim wykorzystana technika pełnej separacji kodu HTML i logiki aplikacji (Code Behind). Kod HTML jak i C# są zawarte w tym samym pliku. Znaczenie wyróżnionych fragmentów kodu jest następujące:

1. Dyrektywa @Page wskazująca język w jakim została zaimplementowana logika aplikacji dla strony. W tym wypadku: C#.

2. Skrypt uruchamiany po stronie serwera (atrybut runat ma wartość „server"), zawierający metodę do obsługi zdarzenia naciśnięcia przycisku w formularzu. Zwyczajowo, nazwy procedur obsługi zdarzeń mają postać: „nazwaElementu_Zdarzenie()", w tym wypadku: submitButton_Click(). Obsługa zdarzenia w naszym przykładzie polega na umieszczeniu tekstu powitalnego w specjalnie do tego celu przygotowanym elemencie na stronie. Imię, które stanowi część tekstu powitalnego jest odczytywane z elementu formularza.

3. Atrybut runat z wartością „server" w znaczniku <FORM> oznacza, że formularz HTML ma być reprezentowany przez kontrolkę po stronie serwera. Kontrolka ta będzie stanowić kontener dla kontrolek reprezentujących elementy formularza.

4. Kontrolka typu Web Control reprezentująca pole tekstowe.

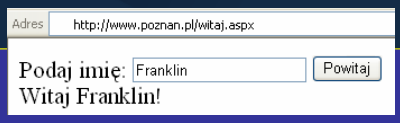
5. Kontrolka typu Web Control reprezentująca przycisk. Atrybut onclick wskazuje nazwę procedury obsługi zdarzenia naciśnięcia przycisku submitButton_Click().

6. Kontrolka typu Web Control reprezentująca etykietę tekstową. Kontrolka nie ma żadnej zawartości, przez co przy pierwszym odwołaniu do strony nie wygeneruje żadnego tekstu w wynikowym dokumencie HTML.

Każda z kontrolek na stronie posiada unikalny identyfikator (nazwę) w postaci atrybutu id. Poprzez ten atrybut kod logiki aplikacji odwołuje się właściwości kontrolek.

ASP.NET – Przykład (VB)

witaj.aspx



1

2

```
<%@ Page Language="vb" %>
<script runat="server">
 Sub submitButton_Click(ByVal sender As Object, _
                        ByVal e As System.EventArgs)
   powitanie.Text = "Witaj "+im.Text+"!„
 End Sub
</script>
<html><body>
 <form runat="server">
  Podaj imię: <asp:TextBox id="im" runat="server"></asp:TextBox>
  <asp:Button id="submitButton" onclick="submitButton_Click"
          runat="server" Text="Powitaj"></asp:Button>
  <br><asp:Label id="powitanie" runat="server"></asp:Label>
 </form>
</body></html>
```


Slajd przedstawia kod strony ASP.NET realizującej tę samą funkcjonalność co na poprzednim slajdzie. Tym razem niezmieniona logika aplikacji została zaimplementowana w języku Visual Basic. Treść strony zawarta w elemencie <HTML> nie uległa zmianie. Zmodyfikowana została (1) dyrektywa @Page, która tym razem wskazuje Visual Basic („vb”) jako język w jakim została zaimplementowana logika aplikacji dla strony. Zmianie uległa więc też (2) zawartość elementu <SCRIPT> z procedurą obsługi zdarzenia naciśnięcia przycisku w formularzu. Kod metody w Visual Basic robi dokładnie to samo co kod C# na poprzednim slajdzie.

Komentarza wymaga zachowanie się kontrolki pola tekstowego w formularzu. Po naciśnięciu przycisku strona jest wyświetlana ponownie, a wartość wprowadzona do pola zostaje odtworzona. Z punktu widzenia użytkownika stan formularza zostaje zachowany. Dzieje się tak dlatego, że architektura Web Forms zapewnia stanowy interfejs użytkownika bez konieczności oprogramowania takiego, zazwyczaj pożądanego, zachowania przez programistę. Temat ten zostanie rozwinięty w ramach jednego z kolejnych wykładów.

C# czy Visual Basic?

- C# (C sharp)
  - wprowadzony razem z .NET
  - nowoczesny i preferowany język dla .NET
  - atrakcyjny dla programistów C++ i Java
- Visual Basic .NET
  - Visual Basic rozwinięty do w pełni obiektowego języka programowania na poziomie C++ i C#
  - atrakcyjny dla programistów doświadczonych w tworzeniu graficznych aplikacji dla Windows w VB

Mimo iż .NET wspiera ponad 20 języków programowania, najczęściej wykorzystywane są dwa z nich: C# i Visual Basic .NET. Wybór języka jest kwestią preferencji programisty, wynikających z jego doświadczeń i upodobań, gdyż te same biblioteki klas .NET są dostępne z wszystkich języków .NET, ten sam jest też generowany kod wynikowy (MSIL).

Domyślnym językiem dla .NET jest C# (wymawiany „C sharp”), wprowadzony razem z .NET Framework i właśnie ten język będzie wykorzystywany w przykładach na kolejnych slajdach. C# jest określany jako następca C++ i jako taki wykazuje wiele podobieństw składniowych z językami C++ i Java, ale można w nim dostrzec również pewne wpływy Visual Basic, który jako język wysokiego poziomu znany jest z łatwości programowania i wysokiej produktywności programistów. C# z pewnością jest atrakcyjny dla programistów C++ i Java, ale muszą oni przyzwyczaić się różnic składniowych i specyfiki języka C#. Należy przy okazji wspomnieć, że z myślą o programistach Java, Microsoft zaoferował również język J#, jako wersję Javy dla .NET, kompilowaną do kodu pośredniego MSIL. J# nie jest jednak oparty na najświeższych specyfikacjach języka Java i ma znaczenie w zasadzie jako możliwość migracji aplikacji w języku Visual J++, oferowanym wcześniej przez Microsoft.

Visual Basic .NET to oferowany od lat przez Microsoft Visual Basic rozwinięty do postaci w pełni obiektowego języka programowania na poziomie C++ i C#. Visual Basic .NET jest atrakcyjnym wyborem dla programistów doświadczonych w tworzeniu graficznych aplikacji dla Windows w Visual Basic. Uważany jest za język, w którym łatwo w sposób produktywny się programuje.

Porównanie C# i Java

- Kwestionowana potrzeba opracowania C#
- Podobieństwa koncepcyjne, C# i Java to następcy C++
  - uproszczenie, kod pośredni
- W C# dostępne mechanizmy C++ niedostępne w Javie
  - np. przeciążanie operatorów
- Różnice w organizacji kodu
  - kilka klas w jednym pliku, koncepcja assembly
- Inne kwalifikatory dostępu („internal” dla assemblies)
- Drobne różnice składniowe we wspólnych konstrukcjach

C# wykazuje na tyle znaczące podobieństwo do języka Java, że nie brak opinii, iż decyzja Microsoft o opracowaniu nowego języka była decyzją polityczną, a nie wynikającą z potrzeby zaoferowania nowych możliwości. Trzeba jednak uczciwie przyznać, że C# zaoferował możliwości niedostępne w języku Java w momencie jego pojawienia się. Przykładem jest pętla „foreach” od początku dostępna w C#, a uwzględniona w Javie dopiero od wersji 5.0. Nie sposób oprzeć się wrażeniu, że stało się pod wpływem sukcesu C#.

C# podobnie jak Java stanowi uproszczenie w stosunku do C++ rezygnując ze wskaźników, wzorców i dziedziczenia wielobazowego. Podobnie jak Java, wprowadza interfejsy (z możliwością implementowania wielu interfejsów), oraz kompilację do kodu pośredniego (MSIL). Kolejnym podobieństwem są przestrzenie nazw (ang. namespaces) dla klas, które pozwalają w C# na użycie tej samej nazwy klasy w różnych przestrzeniach nazw, pokrywając się funkcjonalnością z pakietami w Javie.

C# zachował jednak wiele możliwości C++, niedostępnych w Javie np. przeciążanie operatorów i jawne wskazywanie metod wirtualnych. Ponadto, w C# typy, które w Javie są wyróżnionymi typami prostymi (np. int, double) są obiektami, ale mimo to są domyślnie przekazywane przez wartość z możliwością przekazywania przez referencję. C# oferuje struktury, które mogą być przekazywane przez wartość.

Organizacja kodu w C# jest inna niż w Javie. C# nie ogranicza możliwości definiowania kilku klas w jednym pliku źródłowym. Nie ma w C# pakietów, odpowiadające im w zakresie rozstrzygania unikalności nazw przestrzenie nazw nie wprowadzają nowego kwalifikatora widzialności (jak pakiety Java). Nową formą organizacji kodu są tzw. assemblies, widzialność składowych w ramach których określa specyficzny dla C# kwalifikator „internal”.

Istnieje też wiele drobnych różnic składniowych między C# i Javą w obrębie analogicznych konstrukcji składniowych, np. w C# składnia dziedziczenia jest w postaci z C++.

Rodzaje kontrolek ASP.NET

- HTML Controls
  - otoczki dla elementów HTML
  - umożliwiają programowy dostęp do elementów HTML
  - tworzone poprzez dodanie do znaczników HTML atrybutu runat=”server”
- Web Controls
  - preferowany, rozszerzalny zestaw łatwych do oprogramowania kontrolek
  - kontrolki proste i złożone
  - elementy interfejsu i kontrolki walidacyjne
  - tworzone przez specjalne znaczniki <asp: ...> zawsze z atrybutem runat=”server”

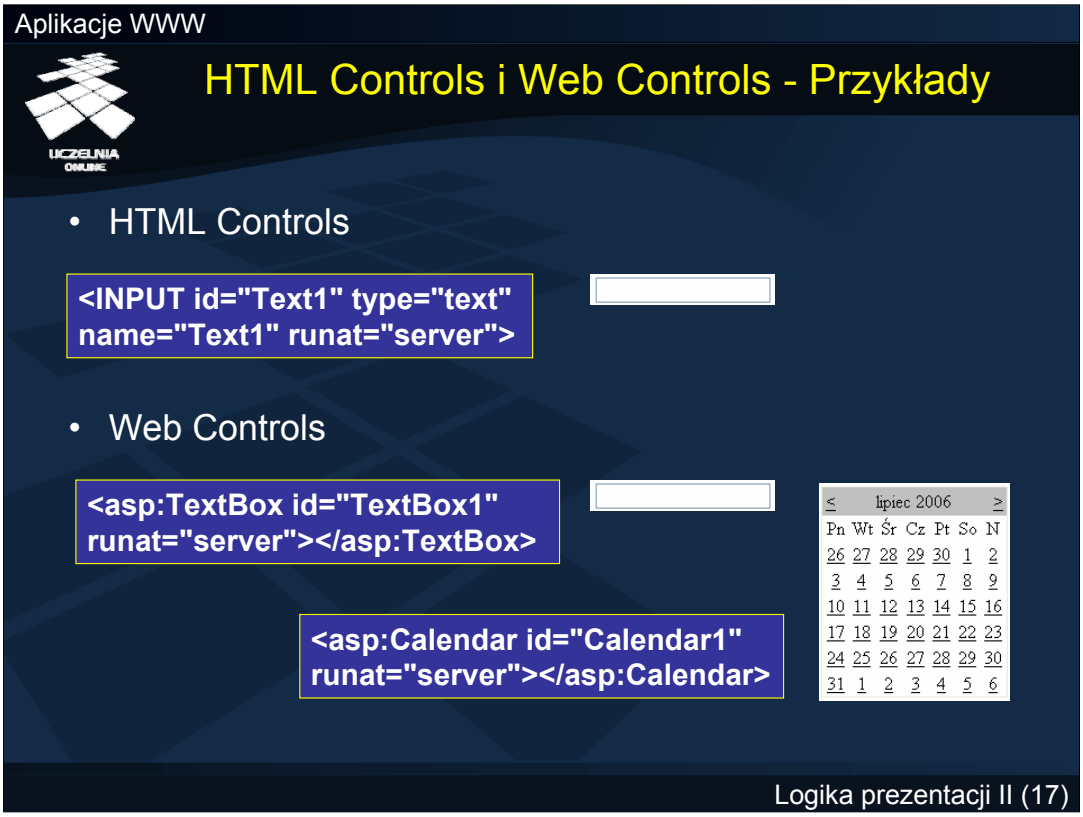
Interfejs użytkownika w ASP.NET tworzony jest z komponentów nazywanych kontrolkami. Podejście to szczególnie sprawdza się w wizualnym tworzeniu aplikacji w środowisku Visual Studio, gdyż kontrolki mogą być umieszczane na stronie metodą drag-and-drop, ich właściwości mogą być ustawiane poprzez paletę właściwości, itp. Kontrolki to elementy interfejsu pracujące po stronie serwera, reprezentowane wizualnie poprzez elementy HTML.

