Cetec
Capacitações

Centro Paula Souza

GOVERNO DO ESTADO
SÃO PAULO

GOVERNO FEDERAL
BRASIL
PÁTRIA EDUCADORA

- Recuperando dados de um formulário
- Procedimentos e Funções
- Master Page
  - ✓ Conceito
  - ✓ Definições

Recuperando dados de um formulário

Para recuperar dados de um formulário, utilizaremos o objeto Request. As funções deste objeto são:

- **Request.Form** - recupera valores de variáveis enviadas via formulário pelo método post.
- **Request.QueryString** - recupera valores de variáveis enviadas via formulário pelo método get.
- **Request.ServerVariables** - recupera informações sobre o browser do usuário e sobre o servidor WEB.
- **Request.Cookies** - recupera informações do cookie do browser do usuário.
- **Request.ClientCertificate** - mostra informações sobre o certificado de segurança do usuário. Somente é usado em sites com transmissão segura de dados (https)

Request.form

Recupera valores de variáveis enviadas via formulário pelo método post (Quando o usuário clica no botão enviar ou submit do formulário).

Sintaxe: R**equest.form["nomedavariavel“] ou simplesmente Request["nomedavariavel“]**

Abaixo, formulário de cadastramento.

```
<form method="post" action="retrieve.aspx">
    Nome:<input type="text" name="nome">
    Endereço: <input type="text" name="endereco">
    <input type="submit" value="enviar">
</form>
```

A seguir, a rotina de recuperação destes dados preenchidos (nome e endereço). Abaixo, nome e endereço do usuário que preencheu o formulário:

```
<%
  Response.Write "Nome: " & Request.Form["nome“]
  Response.Write "Endereço: " & Request.Form["endereco“]
%>
```

Request.QueryString

Recupera valores de variáveis enviadas via formulário pelo método get (lê a URL que foi enviada pelo usuário).

Sintaxe: **Request.QueryString["nomedavariavel“] ou, também, simplesmente Request["nomedavariavel“]**

Abaixo, formulário de cadastramento.

```
<form method="get" action="retrieve.aspx">
    Nome: <input type="text" name="nome">
    Endereço: <input type="text" name="endereco">
    <input type="submit" value="enviar">
</form>
```

A seguir, a rotina de recuperação destes dados preenchidos (nome e endereço). Abaixo, nome e endereço do usuário que preencheu o formulário:

```
<%
    Response.Write "Nome: " & Request.QueryString["nome“]
    Response.Write "Endereço: " & Request.QueryString["endereco“]
%>
```

Request.ServerVariables

Recupera informações sobre o browser do usuário e sobre o servidor WEB.

Sintaxe: **Request.ServerVariables["valorsolicitado“]**

Abaixo um exemplo de como conseguir alguns valores do Browser do usuário

```
<%
    //Busca o IP remoto do usuário
    Response.Write(Request.ServerVariables["remote_Addr“]);
    //Busca o nome ou o IP do servidor
    Response.Write(Request.ServerVariables["server_name“]);
%>
```

Request.Cookies

Recupera informações do cookie do browser do usuário.

Sintaxe: **Request.Cookies["nomedocookie“]["parâmetrodocookie“]**

Abaixo, rotina que adiciona (Response.Cookies) e que recupera (Request.Cookies) informações do cookie.

```
<%
    //a função time retorna o horário do servidor
    Response.Cookies[“aspnet“]["horavisita“] = DateTime.Now.ToString();
    //acima, rotina de criação de cookie
    Response.Write(Request.Cookies[“aspnet“]["horavisita“]);
    //acima, rotina de recuperação da informação colocada no cookie de nome aspnet
%>
```

Request.ClientCertificate

Mostra informações sobre o certificado de segurança do usuário. Somente é usado em sites com transmissão segura de dados (https).

Sintaxe: **Request.ClientCertificate[parâmetro[sub-parâmetro]]**

Abaixo, exemplo de recuperação de informações sobre um site seguro.

```
<%
    //Retorna a data inicial da validade do certificado
    Reponse.Write(Response.ClientCertificate[“validade“]);
%>
```

Procedimentos

Um procedimento é qualquer bloco de código agrupado com fins de reutilização e organização. Existem dois tipos principais de procedimentos: os que não retornam valores e as funções, que retornam um valor ao código executor. Em C#, um procedimento que não retorna valor é iniciado pela palavra reservada void. O exemplo abaixo é um procedimento simples:

```
void Verificacao()
{
   //...
}
```

Ao ser chamado, Verificacao não recebe nenhum parâmetro e não retorna nenhum valor ao código que o executou, simplesmente executa algumas linhas de código.

Para executar o procedimento basta chamar o nome do método:

```
Verificacao();
```

Procedimentos

No próximo exemplo o método recebe dois parâmetros. Parâmetros são necessários quando o código que será executado precisa internamente de valores para que a rotina possa ser executada devidamente:

```
void Verificacao(int a, int b)
{
   //...
}
```

Para executarmos um procedimento com parâmetros, precisamos informar os valores para cada parâmetro:

```
Verificacao(10, 20);
```

Funções

Funções são estruturalmente semelhantes a um método. As diferenças são que deve ser iniciada pelo tipo de dado de retorno e deve retornar o valor ao código executor usando a palavra reservada **return**:

```
int Somar()
{
   //...
   return total;
}
```

No exemplo anterior **total** é uma variável declarada no corpo do procedimento. A chamada de uma função deve ser feita na forma de uma atribuição, de forma que possamos obter o valor de retorno:

```
int resultado = Somar();
```

Funções

função também pode receber parâmetros em sua declaração:

```
int Somar(int n1, int n2)
{
   //...
   return total;
}
```

Neste caso a chamada de função deve contemplar os parâmetros:

```
int resultado = Somar(30, 40);
```

Master Page

Master Page permite a criação de uma página modelo para todas as outras páginas (**WebForm.aspxx**). Esta página será armazenada com as extensão **.Master**.

