ПWS

ПОИТ-4,

лабораторная 4

4 часа

**Разработка и исследование простейшего SOAP-сервиса**

1. Разработайте ASMX-сервис с именем **Simplex** помощью технологии ASP.NET.
2. Найдите на сайте W3C описание стандарта WSDL.
3. Найдите на сайте W3C описание протоколов SOAP 1.1, SOAP 1.2.
4. Установите для ASMX-сервиса собственное пространство имен http://XYZ/, где XYZ – ваши инициалы.
5. ASMX-сервис **Simplex** включает в себя следующие три метода.

|  |  |  |
| --- | --- | --- |
| Имя метода и его назначение | Параметры метода | Возвращаемый тип |
| **Add**  Возвращает значение суммы двух параметров | int x,  int y | int |
| **Concat**  Возвращает конкатенацию первого и второго параметров | string s  double d, | string |
| **Sum**  Возвращает объект A:  поле s – конкатенация a1.s и a2.s;  поле к – сумма a1.k и a2.k;  поле f – сумма a1.f и a2.f; | A a1,  A a2 | A |

1. Для класса ASM-сервиса **Simplex** определите параметр **Description** атрибута **WebService**. Поясните его назначение.
2. Для каждого метода ASM-сервиса определите параметр **Description** атрибута **WebMethod**. Поясните его назначение.
3. Для каждого метода ASM-сервиса **Simplex** определите параметр **MessageName** атрибута **WebMethod**. Поясните его назначение.
4. Подключитесь с помощью браузера к ASMX-сервису **Simplex**.
5. Проверьте работоспособность методов **Add** и **Concat** c

помощью стандартной web-формы описания ASMX-сервиса **Simplex**.

1. Для проверки работоспособности метода **Sum** ASMX-сервиса **Simplex** разработайте WinForm-приложение. Используя объект **Request** в методе **Sum** ASMX-сервиса, продемонстрируйте и исследуйте содержимое тела запроса, пришедшего от клиента.
2. Исследуйте WSDL-описание (отображается в браузере), ASMX-сервиса **Simplex** и укажите:

- где применяется собственное пространство имен;

* описание типов данных для всех методов сервиса;
* описание входных и выходных сообщений всех методов сервиса;
* описание методов сервиса;
* описание протоколов обмена данных;
* описание сервиса.

1. Ознакомьтесь со структурой http-запросов для доступа к ASMX-сервису **Simplex** (отображается на стандартной web-форме в браузере).
2. С помощью приложения **POSTMAN** выполните два типа запросов (SOAP 1.2, HTTP POST) к ASMX-сервису **Simplex** и убедитесь в их работоспособности.
3. Разработайте WinForm-приложение. С помощью утилиты **WSDL.EXE** сгенерируйте proxy-класс. Продемонстрируйте его работоспособность в отдельном приложении.
4. Разработайте WebForm-приложение. С помощью утилиты **WSDL.EXE** сгенерируйте класс для отдельного ASMX-сервиса. Продемонстрируйте его работоспособность.
5. Создайте в ASMX-сервисе **Simplex** дополнительный метод **AddS.** Метод **AddS** аналогичен методу **Add,** но предназначен для вызова с помощью AJAX-запроса, отправляющего и принимающего сообщения в формате JSON.
6. Разработайте ASP.NET-приложение, применяющее AJAX-запрос (jQuery) и демонстрирующее работоспособность метода **AddS** ASMX-сервиса **Simplex.**

**Web-сервис** – прил, предост откр интерфейс для других приложений.Его юзеры – другие приложения. Мы говорим здесь о WEB.API приложениях.

**SOA** – сервис-ориентид-архитектура – наше прил взаимод. по протоколу http с др. приложениями, чтобы из них брать инфу или вып. другие д-вия.

**ASMX сервисы** – обесп способ взаимод-я rpc  
**4 технологии, на к-х основ этот веб-сервис**: XML, SOAP, WSDL, UDDI

**\* XML** – структура языка разметки, с пом. него мы м. разраб собст. языки для разметки наших д—х, храним эти д-е в фс либо в бд

\* **SOAP** – протокол обмена инфой в ASMX между клиентом и сервером. Клиент м. отсылать запросы по протоколу SOAP и по нему получать ответы в формате xml.