Dostępne kontrolki zostały podzielone na dwie grupy: HTML Controls i Web Controls. Oba zestawy kontrolek częściowo pokrywają się funkcjonalnością, mogą też współistnieć w ramach jednej strony.

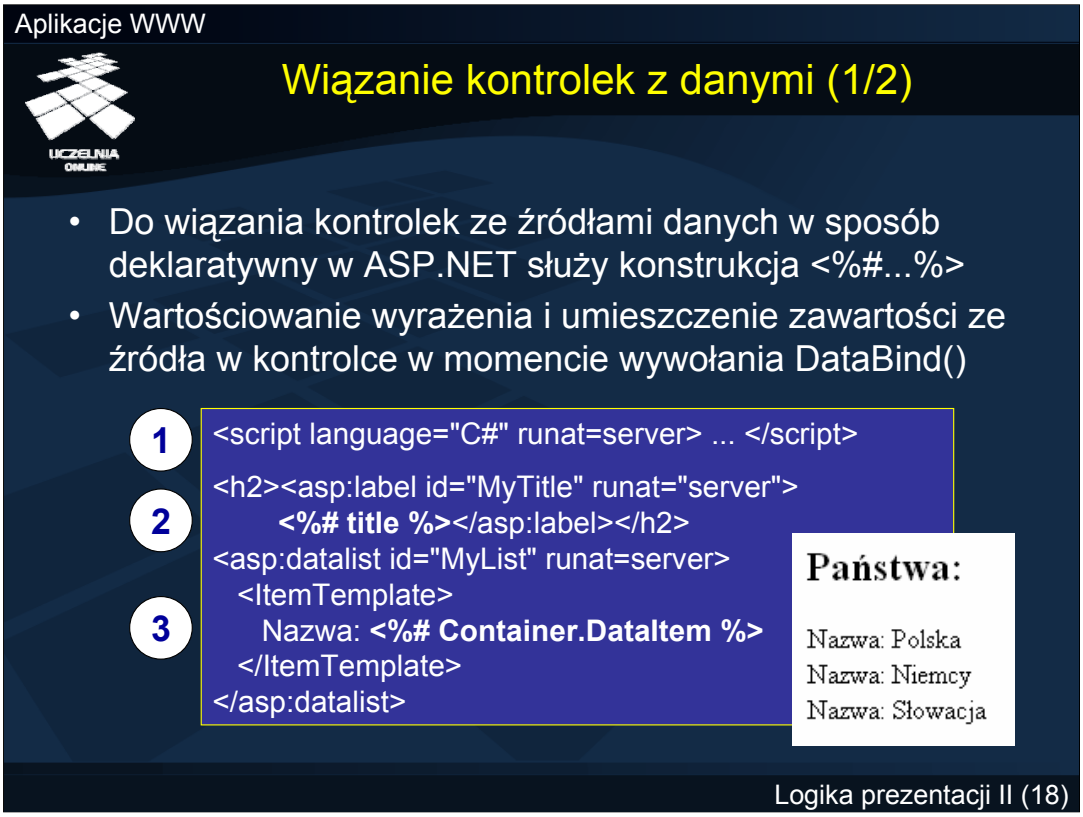
HTML Controls to „otoczki” dla elementów HTML, umożliwiające programowy dostęp do elementów HTML. Tworzone są poprzez dodanie do znaczników HTML atrybutu runat=”server”.

Web Controls to preferowany zestaw łatwych do oprogramowania kontrolek kluczowych dla koncepcji Web Forms tworzenia interfejsu użytkownika w aplikacjach WWW na wzór aplikacji desktopowych. Kontrolki Web Controls to zarówno proste kontrolki, wyświetlane podobnie jak HTML Controls jako pojedyncze elementy HTML, jak i złożone, wyświetlane w postaci wielu elementów HTML. Zestaw kontrolek Web Controls jest z założenia rozszerzalny. Twórcy aplikacji mogą w razie potrzeby implementować własne kontrolki, powstają też biblioteki kontrolek uzupełniające zestaw standardowy. Web Controls obejmują głównie elementy interfejsu, ale również kontrolki walidacyjne. Web Controls umieszczane są na stronie poprzez specjalne znaczniki <asp: ...>, zawsze z atrybutem runat=”server”.

Generalnie zalecane jest używanie Web Controls gdy tylko jest to możliwe, a HTML Controls do programowego dostępu do właściwości elementów HTML, dla których nie ma odpowiadających im Web Controls.



Slajd pokazuje przykłady kontrolek HTML Controls i Web Controls. Pierwsza kontrolka to pole tekstowe jako HTML Control - HtmlInputText. Do jej utworzenia wykorzystano standardowy znacznik HTML <INPUT>. O tym, że element ten jest kontrolką, świadczy atrybut runat=”server”. Element ma również nadaną wartość atrybutu id, poprzez którą będzie się do niego odwoływał kod aplikacji. Druga kontrolka to przykład prostej kontrolki Web Control - TextBox, wyświetlanej podobnie jak pierwsza z kontrolek w formie pola tekstowego. Trzecia kontrolka to kalendarz - Calendar, będący przykładem złożonej kontrolki Web Control.



ASP.NET umożliwia deklaratywne wiązanie kontrolek ze źródłami danych, takimi jak kolekcje wartości zbudowane programowo czy pobierane z bazy danych za pomocą konstrukcji składniowej <%#...%>

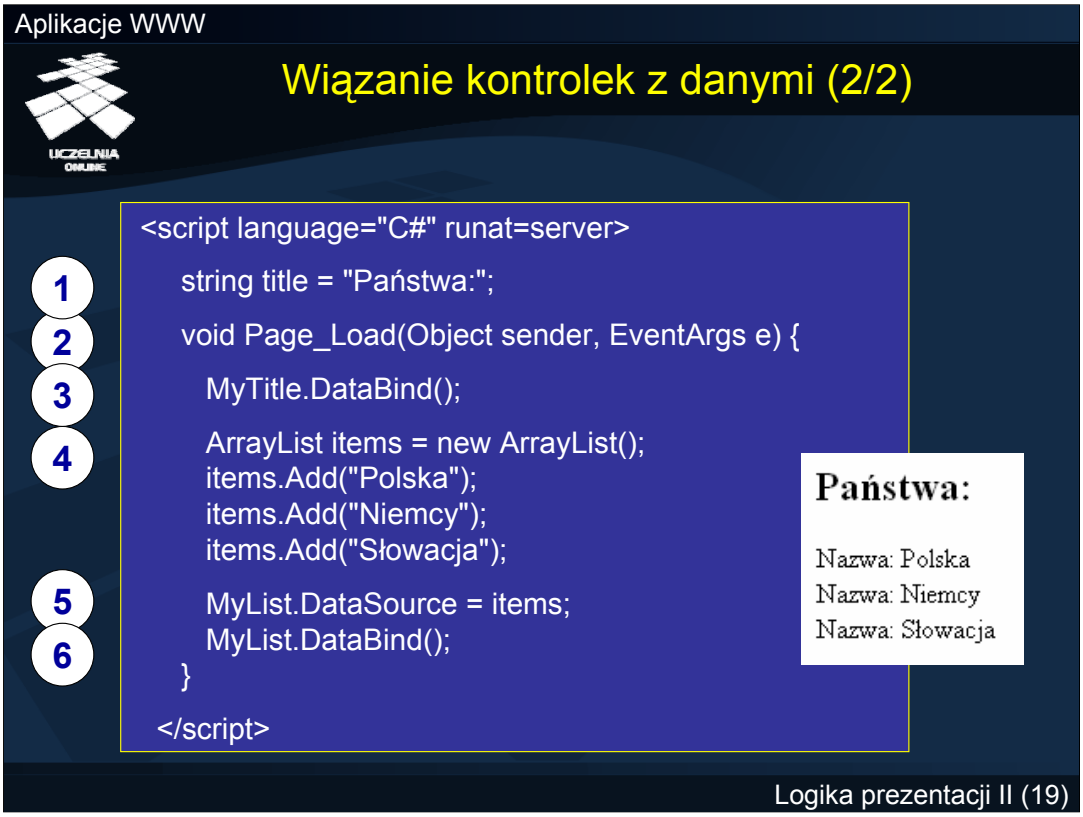
W odróżnieniu od konstrukcji <%=...%> wypisujących wartość wyrażenia, wartościowanie wyrażenia <%#...%> i umieszczenie zawartości ze źródła danych w kontrolce ma miejsce dopiero w momencie wywołania metody DataBind() na rzecz kontrolki lub kontenera, w którym jest zawarta (np. strony).

Slajd pokazuje prosty przykład wiązania danych dla etykiety tekstowej Label i nieco bardziej złożony dla kontrolki DataList, wyświetlającej kolekcję danych w postaci listy tekstowej. Znaczenie wyróżnionych fragmentów kodu jest następujące:

1. Kod przygotowujący dane źródłowe dla kontrolek na stronie i dokonujący wiązania kontrolek z danymi, przedstawiony w całości na następnym slajdzie.

2. Wyrażenie wiążące kontrolkę Label ze zmienną „title” w kodzie.

3. Znacznik <asp:datalist> reprezentujący kontrolkę DataList, prezentującą zawartość kolekcji danych w postaci listy tekstowej. Zagnieżdżony element <ItemTemplate> zawiera definicję struktury pojedynczego wiersza, w tym wypadku tekst „Nazwa:”, a po nim wyrażenie wiążące wiersz listy z elementem źródłowej kolekcji. Źródło danych jest programowo wskazane w kodzie (co zostanie pokazane na następnym slajdzie). Wyrażenie wiążące dane dla kontrolki DataList ma zawsze postać „Container.DataItem”.



Slajd pokazuje kod C# przygotowujący dane źródłowe dla kontrolek na stronie i dokonujący wiązania kontrolek z danymi. Znaczenie wyróżnionych fragmentów kodu jest następujące:

1. Źródłem danych dla etykiety jest składowa typu string, zadeklarowana na poziomie klasy strony.

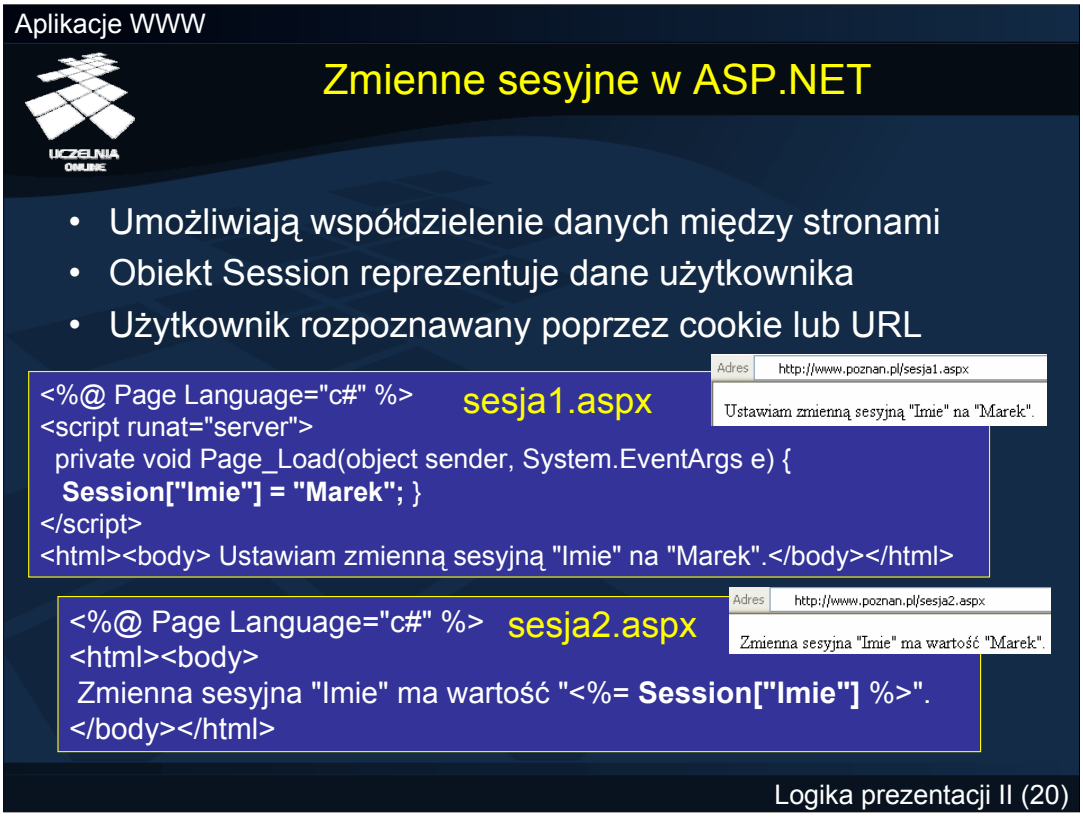
2. W naszym przykładzie, wiązanie danych nastąpi w momencie ładowania strony, gdyż wywołania metody DataBind() na rzecz kontrolek zostały umieszczone w metodzie Page_Load().

