Введение в CSP

Обзор

- Caché Server Pages
- Приложение CSP
- Принцип работы CSP
- Создание и компиляция страниц CSP
- Среда программирования
- Выражения
- CSP-тэги
- Скрипты
- Методы CSP
- Ссылки
- Контекст
- Классы контекста
- Сессия
- %Request и %Response
- Метод OnPreHTTP()
- Включение файлов
- Наследуемые методы %CSP.Page
- Создание шаблонов и собственных тегов

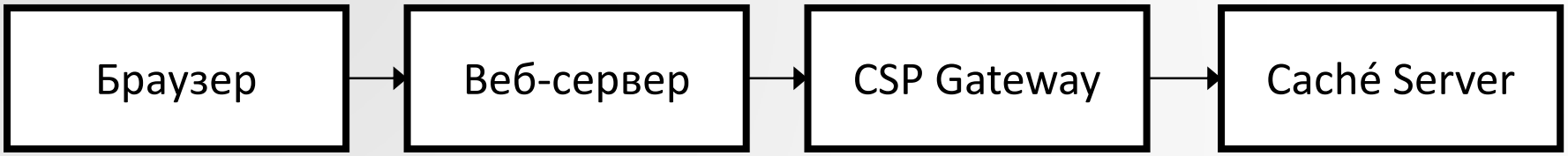
Caché Server Pages

- *Caché Server Pages*: архитектура и инструментарий, используемые для создания интерактивного приложения CSP.
- Программисты разрабатывают *приложение CSP,* добавляя CSP-разметку и скрипты в страницы HTML.

Приложение CSP

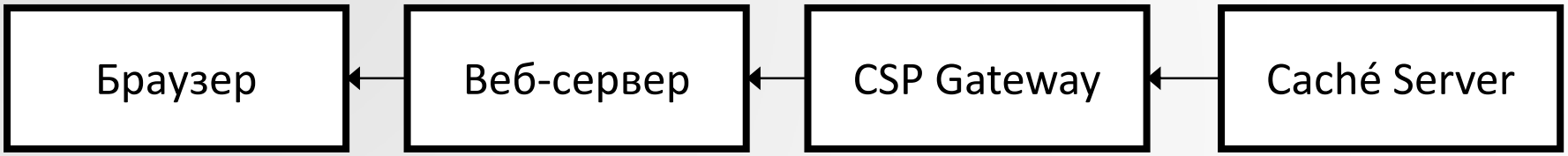
- *CSP-приложение*: набор файлов Caché Server Pages (*CSP-страниц*), размещенных под одним URL.
- Код CSP-страницы обрабатывается сервером, и если в нем есть вызовы серверных функций, они выполняются, а результат их выполнения передается в браузер.

Принцип работы CSP



1. Браузер запрашивает CSP-страницу у веб-сервера.
2. Веб-сервер отвечает, получает расширение «.csp» и передает запрос CSP Gateway.
3. CSP Gateway передает запрос в Caché Server.

Принцип работы CSP



4. Сервер Caché выполняет серверный код и создает содержимое.
5. Готовая страница отправляется в CSP Gateway и на веб-сервер, а потом обратно в браузер.

Создание и компиляция CSP-страницы

- Создание CSP-страницы:
  - Используйте редактор (Caché Studio или любой текстовый редактор).
  - Используйте мастер веб форм в Caché Studio.
- Компиляция
  - Кнопка «*Compile*» (ctr+f7) в Studio компилирует страницу.
  - Caché автоматически компилирует измененные страницы по запросу браузера.

Компиляция в командной строке

- Для компиляции всех страниц CSP используйте команду:

```
do $system.CSP.LoadPageDir(app,flags)
```

- Параметры:
  - ***app***: название веб-приложения. Пример: `/csp/user`.
  - ***flags***:
    - *«**c**»* значит «compile» (скомпилировать). Без него страницы только загружаются.
    - *«**k**»* значит «keep» (оставить сгенерированный исходный код).
    - Для получения полного списка меток используйте:

```
d $system.OBJ.ShowFlags()
```

Упражнение 1-1

Программирование в CSP

Среда программирования

- Среда программирования CSP сочетает в себе:
  - Код **JavaScript** в браузере клиента.
  - Код на языках Caché **(COS, Basic, MultiValue)** на сервере.
    - Выражения
    - Тэги
    - Скрипты
    - Методы
  - Поддержку сессии

Задачи серверной части

- Генерирует заполненный серверными данными HTML и/или JavaScript на сервере.
  - Также генерирует операторы, которые привязывают данные Caché к переменным JavaScript для дальнейшего использования в JavaScript-коде.
- Задает переменные, вычисляет значения и выполняет действия на сервере.

