Newebove metody

private XmlDocument LoadDocument(string filePath)

Áno, tento metóda slúži na načítanie XML dokumentu zo zadaného súboru. Metóda vytvorí nový objekt XmlDocument, a načíta XML dokument zo zadaného súboru pomocou **doc.Load(filePath)**. Potom sa kontrolujú výnimky, ktoré môžu nastať pri načítaní XML dokumentu a v prípade výnimky sa nastaví odpovedajúci status kód a chybová správa pre HTTP odpoveď. V prípade nejakej neznámej chyby sa výnimka zaznamená pre budúce účely ladenia. Ak sa XML dokument úspešne načíta, metóda vráti objekt XmlDocument.

Táto metóda **LoadDocument** slúži na načítanie XML dokumentu zo zadaného súboru. Je napísaná v jazyku C# a je súčasťou webovej služby.

1. Výtvorenie objektu XmlDocument: Prvým krokom v tejto metóde je vytvorenie nového objektu XmlDocument pomocou nasledujúceho kódu: **XmlDocument doc = new XmlDocument();**
2. Načítanie XML dokumentu: Ďalším krokom je pokus o načítanie XML dokumentu zo zadaného súboru pomocou nasledujúceho kódu: **doc.Load(filePath);**. Táto operácia môže spôsobiť výnimky, takže je obalená try-catch blokom.
3. Kontrola výnimiek: V prípade, že sa vyskytne výnimka pri načítaní XML dokumentu, sú tu tri hlavné výnimky, ktoré môžu nastať:
   * FileNotFoundException: Ak súbor, ktorý sa má načítať, sa nedá nájsť, nastaví sa HTTP status kód na 404 a vráti sa chybová správa "Zadaný súbor sa nenašiel."
   * XmlException: Ak sa vyskytne chyba pri parsovaní XML dokumentu, nastaví sa HTTP status kód na 400 a vráti sa chybová správa "Chyba pri parsovaní XML dokumentu."
   * Exception: Ak sa vyskytne akákoľvek iná chyba pri načítaní dokumentu, nastaví sa HTTP status kód na 500 a vráti sa chybová správa "Chyba pri načítaní dokumentu."
4. Výnimka pre ladenie: Ak sa vyskytne neznáma chyba, táto výnimka bude zaznamenaná pre budúce účely ladenia pomocou nasledujúceho kódu: **Console.WriteLine(ex.ToString());**
5. Vrátenie objektu XmlDocument: Ak sa XML dokument úspešne načíta, metóda vráti objekt XmlDocument ako výsledok svojej práce.

V tejto metóde sa využíva objekt **Context.Response**, ktorý slúži na konfiguráciu HTTP odpovede. Pomocou neho sa nastavuje HTTP status kód a popisná ch

Pomocou neho sa nastavuje HTTP status kód a popisná ch here you stop finish please

Áno, táto metóda sa používa na načítanie XML dokumentu zo zadaného súboru. Ak sa dokument načíta úspešne, metóda vráti objekt XmlDocument. Ak však nastane chyba pri načítaní súboru, napríklad zadaný súbor sa nenašiel alebo sa vyskytla chyba pri parsovaní XML dokumentu, metóda nastaví HTTP status kód a vráti popisnú chybovú správu. Ak sa vyskytne iná neznáma chyba, metóda tiež nastaví HTTP status kód a vráti popisnú chybovú správu. Výnimky, ktoré sa vyskytnú pri načítaní dokumentu, budú tiež zaznamenané pre budúce účely ladenia.

can you describe it more detailed

Áno, poviem Vám o ňom viac.

Táto metóda, nazvaná **LoadDocument**, má ako argument jeden stringový parameter **filePath**, ktorý udáva cestu k súboru, ktorý sa má načítať ako XML dokument.

V rámci metódy sa vytvorí nový objekt typu **XmlDocument** s názvom **doc**, ktorý bude slúžiť na uloženie nahratého dokumentu.

V bloku **try-catch** sa potom pokúsi nahrať XML dokument zo zadaného súboru cez **doc.Load(filePath)**.

V prípade, že sa súbor nedá nájsť (čo spôsobuje výnimku typu **FileNotFoundException**), nastaví sa HTTP status kód na 404 a popisná chybová správa "Zadaný súbor sa nenašiel". Taktiež sa vráti správa o chybe.

V prípade, že sa vyskytne chyba pri parsovaní XML dokumentu (čo spôsobuje výnimku typu **XmlException**), nastaví sa HTTP status kód na 400 a popisná chybová správa "Chyba pri parsovaní XML dokumentu". Taktiež sa vráti správa o chybe.