3. Operacja wiązania danych dla etykiety.

4. Przygotowanie kolekcji ArrayList, która będzie wykorzystana jako źródło danych dla kontrolki DataList.

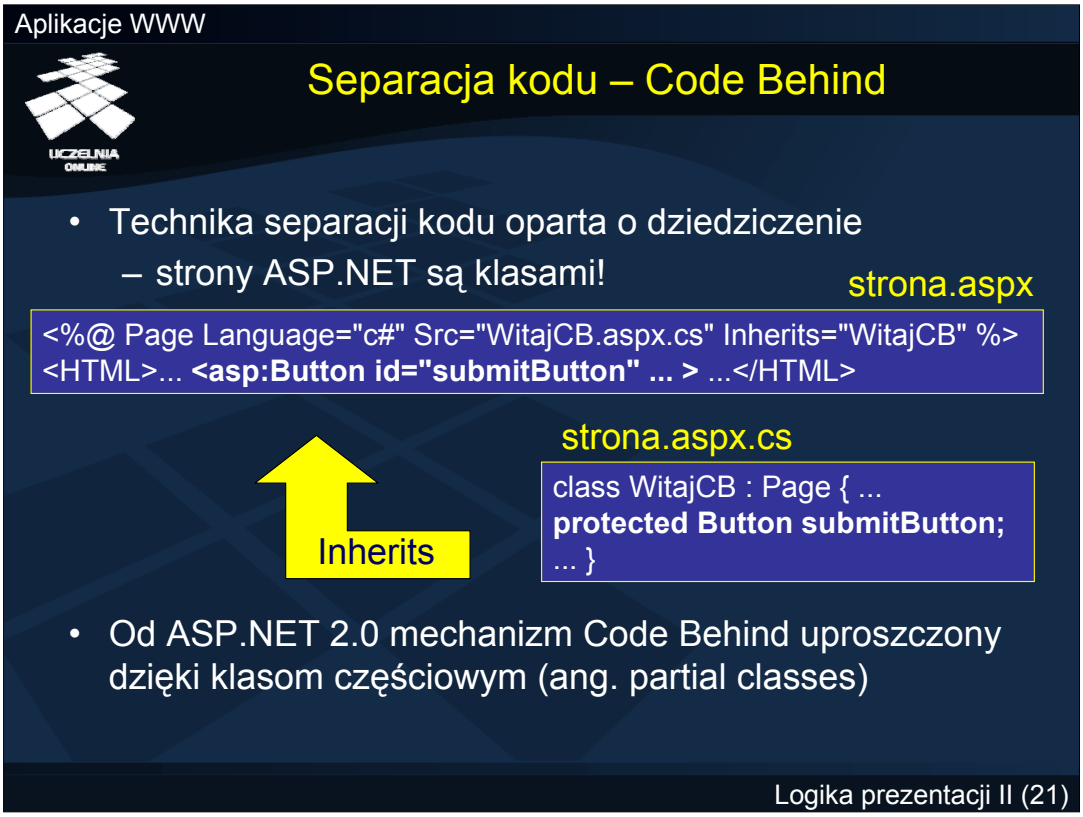
5. Wskazanie przygotowanej kolekcji ArrayList jako źródła danych dla kontrolki DataList.

6. Operacja wiązania danych dla kontrolki DataList.



Podobnie jak serwlety Java, ASP.NET emuluje mechanizm sesji, pozwalając na współdzielenie danych między stronami aplikacji dla poszczególnych użytkowników. Dane sesji użytkownika są dostępne poprzez obiekt Session. Użytkownicy są automatycznie rozpoznawani w oparciu o identyfikator sesji umieszczany w zmiennej cookie lub kodowany w adresie URL.

Na slajdzie przedstawiono kod i efekt działania dwóch stron ASP.NET współdzielących informacje poprzez zmienną sesyjną „Imie”. Obiekt Session zachowuje się jak tablica asocjacyjna, w której kluczami są nazwy zmiennych sesyjnych, a wartościami - wartości tych zmiennych. Strona sesja1.aspx ustawia wartość zmiennej sesyjnej w momencie jej ładowania. Kod ustawiający zmienną sesyjną został zawarty w metodzie Page_Load(). Strona sesja2.aspx wyświetla wartość zmiennej sesyjnej. Zaprezentowany efekt działania przykładu w przeglądarce został zaobserwowany po wywołaniu strony sesja1.aspx, a po niej sesja2.aspx.



Charakterystycznym problemem dla pierwszych technologii „server pages” (takich jak klasyczne ASP) była nieczytelność kodu stron na skutek przemieszania HTML ze wstawkami kodu wykonywalnego. ASP.NET umożliwia pełną separację kodu HTML i wykonywalnego poprzez architekturę Code Behind. Technika Code Behind jest zalecana szczególnie dla złożonych stron i jest domyślnie stosowana przez środowisko Visual Studio.

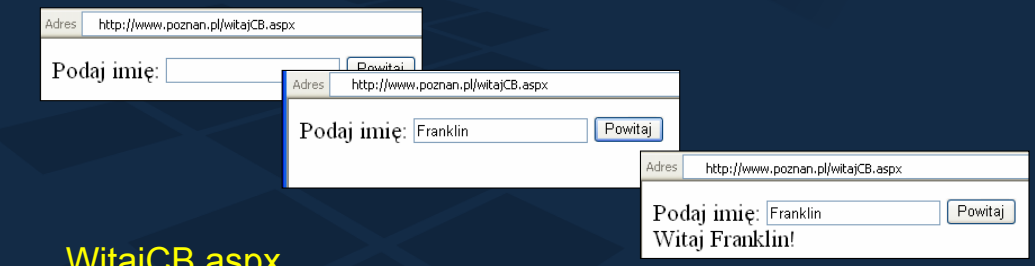
Technika Code Behind wykorzystuje fakt, że strony ASP.NET są klasami i jest oparta o mechanizm dziedziczenia. Każda strona ASP.NET reprezentuje klasę dziedziczącą bezpośrednio lub pośrednio z klasy bibliotecznej Page (dla stron, dla których nie została jawnie wskazana klasa bazowa, klasą bazową jest Page). Code Behind polega na przeniesieniu kodu wykonywalnego do klasy zdefiniowanej w zewnętrznym pliku i wskazaniu tej klasy jako klasy bazowej dla strony za pomocą atrybutu Inherits dyrektywy @Page. Dyrektywa ta zawiera też atrybut wskazujący nazwę pliku Code Behind: Src – gdy pliki są instalowane na serwerze w postaci źródłowej i kompilowane przez runtime ASP.NET, Codebehind – gdy aplikacja jest instalowana na serwerze w postaci skompilowanej (np. zbudowana w Visual Studio).

Zwyczajowe rozszerzenia nazw plików Code Behind to: .aspx.cs dla kodu w języku C# i .aspx.vb dla kodu w języku Visual Basic .NET.

W ASP.NET 1.x klasa bazowa w pliku Code Behind musi zawierać składowe odpowiadające kontrolkom strony ASP.NET, aby możliwe było w niej dodawanie metod obsługujących zdarzenia generowane przez kontrolki. Sposób wskazania metod obsługujących poszczególne zdarzenia jest w przypadku skorzystania z techniki Code Behind inny niż gdy kod HTML i wykonywalny są umieszczone razem w pliku strony. Sposób ten zostanie zilustrowany przykładem na kolejnym slajdzie.

W ASP.NET 2.0 technika Code Behind została uproszczona dzięki mechanizmowi klas częściowych (ang. partial classes).

Code Behind – Przykład (1/2)



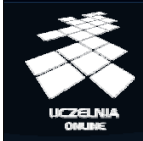
WitajCB.aspx

```
1  <%@ Page Language="c#" Src="WitajCB.aspx.cs" Inherits="WitajCB" %>
   <html><body><form runat="server">
2   Podaj imię: <asp:TextBox id="im" runat="server"></asp:TextBox>
    <asp:Button id="submitButton" runat="server"  Text="Powitaj"></asp:Button>
    <br><asp:Label id="powitanie" runat="server"></asp:Label>
   </form></body></html>
```


Ten i kolejny slajd pokazują przykład separacji kodu techniką Code Behind w kontekście prostej aplikacji pozdrawiającej użytkownika po imieniu. Niniejszy slajd prezentuje treść kodu strony. Strona nie zawiera żadnego kodu wykonywalnego. Znaczenie wyróżnionych wierszy jest następujące:

1. Dyrektywa @Page wskazująca, że klasa strony dziedziczy z klasy WitajCB, a plik Code Behind dla strony nazywa się WitajCB.aspx.cs.

2. Definicja kontrolki przycisku nie zawiera atrybutu onclick. Procedura obsługi zdarzenia z przycisku musi być wskazana w inny sposób w pliku Code Behind.



Code Behind – Przykład (2/2)

WitajCB.aspx.cs

```
1  using System;
   using System.Web.UI;
   using System.Web.UI.WebControls;
   using System.Web.UI.HtmlControls;

2  public class WitajCB : Page {
3    protected TextBox im;
     protected Button submitButton;
     protected Label powitanie;

4    override protected void OnInit(EventArgs e) {
       InitializeComponent(); base.OnInit(e);  }

5    private void InitializeComponent() {
       this.submitButton.Click += new System.EventHandler(
                                    this.submitButton_Click);  }

6    private void submitButton_Click(object sender, EventArgs e) {
       powitanie.Text = "Witaj "+im.Text+"!";   } }
```


Slajd pokazuje plik Code Behind dla przykładowej strony ASP.NET przedstawionej na poprzednim slajdzie. Znaczenie wyróżnionych fragmentów kodu jest następujące:

1. Import wskazanych przestrzeni nazw, w celu umożliwienia posługiwania się nazwami klas z tych przestrzeni bez konieczności prefiksowania ich nazwą przestrzeni. Przestrzeń nazw System zawiera klasy systemowe, System.Web.UI m.in. klasę Page, System.Web.UI.WebControls klasy kontrolek typu Web Controls, a System.Web.UI.HtmlControls klasy kontrolek typu Html Controls.