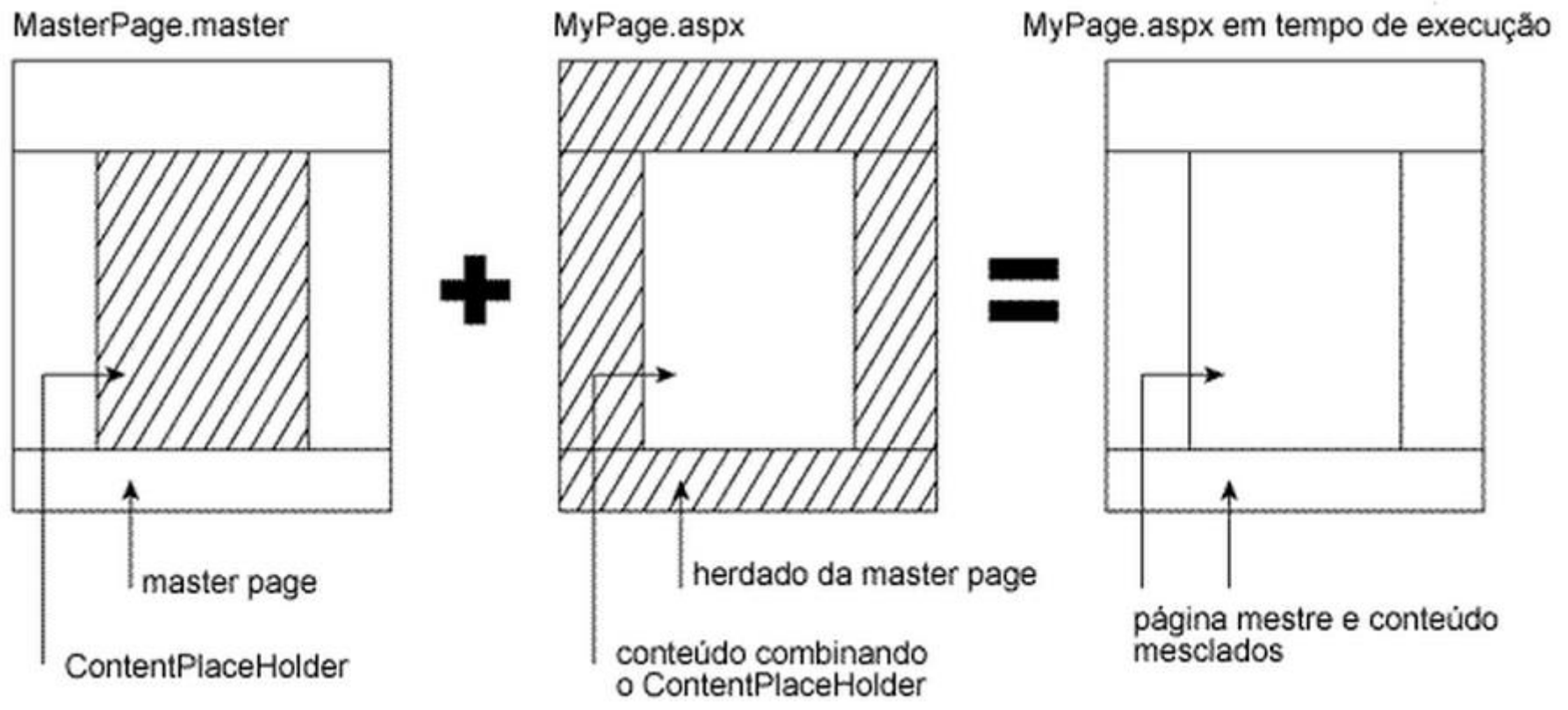
Esta página define áreas (**ContentPlaceHolders**) para gerenciar o conteúdo da página, sendo que este conteúdo pode ser modificado de acordo com as páginas **.aspxx**.

Uma **Master Page** contém um layout que inclui texto, HTML e controles de servidor. Em lugar de uma directiva **"@ Page"**, ela contém uma directiva **"@ MasterPageFile"**. A página mestre contém todos os elementos HTML colocados no topo da página, incluindo <html>, <head> e <form>. Devemos incluir em nossa página mestre, tipicamente, a estrutura da página (usualmente uma tabela), nome da companhia, navegação de site, etc[...]. Para permitir inserção de conteúdos na página mestre devemos adicionar um ou mais controles **<asp: ContentPlaceHolder/>**. Uma página mestre herda da classe MasterPage.

O conteúdo da página é definido pelo controle **<asp:ContentPlaceHolder/>** na página mestre. Um **"Content Page"** é usado em um arquivo .aspxx e é fronteira para a página mestre usando o atributo **MasterPageFile** na directiva **"@ Page"**.

Master Page

Páginas Mestre definem a estrutura e o conteúdo das páginas preenchidas em branco, como mostra a figura abaixo:



Referências

https://msdn.microsoft.com/pt-br/library/ms178194(v=vs.100).aspxx