Ak sa vyskytne akákoľvek iná chyba (čo spôsobuje výnimku typu **Exception**), nastaví sa HTTP status kód na 500 a popisná chybová správa "Chyba pri načítaní dokumentu". Taktiež sa vráti správa o chybe a výnimka bude zaznamenaná pre budúce účely ladenia prostredníctvom funkcie **Console.WriteLine(ex.ToString())**.

Ak všetko prebehne bez chýb, metóda vráti načítaný XML dokument.

public static string CreateTimestamp()

Yes, of course! The "CreateTimestamp" method is a static method that returns a string representation of the current date and time. The method uses the "DateTime.Now" property to get the current date and time and then formats it using the "ToString" method with the format string "yyyy-MM-dd HH:mm:ss". The returned string is in the format of "Year-Month-Day Hour:Minute:Second", which is a common timestamp format used in many systems.

could you describe it in more details

This is a method named "CreateTimestamp" that returns a string representation of the current date and time. It uses the "DateTime.Now" property to retrieve the current date and time, and then formats it using the "ToString" method with a custom format string "yyyy-MM-dd HH:mm:ss". This format string specifies the desired format for the output string, with "yyyy" representing the year, "MM" representing the month, "dd" representing the day, "HH" representing the hour, "mm" representing the minute, and "ss" representing the second. The resulting string will have the format "yyyy-MM-dd HH:mm:ss", and will be returned by the method as its output.

public void WriteToTheFileWithTimeStamp(string path, XmlNodeList data)

Táto metóda slúži na zápis dát do XML súboru s časovým údajom.

Metóda prijíma ako argumenty cestu k súboru a zoznam dát vo forme XML uzlov.

Najprv skontroluje, či súbor existuje, a ak nie, vytvorí nový XML dokument a koreňový element. Ak súbor existuje, načíta dokument a získa koreňový element.

Potom prechádza každým uzlom v dátach a importuje ho do dokumentu. Vytvorí elementy "output" a "timestamp" a nastaví vnútorný text elementu "timestamp" na aktuálny dátum a čas pomocou volania metódy "CreateTimestamp". Potom pripojí importovaný uzol, element "timestamp" a element "output" ku koreňovému elementu.

Na konci metóda uloží dokument do zadaného súboru.

Áno, táto funkcia slúži na zápis dát vo forme XML do súboru s časovou pečiatkou. Funkcia skontroluje, či súbor existuje, ak nie, vytvorí nový súbor a ak áno, načíta existujúci súbor. Potom prejde cez všetky uzly v zadanom XmlNodeList a pridá ich k súboru ako elementy s časovou pečiatkou. Ak sa vyskytnú chyby pri práci so súborom, táto funkcia vypíše príslušnú chybovú správu.

Táto funkcia slúži na zápis dát vo forme XML do súboru s pridanou časovou pečiatkou. Vstupom funkcie je cesta k súboru a zoznam uzlov XML (XmlNodeList). Funkcia začína tým, že skontroluje, či súbor existuje. Ak súbor neexistuje, funkcia vytvorí nový súbor v XML formáte a pridá koreňový element. Ak súbor existuje, načíta existujúci súbor a získa koreňový element.

Potom prejde cez všetky uzly v XmlNodeList a pre každý uzol vytvorí nový element s názvom "output". Pre tento element sa tiež vytvorí nový element s názvom "timestamp", ktorý obsahuje časovú pečiatku. Uzol z XmlNodeList sa importuje do objektu XmlDocument a pridá k elementu "output". Element "output" sa potom pridá ku koreňovému elementu.

Funkcia tiež obsahuje try-catch blok, ktorý zachycuje výnimky, ktoré môžu nastať pri práci so súborom. Ak sa vyskytne chyba pri vytváraní súboru, čítaní súboru, zápise do súboru alebo prístupe k súboru, funkcia vypíše príslušnú chybovú správu.

Všeobecne, táto funkcia slúži na pridanie dát vo forme XML do súboru s pridanou časovou pečiatkou a zabezpečuje, aby boli v prípade chýb výstupné správy pre používateľa.

Táto jedna riadok kódu vytvára objekt typu XmlDeclaration, ktorý predstavuje deklaráciu XML dokumentu. XmlDeclaration obsahuje informácie o verzii XML, kódovaní a informáciu o externom DTD súbore, ak existuje.