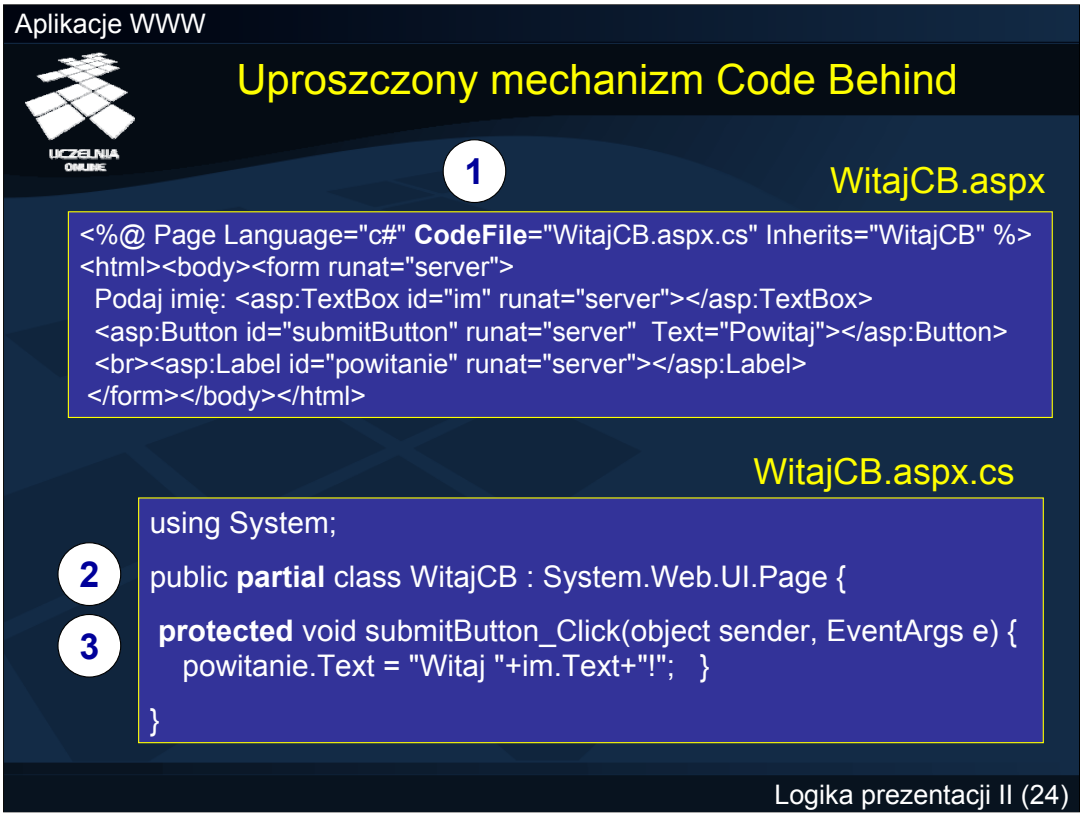
2. Klasa WitajCB, będąca klasą bazową dla strony ASP.NET, jest definiowana jako klasa dziedzicząca z Page. Dzięki temu strona ASP.NET będzie dziedziczyć pośrednio z Page.

3. Definicja składowych klasy reprezentujących kontrolki ze strony ASP.NET, którą obsługuje ten kod Code Behind.

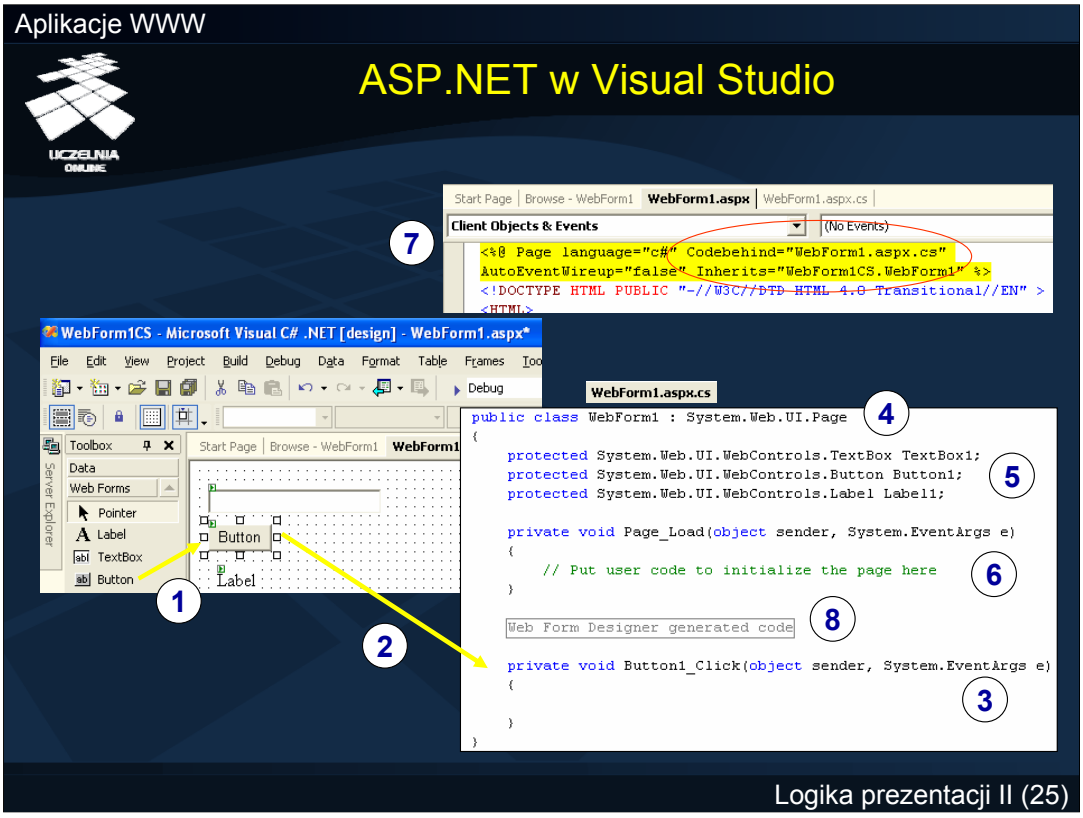
4. Metoda inicjalizująca. Jej zadaniem jest wywołanie metody pomocniczej, która wskaże procedury obsługi zdarzeń. Taki schemat z wyodrębnioną metodą initializeComponent() stosuje Visual Studio.

5. Metoda pomocnicza, wywoływana podczas inicjalizacji strony. Jej zadaniem jest wskazanie metody klasy (w tym wypadku submitButton_Click()) jako procedury obsługi danego zdarzenia z danej kontrolki (w tym wypadku zdarzenia Click przycisku submitButton).

6. Kod metody wskazanej jako procedura obsługi zdarzenia Click przycisku submitButton.



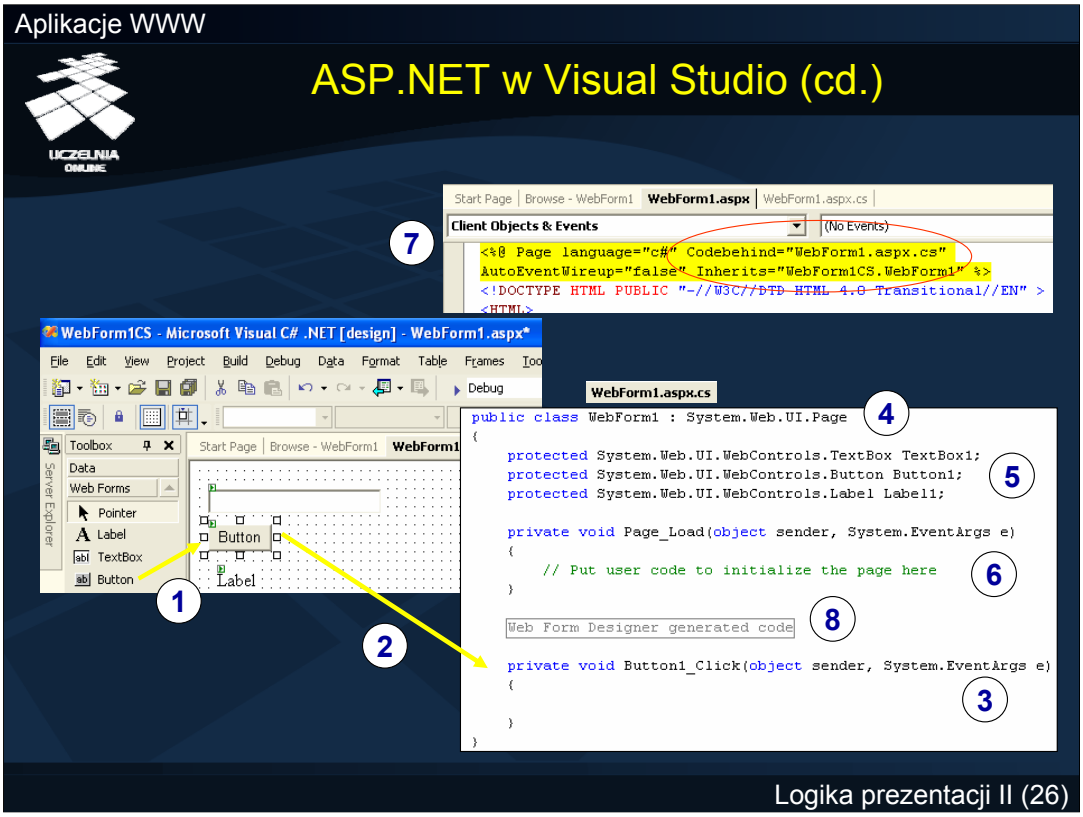
W ASP.NET 2.0 łączenie kodu ze stroną techniką Code Behind zostało uproszczone, co ilustruje zmodyfikowany przykład strony przedstawiony na slajdzie. W nowym podejściu, klasa strony jest deklarowana jako klasa częściowa (słowem kluczowym „partial” (2)) co umożliwia kompilację kodu Code Behind i strony do jednej klasy. Strona odwołuje się do pliku Code Behind atrybutem CodeFile (1). Klasa częściowa Code Behind nie może zawierać składowych prywatnych, dlatego kwalifikator dostępu dla metody obsługującej naciśnięcie przycisku został zmieniony na „protected” (3). Zaletą uproszczonego mechanizmu Code Behind jest to, że nie trzeba już w klasie Code Behind deklarować składowych odpowiadających kontrolkom na stronie.



Kod aplikacji ASP.NET można tworzyć w edytorze tekstowym i kompilować kompilatorami z linii poleceń, zawartymi w .NET Framework SDK. Możliwe jest też umieszczenie na serwerze aplikacji w postaci źródłowej i poleganie na kompilacji przez środowisko uruchomieniowe ASP.NET przy pierwszym odwołaniu do strony.

W praktyce, większość twórców aplikacji .NET korzysta ze zintegrowanego środowiska programistycznego Visual Studio. Z racji tego, że za platformą .NET stoi jedna korporacja istnieje w jej przypadku ścisły związek między technologią a środowiskiem do tworzenia w niej aplikacji. Dodatkowo, koncepcja Web Forms w ASP.NET jest na tyle mocno zorientowana na programowanie wizualne, że nie można omawiać czy oceniać ASP.NET w oderwaniu od środowiska Visual Studio.

Na slajdzie pokazano przykład ilustrujący wizualno-zdarzeniowe tworzenie aplikacji ASP.NET w Visual Studio. (1) Formularz tworzy się poprzez dodawanie do niego kolejnych kontrolek z palety techniką drag-and-drop (w trybie Design). Istnieje oczywiście możliwość przełączenia się w tryb HTML, w celu podejrzenia czy ręcznej edycji źródła strony. (2) W celu obsługi domyślnego zdarzenia generowanego przez kontrolkę na stronie, należy dwukrotnie kliknąć na danej kontrolce i szablon metody obsługującej dane zdarzenie (3) zostanie dodany w pliku Code Behind związanym ze stroną.



W przypadku tworzenia aplikacji w trybie wizualno-zdarzeniowym, środowisko Visual Studio utworzy plik Code Behind w momencie dodawania pierwszej procedury obsługi zdarzenia dla strony. Utworzona w nim będzie (4) klasa dziedzicząca z Page. W klasie pojawią się (5) składowe reprezentujące poszczególne kontrolki ze strony oraz (6) szablon metody Page_Load(), w której można umieścić kod inicjalizujący dla strony. Wygenerowany będzie też kod delegujący wygenerowane metody jako procedury obsługi konkretnych zdarzeń, zwinięty w edytorze jako region „Web Form Designer generated code" (7).