Задачи **JavaScript**

- Вызывает серверный код, передает данные форм и переменные JavaScript на сервер.
- Задает переменные, вычисляет значения и выполняет действия в браузере.
  - Примеры действий:
    - Обновление содержимого полей.
    - Вывод сообщений об ошибках.
    - Обработка событий интерфейса.

Выражения

- Выражения CSP – самый простой способ отобразить динамическое содержимое в любом месте страницы.
- Конструкции работают с возвращающими данные выражениями серверных языков Caché.
  - `#(expr)#` заменяется текущим значением *expr.*
  - `##(expr)##` заменяется значением времени компиляции *expr*

Упражнение 1-2

CSP-теги

CSP-теги

- CSP-теги – это абстракции функционала серверных языков.
- Кавычки обрамляют каждый атрибут.
- CSP-теги позволяют веб-разработчику создавать динамические CSP-страницы без необходимости изучать дополнительный язык программирования.

Пример CSP-тегов

- <csp:comment>
- <csp:if>
- <csp:loop>
- <csp:while>
- <csp:continue>
- <csp:quit>

Страница в документации Caché:

/csp/docbook/DocBook.UI.Page.cls?KEY=RCSP_CSPTAGS

<csp:if>

- **Синтаксис:**

```
<csp:if condition="cond">
```

- **Описание:**
  - Все, заключенное между <csp:if> и </csp:if>, включается на страницу, пока артрибут **condition**=true.
  - Аналогично конструкции If в ObjectScript/Basic.
  - **cond** может быть любым логическим выражением.

<csp:loop>

- **Синтаксис**:

```
<csp:loop counter="var" from="from" to="to" step="step">
```

- **Атрибуты**:
  - ***var*** - переменная, которая меняет свое значение для каждой итерации.
  - ***from*** – выражение для начального значения.
  - ***to*** – выражение для завершающего значения.
  - ***step*** – выражение для значения итерации.

Пример использования тегов

```
<select>
      <csp:loop counter=i from=0 to=9>
            <csp:if condition='i=3'>
                  <option selected>
            <csp:else>
                  <option>
            </csp:if>
            #(i)#</option>
      </csp:loop>
</select>
```

Преобразование на сервере

```
    For i=0:1:9 {
    Write !," "
    If '(i=3) Goto %csp00001 ;{
    Write !," "
    Write "<option selected>"
    Write !," "
    Goto %csp00002 ;}
%csp00001       ;{
    Write !," "
    Write "<option>"
    Write !," "
%csp00002       ;}
    Write !," "_(i)_"</option>",!
    Write " "
    }
```

Упражнение 1-3

Скрипты и методы

Скрипты

- Скрипт - код запускаемый немедленно, в зависимости от позиции в исходном коде.
- Каждый скрипт ***language***=*cache* или ***language***=*basic* должен использовать атрибут *runat*.
  - ***runat***=*server* значит, что код запускается при каждом запросе страницы.
  - ***runat***=*compiler* значит, что код запускается однажды, при компиляции страницы

Скрипты

- Синтаксис скриптов обрамляется <*script*>, #[]# или <*server*>:
  - **ObjectScript** (<*script*>)

```
<script language="cache" runat="server">
set x = a + b
</script>
```

  - **Basic**

```
<script language="basic" runat="server">
x = a + b
</script>
```

  - **ObjectScript** (одной строкой)

```
# [set x = a + b write x ]#
```

  - **ObjectScript** (<*server*>)

```
<server>
for k=1:1:6
{
          w "<br>",!
          w k,!
}
</server>
```

Методы CSP

- Методом в данном контексте называется скрипт **с именем**, вызываемый из других скриптов или выражений на странице.
- Установленные внутри метода переменные доступны для других методов, вызванных позднее.
  - В качестве альтернативы используйте атрибуты ***procedureblock*** и ***publiclist*** для контроля видимости переменной.