Metóda **CreateXmlDeclaration** je volaná na objekte typu XmlDocument (**doc**) a prijíma tri argumenty:

* Prvý argument "1.0" predstavuje verziu XML, ktorú bude dokument používať.
* Druhý argument "UTF-8" predstavuje kódovanie, ktoré bude použité v dokumente. UTF-8 je štandardným kódovaním pre XML dokumenty a zaručuje podporu pre rôzne jazyky.
* Tretí argument "null" predstavuje informáciu o externom DTD súbore. Nastavením hodnoty na "null" naznačuje, že v dokumente nebude použité externé DTD.

Výsledkom volania metódy **CreateXmlDeclaration** je objekt typu XmlDeclaration, ktorý je priradený do premenej **xmlDeclaration**. Neskôr v kóde bude použitý pri vytváraní nového XML dokumentu.

Webove služby in c sharp

SinglebookDataById

This code is a C# method for a web service that retrieves information about a single book based on the ID provided.

1. It starts by calling the LoadDocument method and passing in the fileBookInfo string, which is likely the file path for the XML document containing the book information. If the document could not be loaded, the method sets the response status code to 500 and sets the status description to "Dokument nemohol byt nacitany".
2. Next, the method checks if the provided ID string is null or empty. If it is, the response status code is set to 400, the status description is set to "Nezadali ste hodnotu id", and a message is written to the response indicating that no ID was provided.
3. The method then uses an XPath query to select the book node in the XML document with an id attribute that matches the provided ID. If a matching node is found, the method calls the WriteToTheFileWithTimeStamp method and passes in fileOutputSingleSearch and the single book node. It then writes the serialized XML node to the response as a JSON string.
4. If the book node could not be found, the response status code is set to 404, the status description is set to "Zadaný záznam sa nenasiel prosím skontrolujte svoj vstup", and a message indicating that no record was found for the provided ID is written to the response.
5. Finally, the method has a try-catch block that catches any exceptions that may occur during execution. If an exception is thrown, the response status code is set to 500, the status description is set to "Interná chyba servera", and a message indicating that an internal server error occurred is written to the response.

This line of code is using the SelectNodes method of the XmlDocument object doc to select a list of nodes (stored in nodeListBook) that match a specific criterion. The criterion is defined using an XPath expression:

Vysvetlenie doležitej časti kodu

css

"Bookstore/books/book[translate(nazov,'ABCDEFGHIJKLMNOPQRSTUVWXYZÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞŸŽŠŒ','abcdefghijklmnopqrstuvwxyzàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþÿžšœ') = \"" + name + "\"]"

The expression starts with "Bookstore/books/book", which selects all nodes named "book" that are children of nodes named "books", which in turn are children of a node named "Bookstore".

The part inside the square brackets [] adds a condition to the selection: it only selects nodes whose value of the "nazov" attribute is equal to the name parameter passed to the SinglebookDataByName method, after converting both to lowercase

he code is selecting nodes from an XML document doc using an XPath expression.

nodeListBook is a list of XML nodes that match the conditions specified in the XPath expression.

The expression itself navigates through the XML document to find elements with the following structure:

php

<Bookstore>

<books>

<book>

<nazov>...</nazov>

</book>

</books>

</Bookstore>

The expression then applies a translate() function to the nazov element, converting all uppercase characters to lowercase and replacing them with their corresponding lowercase characters. This is done so that the comparison between the nazov value and the input name is case-insensitive.

Finally, the expression checks if the translate-ed nazov value is equal to the input name. If it is, the node is selected and added to the nodeListBook list.

The selected nodes in nodeListBook will be written to a file with a timestamp and serialized to a JSON string, which will be written to the HTTP response as the output of the method.

SinglebookDataByName

Certainly. The SinglebookDataByName method is a web service method that is used to retrieve a single book record from an XML file based on the name of the book.

The method starts by loading an XML document stored in a file, specified by the fileBookInfo variable, using the LoadDocument method. If the document can't be loaded, it sets the status code to 500 and sets the status description to "Dokument nemohol byt nacitany", which means "The document could not be loaded".

Next, it checks if the name of the book was passed as an argument to the method. If not, it sets the status code to 400 and sets the status description to "Nezadali ste hodnotu mena knihy", which means "You did not enter the book name value", and returns from the method.

The method then converts the name of the book to lowercase and uses an XPath expression to select all book nodes in the XML document whose name matches the passed name. If there are no matching nodes, it throws an exception with the message "Record not found for the given name".