(8) Jednocześnie klasa z pliku Code Behind zostanie wskazana jako klasa bazowa dla strony (atrybut Inherits w dyrektywie @Page), a sam plik Code Behind zostanie wskazany jako kod źródłowy do kompilacji razem ze stroną (atrybut Codebehind w dyrektywie @Page). Atrybut Codebehind jest wykorzystywany, gdy aplikacja ma być skompilowana przed instalacją na serwerze. Gdy strony mają być umieszczane na serwerze w postaci źródłowej i być kompilowane przez runtime ASP.NET, plik Code Behind jest wskazywany na stronie atrybutem Src dyrektywy @Page (jak w jednym z wcześniejszych przykładów).

PHP

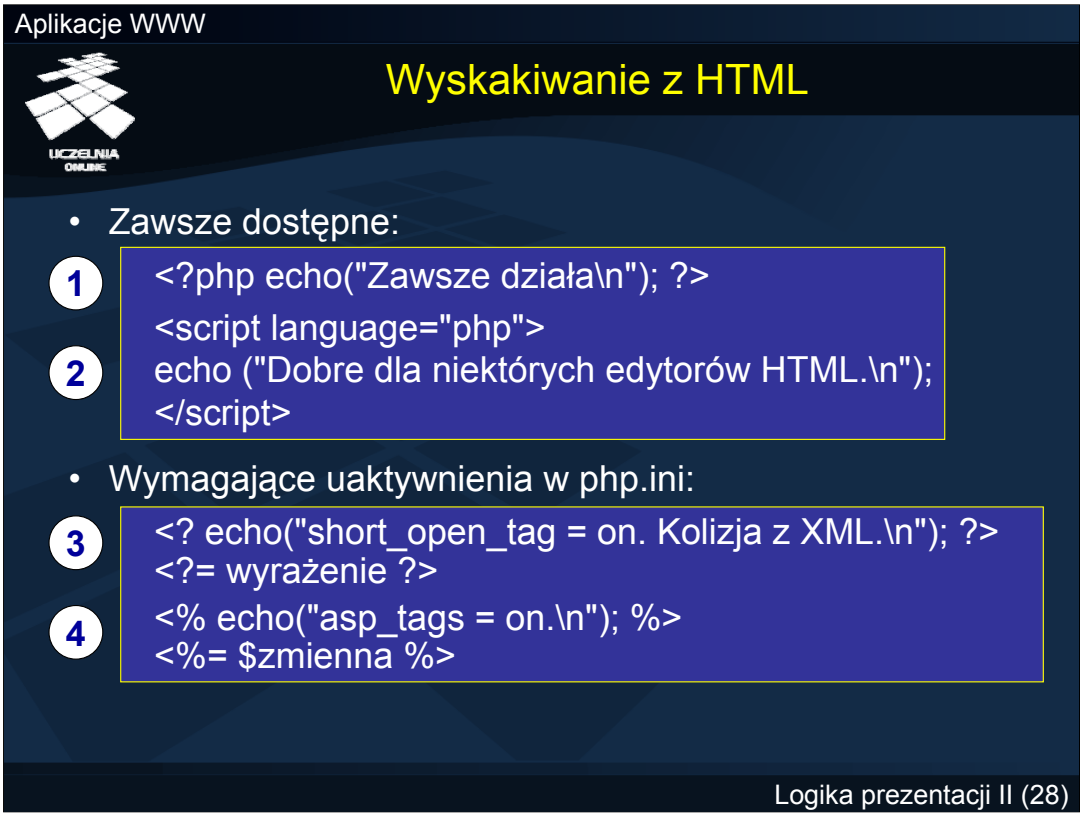
- Język skryptowy opracowany z myślą o aplikacjach WWW
  - dynamiczna generacja stron WWW po stronie serwera
- PHP: Hypertext Preprocessor
- Bardzo łatwy do opanowania
- Szybki i oszczędza zasoby serwera
- Dostępny dla wielu systemów operacyjnych i serwerów HTTP
- Rozwijany na zasadach Open Source

PHP to język skryptowy ogólnego przeznaczenia, ale opracowany z myślą o aplikacjach WWW do dynamicznej generacji stron WWW po stronie serwera. Skrót PHP to formalnie akronim rekursywny „PHP: Hypertext Preprocessor”.

PHP na tle innych technologii jest bardzo łatwy do opanowania w stopniu umożliwiającym tworzenie rzeczywistych aplikacji. Stąd m.in. jego duża popularność. Składnia PHP przypomina składnię języka C z elementami Perla.

PHP jest szybki i oszczędza zasoby serwera. Jest szczególnie popularny w systemie Linux, w połączeniu z serwerem HTTP Apache, ale dostępny jest dla wielu systemów operacyjnych (w tym MS Windows) i serwerów HTTP.

PHP jest rozwijany na zasadach Open Source. Jest darmowy, rozszerzalny, pojawiające się w nim błędy są szybko wykrywane i poprawiane. Wsparcie, rozwiązania typowych problemów i przykłady kodu można bez problemu znaleźć w Internecie.



PHP jest językiem zaprojektowanym z myślą o zagnieżdżaniu w HTML. Przejście do trybu PHP jest określane jako wyskakiwanie z HTML. Istnieją cztery sposoby wyskakiwania z HTML zilustrowane na slajdzie:

1. Podstawowy i zalecany sposób zagnieżdżania kodu PHP.

2. Sposób, który umożliwia pracę z edytorami HTML, które nie wspierają PHP.

3. Składnia z krótkim znacznikiem otwierającym. Nie może być stosowana przy generacji dokumentów XML i XHTML ze względu na kolizję z instrukcjami przetwarzania XML.

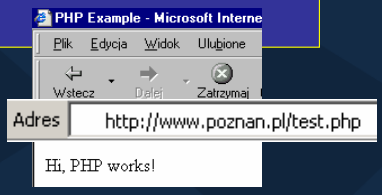
4. Składnia wzorowana na ASP.

Dwa pierwsze sposoby zagnieżdżania kodu PHP w HTML są zawsze automatycznie dostępne. Dwa ostatnie trzeba jawnie uaktywnić w pliku konfiguracyjnym PHP (php.ini).

Przeplatanie kodu HTML i PHP

test.php

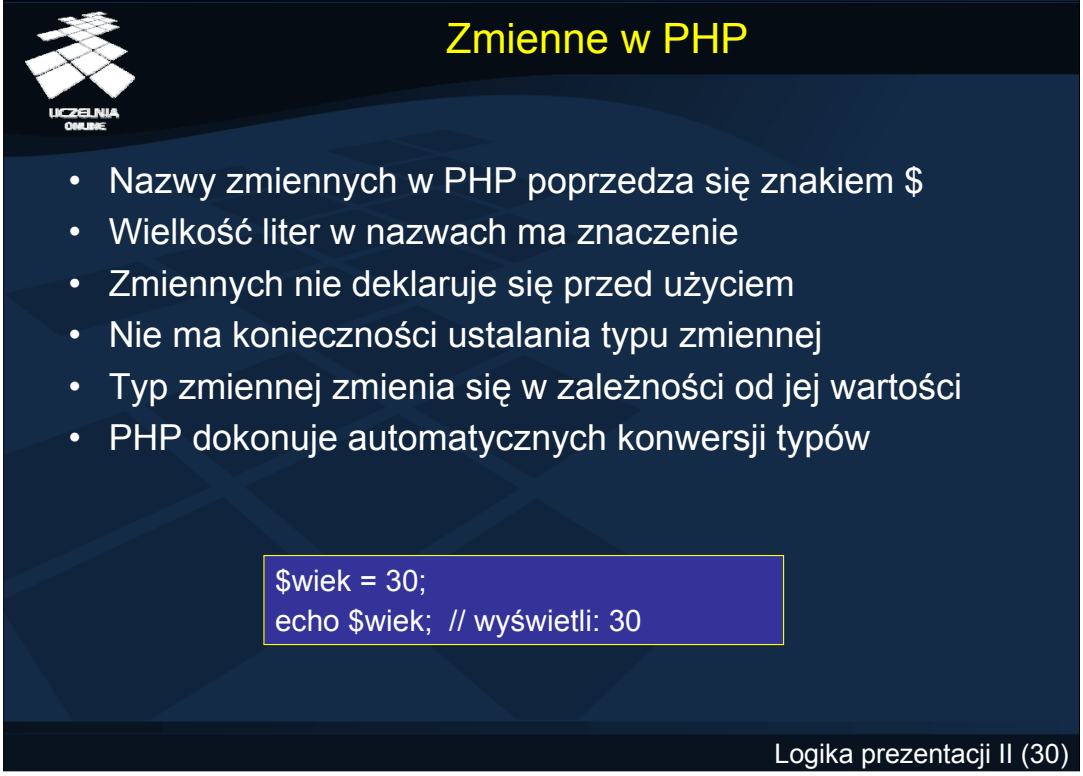
```
<html>
<head>
<title>PHP Example</title>
</head>
<body>
<?php echo "Hi, PHP works!"; ?>
</body>
</html>
```




```
<?php
    if ($wyrażenie) {
?>
    <b>Prawda.</b>
<?php
    } else {
?>
    <b>Fałsz.</b>
<?php
    }
?>
```


Z lewej strony slajdu pokazano kompletny przykład dokumentu PHP wraz z efektem jego działania zaobserwowanym w przeglądarce. Serwer HTTP otrzymując żądanie, odwołujące się do pliku o rozszerzeniu wskazującym, że jest to dokument PHP (zwyczajowo .php), zleca przetworzenie tego pliku modułowi PHP. PHP przetwarzając plik, w niezmienionej postaci wysyła na wyjście kod HTML, a po napotkaniu znacznika przejścia w tryb PHP interpretuje zagnieżdżone instrukcje PHP. W przypadku przykładowego dokumentu test.php, w trybie PHP funkcją echo() wypisany na wyjście jest tekst ‘Hi. PHP works!’. Tekst ten znajdzie się w dokumencie HTML wysłanym przez serwer HTTP do klienta w odpowiedzi na żądanie.

Z prawej strony pokazany jest fragment dokumentu PHP w pełni ilustrujący swobodę przeplatania kodu HTML i PHP. W blokach kodu instrukcji if PHP następuje powrót do trybu HTML. Taki przeplot jest poprawny, ponieważ PHP traktuje tekst pomiędzy „?>” i „<?php” jak gdyby był on argumentem funkcji echo().



Nazwy zmiennych w PHP poprzedzane są znakiem $. Wielkość liter w nazwach ma znaczenie. Podobnie jak w wielu innych językach skryptowych, w PHP zmiennych nie deklaruje się przed użyciem. Nie ma konieczności ustalania typu zmiennej, choć wewnętrznie PHP pamięta dla zmiennej jej typ. Zmienne mogą być typów prostych (boolean, integer, float, string) albo złożonych (tablice, obiekty). Typ zmiennych nie jest deklarowany, PHP ustala go na podstawie kontekstu, w miarę potrzeby dokonując automatycznych konwersji typów.

U dołu slajdu pokazano przykład przypisania wartości zmiennej, a następnie jej wyświetlenia. Podwójny ukośnik rozpoczyna komentarz jednoliniowy.