Методы CSP

- **Синтаксис**:

```
<script language="lang" method="meth" arguments="args"
returntype="type" procedureblock="procblk"
publiclist="publist">
```

- **Атрибуты**:
  - ***lang*** может быть *cache* или *basic*.
  - ***meth*** – название метода.
  - ***args*** – это список параметров и их типов данных.
    Например:
    ```
    val:%Integer, flag:%String
    ```
  - ***type*** – это вернувшееся значение типа данных.

Методы CSP

- **Синтаксис**:

```
<script language="lang" method="meth" arguments="args"
returntype="type" procedureblock="procblk"
publiclist="publist">
```

- **Атрибуты (продолжение)**:
  - Если ***procblk*** равен 1, компилятор генериует метод как процедуру.
  - ***publist*** – это список открытых переменных.
- **Внимание:** *runat=server* является неявным для метода.

Ссылки

- Ссылки между CSP-страницами обычно такие же, как и между HTML-страницами.
  - Атрибут *href* тега <a> может дать ссылку на другую CSP-страницу.
  - Атрибут *action* тега <form> может дать ссылку на другую CSP-страницу.
- При необходимости существует возможность вызывать ссылки через JavaScript.

Упражнение 1-4

Упражнение 1-5

Контекст и сессии

Контекст

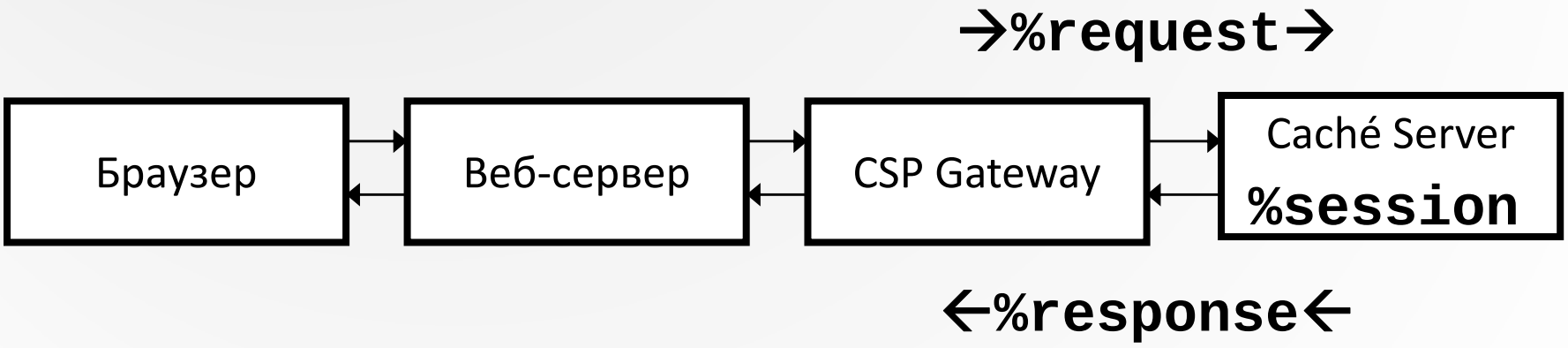
- Обычно веб-приложения в интернете не имеют состояния.
- Приложения, написанные для веб - используют специальные методы, чтобы управлять контекстом приложения или состоянием. CSP предоставляет ряд механизмов управления состоянием.
- Для этого предусмотрены следующие объекты:
  - **%session** – основной объект контекста
  - **%request** – для передачи параметров на сервер
  - **%response** – для передачи параметров на клиент

Классы контекста

- **%CSP.Session** (объект %session) – хранимый класс.

Зарегистрированные классы:

- **%CSP.Request** (объект %request)
- **%CSP.Response** (объект %response)



Сессии

- Когда пользователь запрашивает стартовую страницу (логина) CSP-приложения, начинается новая ***сессия***.
- Caché создает новый экземпляр объекта **%session**.
- Новый **%session ID** хранится в куках.

  Пример: *29ueQXnxga*
- Сессия продолжается и пользователь перемещается по страницам, пока сессия не завершится по тайм-ауту или последняя страница (логаут) не закроет ее.
- Разные приложение использует разные сессии.