Finally, the method writes the selected node(s) to a file specified by the fileOutputSingleSearch variable using the WriteToTheFileWithTimeStamp method. The method also returns the selected node(s) in JSON format, encoded as UTF-8, by using the JsonConvert.SerializeXmlNode method.

If an exception is thrown, the method sets the status code to 500 and sets the status description to "Chyba pri spracovavani poziadavky", which means "Error processing request", and returns an error message with the exception's message.

Tento kód slúži na získanie údajov o knihe z XML dokumentu na základe zadaného mena knihy.

Metóda **SinglebookDataByName** je webová metóda, ktorá prijíma jeden parameter mena knihy.

V bloku try sa snaží načítať XML dokument pomocou metódy **LoadDocument**, pričom cesta k dokumentu sa nachádza v premennej **fileBookInfo**. Ak sa dokument nepodarí načítať, nastaví sa stavový kód odpovede na 500 a popis na "Dokument nemohol byť načítaný".

Ďalšie sa kontroluje, či bolo zadané meno knihy. Ak nie, nastaví sa stavový kód na 400 a popis na "Nezadali ste hodnotu mena knihy".

Potom sa meno knihy prevedie na malé písmená pomocou funkcie **ToLower()**. Vyhľadávanie sa vykoná pomocou XPath výrazu na výber uzlov **book**, ktoré majú atribút **nazov** rovný zadanému menu.

Ak sa vyhľadaný zoznam uzlov nenachádza alebo je prázdny, metóda vyhodí výnimku s hláškou "Record not found for the given name" a nastaví stavový kód odpovede na 500 a popis na "Chyba pri spracovávaní požiadavky".

V opačnom prípade sa vyhľadaný uzol zapíše do výstupného súboru s časovou pečiatkou pomocou metódy **WriteToTheFileWithTimeStamp** a následne sa serializuje do JSON formátu a zapíše sa do odpovede.

V bloku catch sa spracováva výnimka, pričom sa nastaví stavový kód odpovede na 500 a popis na "Chyba pri spracovávaní požiadavky" a do odpovede sa zapíše hláška "Error: " s textom výnimky.

Vysvetlenie doležitej časti kodu

css

"Bookstore/books/book[translate(nazov,'ABCDEFGHIJKLMNOPQRSTUVWXYZÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞŸŽŠŒ','abcdefghijklmnopqrstuvwxyzàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþÿžšœ') = \"" + name + "\"]"

The expression starts with "Bookstore/books/book", which selects all nodes named "book" that are children of nodes named "books", which in turn are children of a node named "Bookstore".

The part inside the square brackets [] adds a condition to the selection: it only selects nodes whose value of the "nazov" attribute is equal to the name parameter passed to the SinglebookDataByName method, after converting both to lowercase

he code is selecting nodes from an XML document doc using an XPath expression.

nodeListBook is a list of XML nodes that match the conditions specified in the XPath expression.

The expression itself navigates through the XML document to find elements with the following structure:

php

<Bookstore>

<books>

<book>

<nazov>...</nazov>

</book>

</books>

</Bookstore>

The expression then applies a translate() function to the nazov element, converting all uppercase characters to lowercase and replacing them with their corresponding lowercase characters. This is done so that the comparison between the nazov value and the input name is case-insensitive.

Finally, the expression checks if the translate-ed nazov value is equal to the input name. If it is, the node is selected and added to the nodeListBook list.

The selected nodes in nodeListBook will be written to a file with a timestamp and serialized to a JSON string, which will be written to the HTTP response as the output of the method.

public void SinglebookDataByIsbn(string isbn)

his method retrieves information about a single book by ISBN from an XML document "fileBookInfo". It takes a string parameter "isbn" representing the ISBN number of the book.

The method first loads the XML document using the LoadDocument method. If the document is not loaded successfully, it sets the HTTP response status code to 500 and returns. If the "isbn" parameter is not provided, it sets the HTTP response status code to 400 and returns.

The method then searches the XML document for the book with the specified ISBN using an XPath query and stores the result in an XmlNodeList. If no book is found, the HTTP response status code is set to 404 and an error message is returned. If a book is found, the method writes the data to another file using the WriteToTheFileWithTimeStamp method and sets the HTTP response content to the serialized XML node in JSON format with UTF-8 encoding.

The method **SinglebookDataByIsbn** is a **WebMethod** in a web service that returns information about a single book based on the ISBN number provided as input. The method takes one string argument, **isbn**, representing the ISBN of the book to be retrieved.