Łańcuchy znaków w PHP

- Ograniczone apostrofami:
  - bez parsowania zmiennych i sekwencji specjalnych
- Ograniczone cudzysłowami:
  - z parsowaniem zmiennych i sekwencjami specjalnymi

```
$wiek = 30;
echo "Mam $wiek lat.\n"; // wyświetli: Mam 30 lat. ↵
echo 'Mam $wiek lat.\n'; // wyświetli: Mam $wiek lat.\n
```

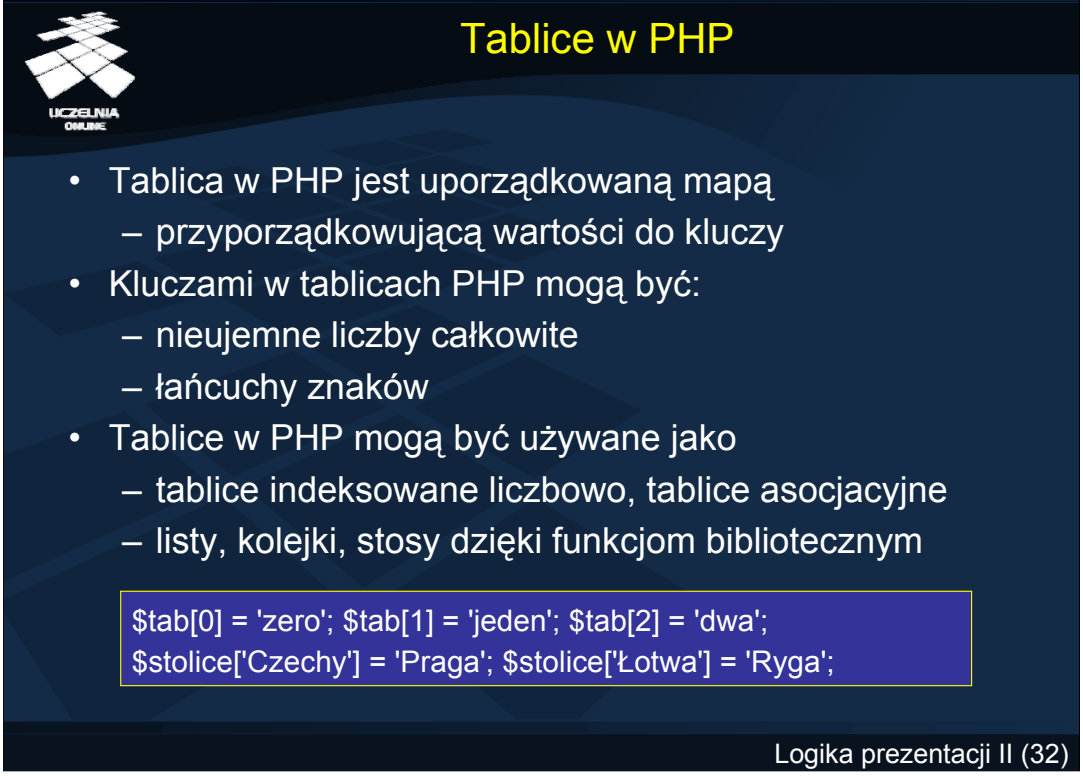
- Konkatenacja operatorem „.”

```
echo "P"."HP"; // wyświetli: PHP
```

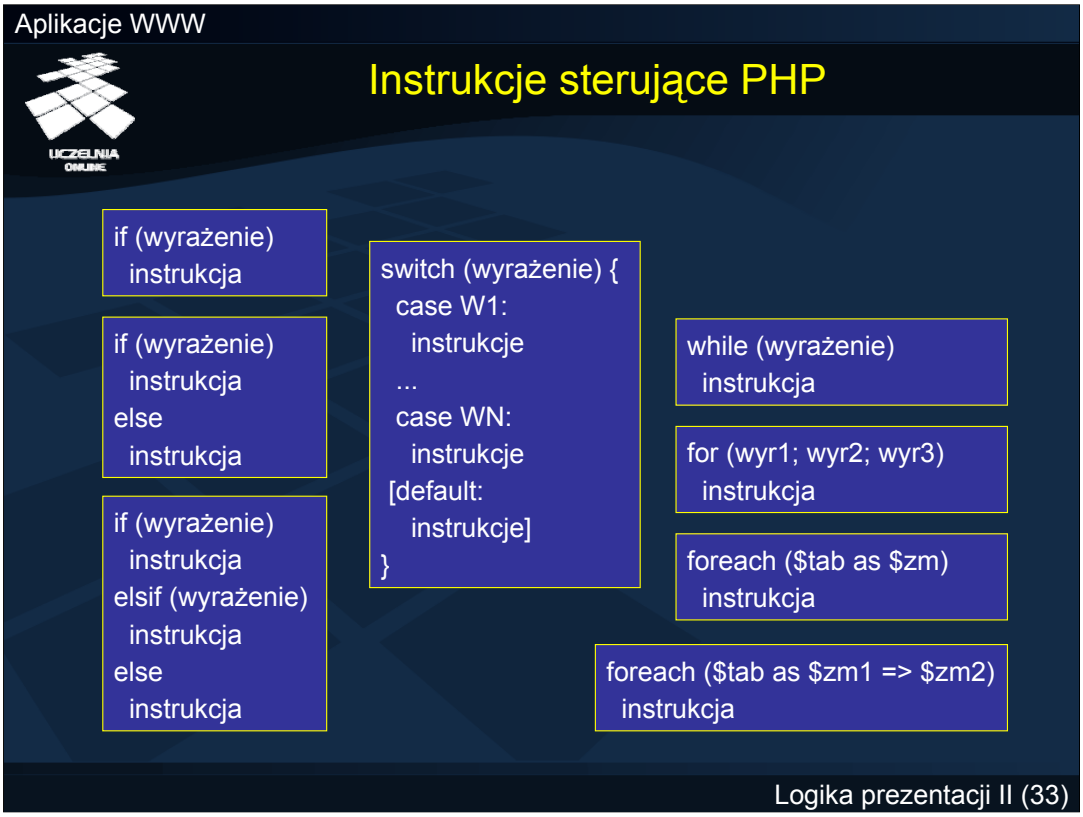
- Wyrażenia regularne
- Bogaty zestaw funkcji bibliotecznych

Łańcuchy znaków w PHP mogą być ograniczone apostrofami lub cudzysłowami. To jaki ogranicznik zostanie użyty ma znaczenie. W łańcuchach ograniczonych apostrofami nie są rozwijane odwołania do zmiennych, nie obowiązują też tzw. sekwencje specjalne jak np. znak nowej linii. W łańcuchach ograniczonych cudzysłowami przy ich przetwarzaniu napotkane nazwy zmiennych są zastępowane aktualną wartością, można też w nich stosować sekwencje specjalne. Różnice między obydwoma notacjami ilustruje przykład na slajdzie.

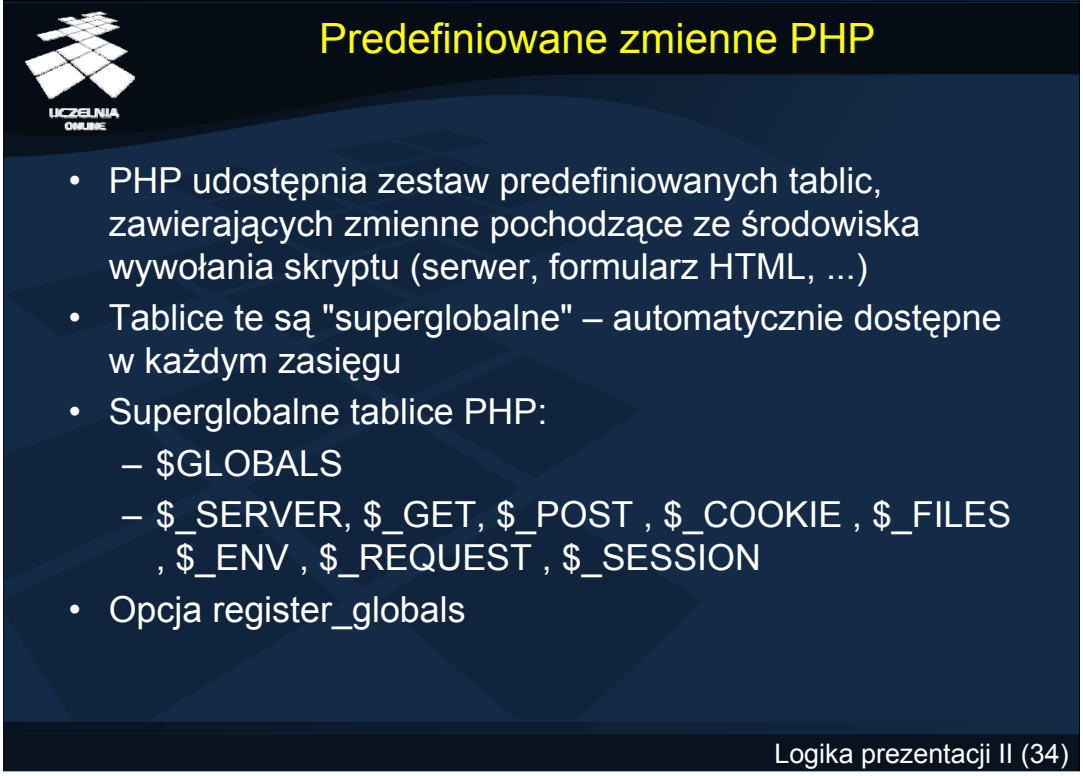
PHP umożliwia konkatenację łańcuchów znaków operatorem kropki, posiada wsparcie dla wyrażeń regularnych, a także bogaty zestaw funkcji bibliotecznych do przetwarzania łańcuchów znaków.



Tablica w PHP jest uporządkowaną mapą przyporządkowującą wartości do kluczy. Kluczami w tablicach PHP mogą być nieujemne liczby całkowite lub łańcuchy znaków. Tablice w PHP mogą bezpośrednio być używane jako tablice indeksowane liczbowo i tablice asocjacyjne, co ilustruje przykład na slajdzie. Można też wykorzystywać tablice jako listy, kolejki, czy stosy dzięki funkcjom bibliotecznym umożliwiającym np. dodawanie elementu na początku lub na końcu tablicy oraz usuwanie elementu z początku lub końca tablicy.



Slajd pokazuje instrukcje sterujące dostępne w PHP. W miejscu gdzie spodziewana jest w danej konstrukcji składniowej instrukcja, może wystąpić pojedyncza instrukcja lub blok instrukcji w nawiasach klamrowych. Większość konstrukcji przypomina te znane z języka C. W instrukcjach pętli można używać instrukcji break i continue. Instrukcją, której nie ma w języku C a jest w PHP jest pętla foreach, umożliwiająca przejście po wszystkich elementach tablicy niezależnie od tego w jaki sposób jest ona indeksowana. Forma pętli foreach u dołu różni się od tej powyżej tym, że podczas iteracji po elementach udostępnia nie tylko wartość, ale również i klucz elementu.



PHP każdemu działającemu skryptowi udostępnia wiele zmiennych pochodzących z zewnątrz, czyli np. ze środowiska serwera, z formularzy HTML itp. Zmienne te są zorganizowane w tablice o nazwach rozpoczynających się od „$_”. Tablice te są „superglobalne” tzn. automatycznie dostępne w każdym zasięgu. Jest to istotne, gdyż zwykłe zmienne globalne w PHP nie są automatycznie widoczne w funkcjach i muszą być w nich zadeklarowane słowem kluczowym „global”.

Superglobalne tablice PHP to:

$GLOBALS – referencje do zmiennych globalnych skryptu (nazwa wyjątkowo bez „_”),

$_SERVER – zmienne tworzone przez serwer,

$_GET – zmienne dostarczone do skryptu metodą GET protokołu HTTP,

$_POST – zmienne dostarczone do skryptu metodą POST protokołu HTTP,

$_COOKIE – zmienne cookies dostarczone do skryptu,

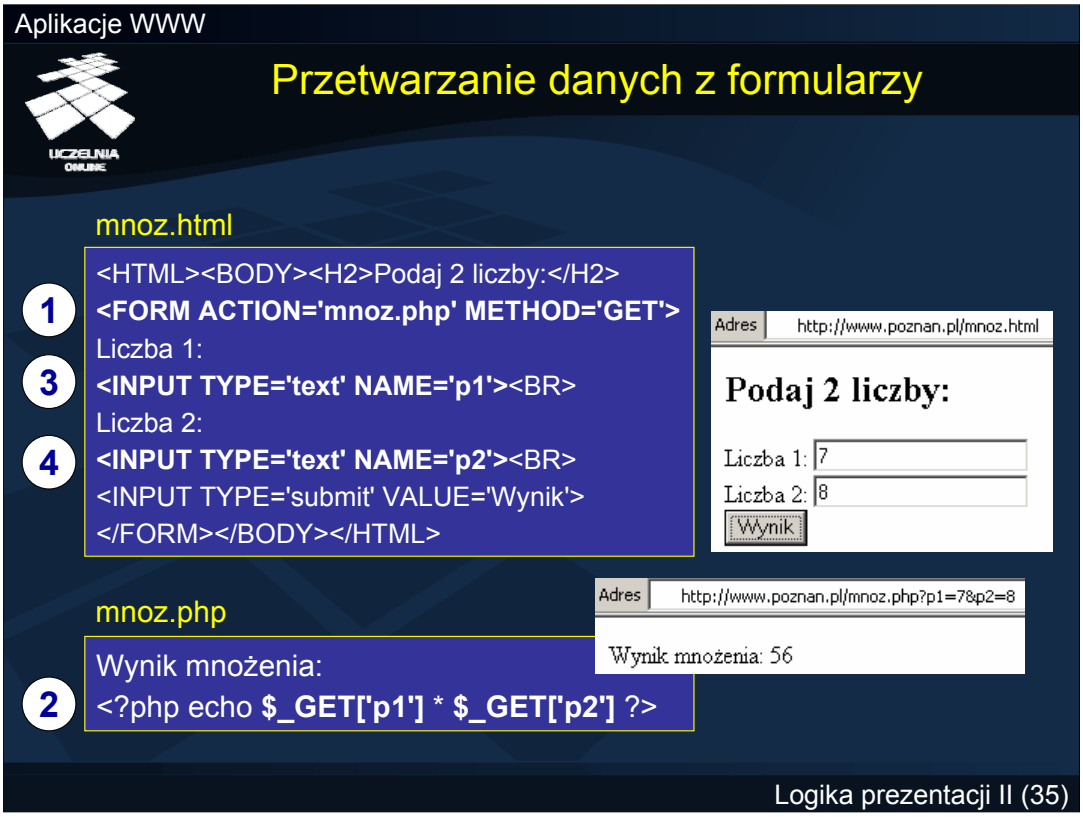
$_FILES – zmienne dostarczone do skryptu przez przesłanie plików do serwera (file upload) metodą POST protokołu HTTP,