Сессия и процессы

- Каждый запрос страницы в течение сессии использует процесс на сервере Caché.
  - Это может быть **один и тот же** процесс каждый раз (*привязка процессов*), но не обязательно.
- Один процесс может обслуживать несколько сессий

Сессии и браузеры

- Несколько окон/вкладок одного браузера на одной машине делят сессии (*по-умолчанию*).
- Каждый экземпляр браузера (IE, Firefox, Chrome, Safari) запускает отдельную сессию.
- Каждое окно IE (до 8 версии), запущенное через кнопку Пуск или иконку IE на одной машине, запускает отдельную сессию.
- Если пользователь или приложение блокирует куки, каждое окно/вкладка браузера на одной машине будет запускать отдельную сессию.
  - Используется механизм включения ID сессии в URL

Сессии и лицензии

- Как разработчику, так и пользователю необходимо учитывать что сессии жестко привязаны к понятию лицензии.
- *Период отсрочки (grace period)* позволяет завершить сессию и перезапустить приложение (в рамках периода отсрочки), используя **ту же самую** лицензионную единицу.
- Лицензионная единица освобождается по истечению *периода отсрочки.*
  - Для одностраничного приложения CSP этот период – 5 минут после завершения сессии. В остальных случаях – сразу после завершения.

Упражнение 1-6

Объект %session

- Сохраняет/получает обрабатываемые данные – начиная с любой страницы – до конца сессии.
  - Чтобы сохранить данные, используйте:

    ```
    do %session.Set(param,data)
    ```

  - Чтобы получить данные, используйте:

    ```
    %session.Get(param)
    ```

- Все сохраняется на диск, отмечается уникальным ID сессии и становится доступным для всех серверных процессов.

%session.**SessionId**

- %session.SessionId содержит уникальный ID сессии.
- CSP схраняет ID сессии в куках для каждой сессии, запущенной на определенной машине.
  - Если браузер пользователя блокирует куки, URL содержат параметр CSPCHD (зашифрованный ID сессии).
- Портал управления отображает текущие сессии:
  - *System Operation* → *CSP Sessions*.

Объект %request

- Получает отправленные данные параметров, полученные как часть запроса от одной страницы к другой.
  - Чтобы получить, используйте команду: `$g(%request.Data)`
- Caché создает экземпляр объекта %request для каждого запроса страницы.
- Caché удаляет %request после генерации страницы.

Использование **%request**

- Имя поля:

```
Name: <input type="text" name="txtName">
```

- ID сотрудника в качестве параметра URL запроса:

```
<a href="empedit.csp?EmpID=#(employee.%Id())#">
  #(employee.Name)# </a>
```

- Получение через %request:
  - `$get(%request.Data("txtName", 1)`
  - `$get(%request.Data("EmpID", 1)`

Использование %request

- Несколько полей с одинаковым именем:

```
A: <input type="checkbox" name="chbTest" value="1">
B: <input type="checkbox" name="chbTest" value="2">
```

`%request.Get("chbTest")` - на сервере вернет value первого выбранного элемента, если такой имеется. Обычно данный подход используется для `type="radio"`.

Объект **%response**

- Предоставляет свойства/методы, которые влияют на HTTP-заголовки, отправляемые браузеру страницей.
  - Обычно **%response** используется в методе OnPreHTTP().
- Caché создает экземпляр объекта %response для каждого запроса страницы.
- Caché уничтожает объект %response после генерации страницы.

Переадресация страницы

- **%response** может вынудить одну страницу запросить вместо себя другую.
- Два свойства:
  - *Redirect.*
  - *ServerSideRedirect*
- Переадресация позволяет:
  - Разным пользователям посещать одну страницу – а потом перемещает каждого на нужную страницу в зависимости от его данных.
  - Разным пользователям посещать одну страницу – а потом всех отправлять на другую.

%response.**Redirect**

- **Синтаксис**:

```
set %response.Redirect = "page"
```

- **Атрибуты**:
  - ***page*** – любая страница (CSP, HTM, другой сайт).
- **Описание:**
  - Перед появлением оригинальная страница запрашивает *page*.
  - Аналогично JavaScript:
    `self.document.location = 'page';`
  - Медленнее, чем ServerSideRedirect.
  - %request.Data уничтожается – на *page* программировать нельзя.
  - Имя новой страницы показывается в браузере.