\* **WSDL** – web service definition language – спец язык, исп формат xml для описания web-сервиса. Он позв нам описать веб-сервис досконально, исчерпывающе. Чтобы мб обратиться к этому сервису с запросами и знать формат ответов кот мы от него получим. \* **UDDI** – Universal Description Discovery & Integration – эта технология подразумевала создание стандарт. хранилищ в wsdl-описаниях – каталоги, кот. будут содерать описание сервисов, Будут какие-то большие компании (алик) и у них будет оч много сервисов, к к-м другие компании смогут подключать свои прогр. продукты.

**Структура SOAP-СООБЩЕНИЯ**

<Envelope> - корневой тег, кот опред сообщение и пр-во имет, исп-ное в доке. Обязателен толко тег <body> (это запрос от К к С, предполаг что будет вызыв процедура GetFoto). Есть еще <soap:header> (служ инфа для программистов), <soap:Fault> (ошибки)

**Структура WSDL дока:**

\* **TYPES** – тд от К к С, мб стандартные и комплексные – структ/класс содержат поля/св-ва, содержащие фундаментальные ТД  
\* **MESSAGE** – опис сообщения между К-С, здесь опис их структура и там исп ТД, кот опис в *types*  
\* **PORTTYPE** – сод. методы удалнного сервера, у сервера есть методы, удал. процедуры кот мы можем вызывать. Эти методы связаны с сообщениями (*messages*), они могут их получать и отправлять, сообщ кот описаны выше  
\* **BINDING** –сод. описание связи soap и протокола транспорта (http), здесь инфа кот позволяет связать soap-протокол с этим http-протоколом, по к-му мы отсылаем сообщение

**ТЕХНОЛОГИИ:**

Вначале появилась ASMX, кот основана на SOAP, потом отдельно разрабатывали платформу **WCF** – отд платформа для разработки прилоежния с SOA . В итоге WCF поглотила ASMX (ASMX – частный случай WCF, тем не менее она достаточно простая и удобная, ms оставил ее отдельно). Но с другой стороны она часть общей технологии asmx. В качестве транспорта исп только веб. Если разработаем сервис asmx, то он обязательно подлежит его публикация на iis-сервер.

**Признаки веб-сервиса ASMX:**  
\* SOAP-протокол  
\* HTTP в кач транспорта  
\* исполняется IIS-сервером (дб опублик)

методы в классе надо декорировать атрибутом [WebMethod], когда вызываем извне.

**КРАТКО:**

жмем **Запустить** – создается сервис (dll, кот имеет свой webconfig, и дб опубликована на IIS), запускается внутренний IIS-сервер, на кот публикуется DLL, и он готов для работы  
Создается узел Localhost, и сразу запускает браузер, кот. делает запрос на этот узел и получает инфу о методах, кот. мы создали, и м проверить их работоспособность. К. новый запрос создает новый экземпляр объекта.

***Proxy class***

класс, построенный на основе описания сервиса(wsdl)

он имеет те же методы, что и удал. сервис, но находится здесь, эмулирует работу удал. сервиса, обращ как с обычным классом – это заместитель, кот притворяется что он сервис, им. те же методы

**WSDL.EXE:**  
Заходим в командную строку VS.

Переходим в папку проекта, где нужно создать прокси класс.

Вводим команду: wsdl http://Ваш\_URL/Ваше\_Название.asmx?wsdl.

Ждём завершения.

Если файл не добавился в проект - смотрим путь создания файла и вручную добавляем.

localhost:51199

1) создать библиотеку служб WCF

IService1.cs: Add, Concat, Sum

Service1.cs: =||=

App.config: tcpNetBinding (другой порт)

2) Host1:

доб. ссылку на службу

App.config: убрать конечную точку tcp

либо при создании клиента в параметре указать кон.точку

3) Host2: =||= для tcp