$_ENV – zmienne dostarczone do skryptu przez środowisko systemu operacyjnego,

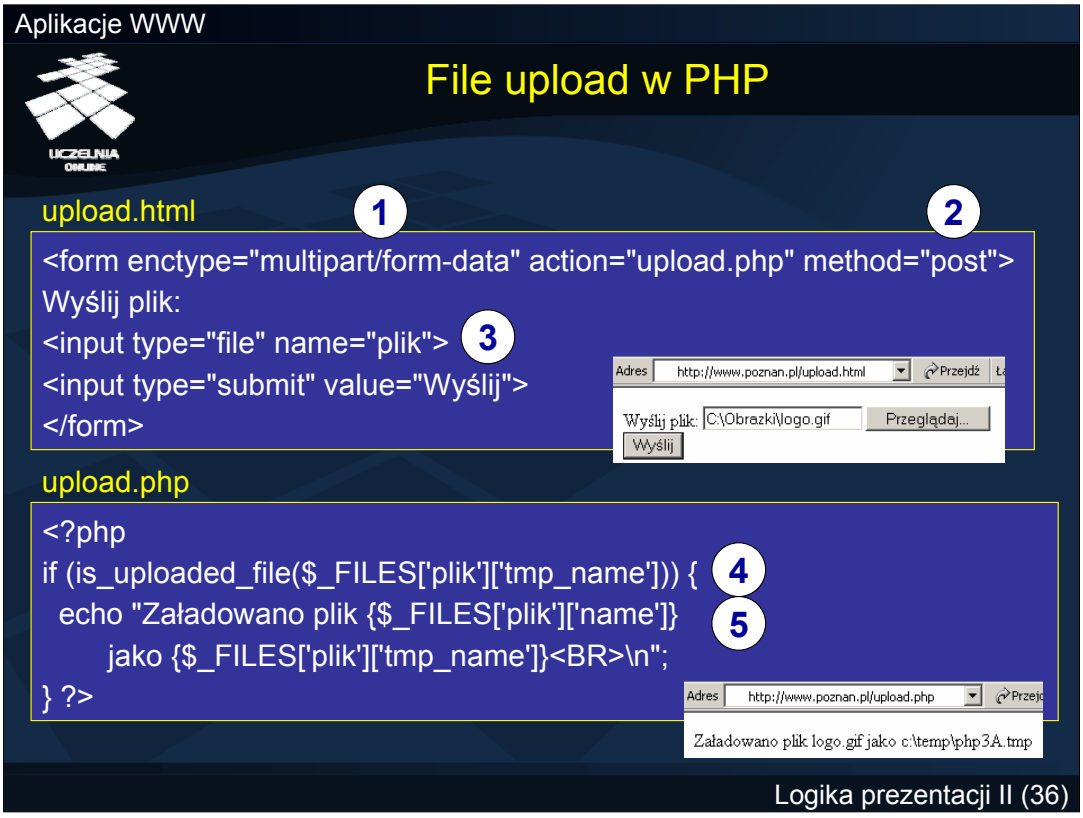
$_REQUEST – zmienne z tablic $_GET, $_POST i $_COOKIE zebrane w jednej tablicy,

$_SESSION – zmienne aktualnie zarejestrowane jako sesyjne.

W starszych wersjach PHP (tj. do PHP 3 włącznie) zmienne przychodzące z zewnątrz były automatycznie rejestrowane jako zmienne globalne w skrypcie bez pośrednictwa tablic. Później rozwiązanie to zarzucono ze względów bezpieczeństwa. Choć nie jest to zalecane, można takie zachowanie przywrócić włączając opcję konfiguracyjną PHP register_globals.

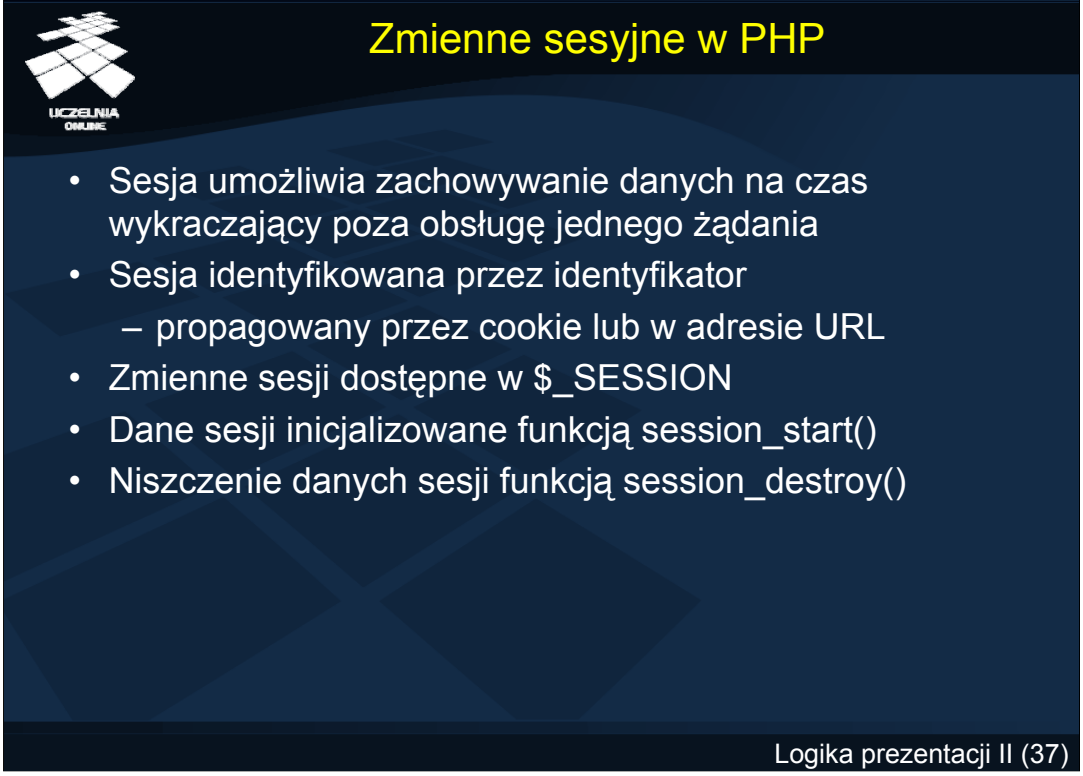


Na slajdzie pokazano przykład prostej aplikacji ilustrującej przetwarzanie danych z formularzy HTML w PHP. Aplikacja składa się z formularza HTML (mnoz.html) służącego do wprowadzenia pary liczb, które mają być przemnożone i dokumentu PHP (mnoz.php), który ma za zadanie wykonać mnożenie po stronie serwera i wygenerować stronę HTML z wynikiem. Formularz HTML wywołuje skrypt PHP metodą GET (1), więc skrypt odczytuje parametry z tablicy superglobalnej $_GET (2). Nazwy parametrów odpowiadają oczywiście nazwom pól w formularzu (3 i 4).

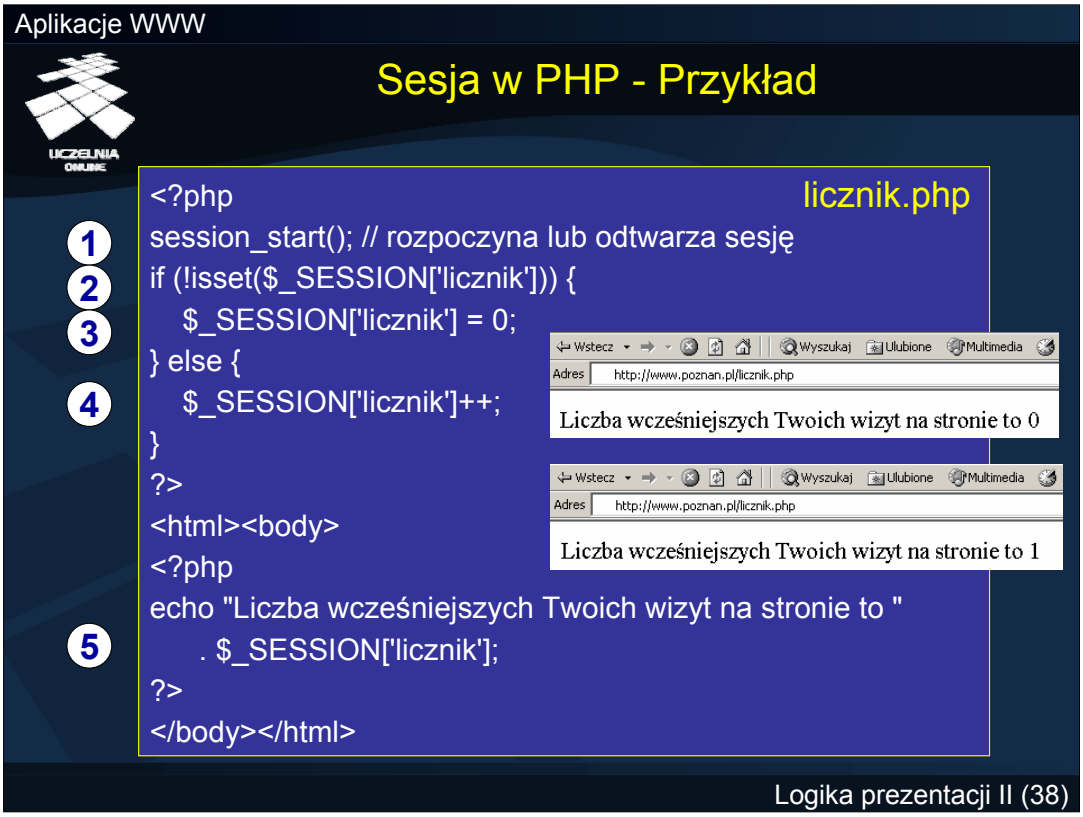


Mechanizm file upload polega na przesyłaniu pliku z lokalnego systemu plików użytkownika na serwer za pośrednictwem formularza HTML wyświetlonego w przeglądarce. Przykładem zastosowania tej techniki są choćby serwisy konferencji naukowych, umożliwiające autorom przesyłanie poprzez WWW plików z propozycjami artykułów. Mechanizm ten posiada wsparcie ze strony języka HTML i protokołu HTTP, wymaga jedynie stworzenia aplikacji gotowej odebrać po stronie serwera plik przesłany z formularza HTML metodą POST. Aplikację taką szczególnie łatwo można napisać w PHP, czego przykład przedstawiono na slajdzie.