%response.**ServerSideRedirect**

- **Синтаксис**:

```
set %response.ServerSideRedirect = "page"
```

- **Атрибуты**:
  - ***page*** – **только** CSP-страница
- **Описание:**
  - Генерирует *page* вместо исходной страницы.
  - Быстрее, чем Redirect.
  - %request.Data **не** уничтожается – на *page* можно программировать.
  - В браузере показывается имя исходной страницы – не *page*.
    - Пользователь не знает, что страница сменилась.
    - Зависимые ссылки относятся к **оригинальной странице** – **не** *page*.

Метод **OnPreHTTP()**

- **Синтаксис**:

```
<script language="cache" method="OnPreHTTP" returntype="%Boolean">
```

- **Описание:**
  - Перед генерацией содержимого страницы Caché вызывает наследованный метод OnPreHTTP().

  Применяется для:
  - Использования функций объекта %response.
  - Скриптования перед содержимым страницы.

Упражнение 1-7

Включение файлов (**include**)

- Стандартный функционал CSP поддерживает включение файлов на определенные позиции, для формирования страницы.
- Содержимое включенных файлов добавляется в момент запроса содержащей их страницы.
- **Включенный файл:**
  - Может использовать CSP-тэги и другие конструкции.
  - Не требует завершенного HTML. Т.е. может **не** содержать тегов <**html**> или <**body**>.

Включение файлов (**include**)

- %**request** и %**response** с главной страницы доступны на включенной странице.
- %**request**.URL – URL включенной страницы.

<csp:include>

- **Синтаксис**:

```
<csp:include page="url">
```

- **Атрибуты**:
  - ***url*** – это имя страницы.

  Может использовать параметры запроса. Страница компилируется отдельно от содержащей страницы.

Методы %CSP.Page

- Каждая CSP-страница наследует методы %CSP.Page.
- Некоторые методы являются пустыми заглушками. Если они реализуются разработчиком, методы вызываются автоматически.
  - Пример: OnPreHTTP().
- Некоторые методы – сервисные, их разработчик может вызывать из выражений и скриптов:

Сервисные методы %CSP.Page

Метод	Описание	Пример
`QuoteJS(arg)`	Корректно форматирует/отображает строки JavaScript.	`I'm here`
		`'I\'m here'`
`EscapeHTML(arg)`	Конвертирует строки HTML, корректно отображая спецсимволы.	`a>b`
		`a>b`
`EscapeURL(arg)`	Конвертирует строки URL, корректно отображая спецсимволы.	`a.csp?x=S:T`
		`a.csp%3Fx%3DS%3AT`

Упражнение 1-8

Упражнение 1-9

Создание шаблонов

- Шаблоны в Studio создаются и реализуются с помощью Caché Server Pages (CSP)
- Каждый шаблон - это одна или несколько серверных страниц, которые работают на Caché server.
- Caché включает в себя набор настраиваемых CSP-тегов, которые выполняют все основные операции с шаблонами

Создание шаблонов

- Простой шаблон:

```
<csp:StudioSimpleTemplate name="MyTemplate" type="CSP">
<b>SOME TEXT<b>
```

- Интерактивный шаблон:

```
<csp:StudioInteractiveTemplate name="MyScript" type="CSP">
<FORM NAME="form" ACTION="MyScript2.csp">
…
```

- При компиляции создают шаблон, который можно вызвать через **Инструменты-Шаблоны-Шаблоны**…-*Имя шаблона*

Создание пользовательских тэгов

- CSP позволяет разрабатывать пользовательские HTML-теги для использования в CSP-файлах.
- Для этого используется механизм Rule-Action.
- Для создания пользовательского тега достаточно создать файл правила **.csr**

Структура правила

```
<csr:rule name="param1" match="param2" language="param3"
empty>

<csr:description>
    Описание компонента
</csr:description>

<csr:action>
    Пользовательский код компонента
</csr:action>

</csr:rule>
```

- Использование атрибутов

```
##(..GetAttribute("value","значение по-умолчанию"))##
```

Упражнение 1-10