Formularz HTML służący do przesyłania plików musi posiadać atrybut enctype="multipart/form-data" (1) oraz wysyłać dane metodą POST (2). Pole w formularzu umożliwiające wybór pliku do przesłania jest tworzone elementem <INPUT> typu „file” (3). Obsługa mechanizmu file upload jest bardzo prosta w PHP, gdyż PHP odbiera przesłane pliki i zapisuje je w katalogu wskazanym w opcjach konfiguracyjnych, udostępniając skryptowi informacje o przesłanych do niego plikach w tablicy superglobalnej $_FILES. Przykładowo, $_FILES['plik']['tmp_name'] dla pliku przesłanego z pola formularza o nazwie „plik” zwraca pełną (wraz z ścieżką), nadaną automatycznie przez PHP nazwę pliku, pod którą został zapisany na serwerze, a $_FILES['plik']['name'] zwraca oryginalną nazwę pliku po stronie klienta (bez lokalnej ścieżki). Skrypt upload.php przedstawiony na slajdzie najpierw sprawdza czy plik został poprawnie przesłany na serwer funkcją is_uploaded_file() (4), a następnie wyświetla potwierdzenie odbioru pliku (5). Typowo, po pozytywnym zweryfikowaniu odbioru pliku, skrypt powinien go skopiować do docelowej lokalizacji funkcją move_uploaded_file().

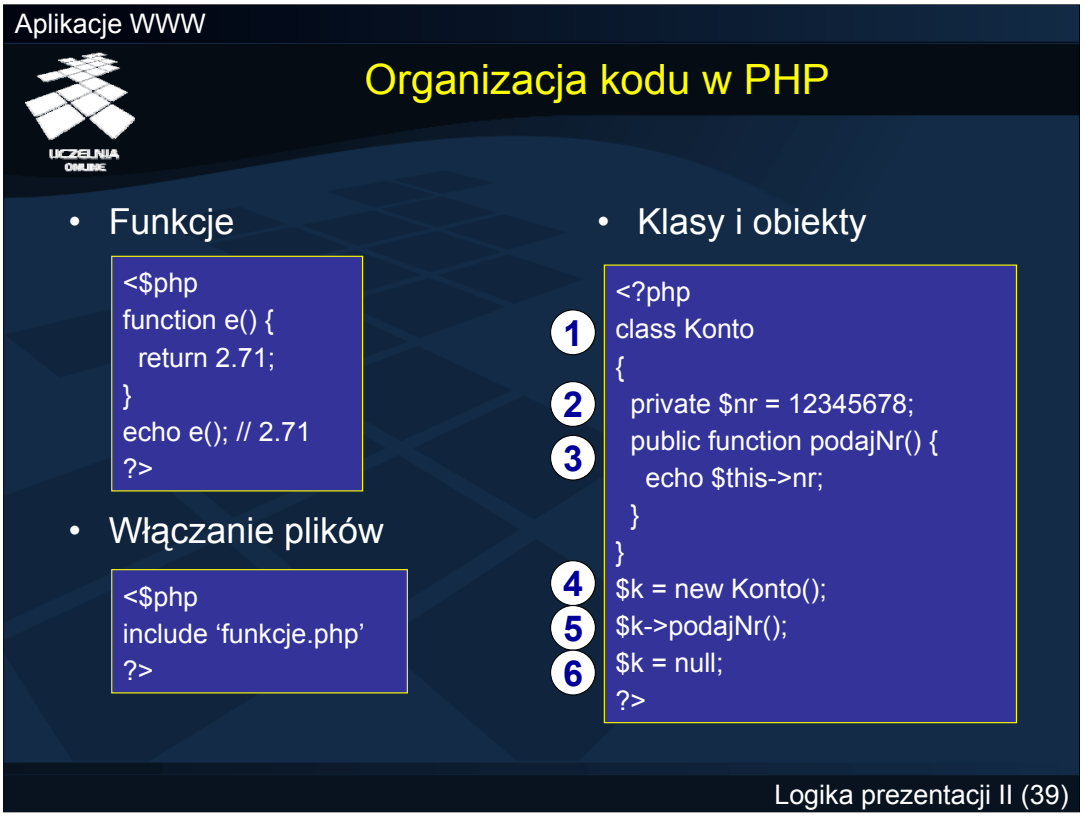


Sesja umożliwia zachowywanie danych na czas wykraczający poza obsługę jednego żądania. Protokół HTTP jest bezsesyjny, więc PHP emuluje sesję dla użytkownika (podobnie jak omawiane wcześniej serwlety Java i ASP.NET). Sesja jest identyfikowana przez identyfikator propagowany domyślnie przez zmienną cookie lub zakodowany w adresie URL gdy zmienne cookie są niedostępne. Zmienne sesji są dostępne poprzez tablicę superglobalną $_SESSION. Rejestracja nowej zmiennej w sesji polega na dodaniu nowego klucza do tej tablicy. Dane sesji są inicjalizowane funkcją session_start(). Funkcja ta kontynuuje bieżącą sesję lub rozpoczyna nową gdy nie ma bieżącej sesji. Wszystkie dane sesji można usunąć funkcją session_destroy().



Zmienne sesyjne mogą być wykorzystane do przekazywania informacji między kolejnymi wywołaniami tej samej strony (jak w przykładzie na slajdzie) lub różnych stron, pozwalając na implementację złożonej logiki aplikacji, obejmującej sekwencję żądań. Przykład przedstawiony na slajdzie zlicza odwiedziny użytkownika na stronie (w czasie bieżącej sesji). Znaczenie wyróżnionych fragmentów kodu jest następujące:

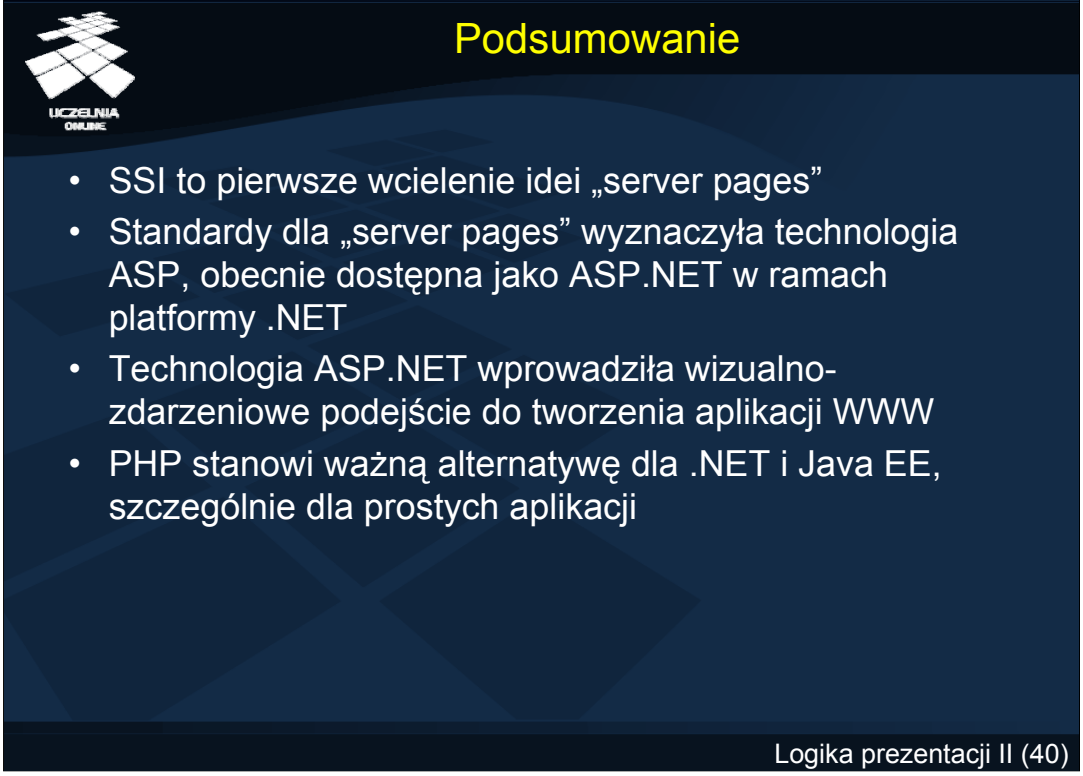
1. Rozpoczęcie lub kontynuacja sesji. Funkcja session_start() musi być wywołana przed wysłaniem jakiejkolwiek zawartości HTML.
2. Sprawdzenie czy zmienna sesyjna „licznik” jest ustawiona.
3. Jeśli zmienna sesyjna „licznik” nie jest ustawiona, jej wartość jest ustawiana na 0, gdyż oznacza to, że strona musiała być wywołana po raz pierwszy w sesji.
4. Jeśli zmienna sesyjna „licznik” jest ustawiona, jej wartość jest zwiększana o jeden.
5. Wyświetlenie wartości zmiennej sesyjnej „licznik”.



W przypadku tworzenia w PHP złożonych aplikacji, zawierających duże ilości wykonywalnego kodu, pojawia się się potrzeba sensownej organizacji kodu. PHP umożliwia programowanie strukturalne poprzez tworzenie funkcji, a od wersji PHP 4 również programowanie obiektowe oparte o klasy. Mechanizmy programowania obiektowego w wersji PHP 5 obejmują m.in. dziedziczenie, konstruktory, destruktor, składowe statyczne i możliwość specyfikowania widzialności składowych (public, protected i private). Zdefiniowane w aplikacji funkcje i klasy mogą być zachowane w odrębnych plikach i włączane tam, gdzie są potrzebne instrukcją include(). Instrukcja ta włącza fragment kodu źródłowego (HTML i/lub PHP), umożliwiając również np. włączanie standardowych nagłówków i stopek.

Na slajdzie pokazano przykład zdefiniowania i wywołania funkcji zwracającej liczbę Eulera, przykład operacji włączenia pliku źródłowego oraz przykład definicji klasy i jej wykorzystania. Znaczenie wyróżnionych fragmentów kodu ilustrującego programowanie obiektowe w PHP jest następujące:

1. Definicja klasy Konto.
2. Definicja składowej (pola) reprezentującej numer konta, z podaniem wartości początkowej.
3. Definicja składowej funkcji (metody) wyświetlającej numer konta.
4. Utworzenie obiektu (instancji) klasy Konto.
5. Wywołanie metody na rzecz obiektu.
6. Przypisanie wartości null do referencji na obiekt. Ponieważ jest to jedyna referencja do obiektu, obiekt jest niszczony.



Server Side Includes (SSI) to pierwsze wcielenie idei „server pages”. Pozwalało na proste dynamicznie generowane wstawki w dokumentach HTML.

Standardy dla „server pages”, rozumiane jako ogólne podejście do tworzenia dynamicznych stron i zestaw znaczników do zagnieżdżania kodu wykonywalnego, wyznaczyła technologia ASP firmy Microsoft. Obecnie dostępna jest nowa, znacząco ulepszona wersja tej technologii - ASP.NET w ramach platformy Microsoft .NET

Rewolucyjnym rozwiązaniem w ASP.NET jest koncepcja Web Forms, która ułatwia tworzenie aplikacji WWW dzięki wizualno-zdarzeniowemu podejściu na wzór tworzenia aplikacji desktopowych. Pewną wadą ASP.NET jest oczywiście ścisły związek technologii z jedną firmą, ale zaletą jest z kolei duża swoboda wyboru języka programowania.

PHP to efektywny, zorientowany na tworzenie aplikacji WWW, i łatwy do nauczenia język skryptowy, który stanowi ważną alternatywę dla ASP.NET i Java EE, szczególnie dla prostych aplikacji.

Materiały dodatkowe

- NCSA HTTPd Tutorial: Server Side Includes (SSI), http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html
- Active Server Pages, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/activeservpages.asp
- .NET Framework Developer Center, http://msdn.microsoft.com/netframework/
- ASP.NET Quickstart Tutorial, http://www.asp.net/QuickStart/aspnet/
- PHP: Documentation, http://www.php.net/docs.php

•NCSA HTTPd Tutorial: Server Side Includes (SSI), http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html

•Active Server Pages, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/activeservpages.asp

•.NET Framework Developer Center, http://msdn.microsoft.com/netframework/

•ASP.NET Quickstart Tutorial, http://www.asp.net/QuickStart/aspnet/

•PHP: Documentation, http://www.php.net/docs.php