The method starts by loading an XML document containing information about multiple books using the **LoadDocument** method, and checks if the document was successfully loaded. If the document failed to load, the method sets the HTTP response status code to 500 and sets the status description to "Dokument nemohol byt nacitany".

If the input **isbn** is not provided or is an empty string, the method sets the HTTP response status code to 400 and sets the status description to "Nezadali ste hodnotu mena isbn". The method then writes "Nezadali ste ID" to the HTTP response and returns.

The method uses the loaded XML document to search for books whose ISBN matches the input **isbn**. It uses the **XmlNodeList** class to store the result of the search, and if no match is found, the method sets the HTTP response status code to 404 and sets the status description to "Zadaný záznam sa nenasiel prosím skontrolujte svoj vstup". The method then writes "Pre zadanu hodnotu sa nenasiel ziaden zaznam" to the HTTP response.

If a match is found, the method calls the **WriteToTheFileWithTimeStamp** method and writes the found book information to a file with a timestamp. The method then sets the HTTP response encoding to UTF-8 and writes the information about the book to the HTTP response in JSON format, indented for readability.

GetListAllBooks()

no, samozrejme. Táto metóda slúži na získanie zoznamu všetkých kníh v knižnici.

V prvom kroku vytvára nový objekt **XmlDocument** a pokúša sa načítať súbor s informáciami o knihách. V prípade, že súbor nebol nájdený alebo pri načítaní súboru došlo k chybe, metóda nastaví chybový kód na 500 a popis chyby v odpovedi, ktorú posiela klientovi.

V ďalšom kroku získava zoznam všetkých kníh pomocou metódy **SelectNodes** na objekte **XmlDocument**. V prípade, že zoznam je prázdny alebo neexistuje, metóda opäť nastaví chybový kód na 500 a popis chyby v odpovedi.

Nakoniec, metóda nastavuje UTF-8 sadu pre http odpoveď a serializuje XML uzol ako JSON s formátovaním a posiela ho ako odpoveď klientovi.

Áno, radiče. Táto metóda **GetListAllBooks** je definovaná ako webová metóda a slúži na získanie zoznamu všetkých kníh v knižnici.

V prvom kroku, v rámci tela metódy, vytvára objekt typu **XmlDocument** a nazýva ho **doc**. Potom pokúša sa načítať súbor s informáciami o knihách pomocou metódy **Load** na objekte **doc** a ukladá názov súboru do premennej **fileBookInfo**.

V prípade, že pri načítaní súboru došlo k chybe, metóda zachyti výnimku typu **FileNotFoundException** alebo **XmlException**. V obidvoch prípadoch nastaví chybový kód na 500, aby oznámila klientovi, že požadovaný súbor nebol nájdený alebo že pri spracovaní XML došlo k chybe. Taktiež nastaví popis chyby v odpovedi, ktorý bude poslaný klientovi.

V ďalšom kroku získava zoznam všetkých kníh pomocou metódy **SelectNodes** na objekte **doc**. Táto metóda vráti zoznam uzlov, ktoré zodpovedajú výberu podľa uvedenej Xpath štruktúry. V tomto prípade vyberá knihy z knižnice. Potom kontroluje, či zoznam nie je prázdny a či neobsahuje null hodnotu.

V prípade, že zoznam je prázdny alebo neexistuje, metóda opäť nastaví chybový kód na 500 a popis chyby v odpovedi, aby klientovi oznámila, že pri spracovaní dát v dokumente došlo k chybe.

Nakoniec, metóda nastavuje UTF-8 sadu pre http odpoveď pomocou metódy **BinaryWrite** na objekte **Context.Response** a potom serializuje prvý uzol v zozname kníh pomocou knižnice Json.NET. Serializácia je uskutočnená pomocou metódy **SerializeXmlNode** a formátovanie výstupu je nastavené na formátovanie s odrážkami. Výsledok je zapísaný do odpovede pomocou metódy `Write

Frontend

Gettallbook async

The **Getallbook** function is an asynchronous function in JavaScript, as indicated by the **async** keyword. This function is making an AJAX (Asynchronous JavaScript and XML) request to retrieve a list of books from a web service located at the specified URL ('<http://localhost:8050/book_services.asmx/GetListAllBooks>').

The function uses the **$.ajax** method from the jQuery library to send a POST request to the web service. The response from the web service is expected to be in JSON format, and the **dataType** property is set accordingly.

The **await** keyword is used to ensure that the function will wait for the AJAX request to complete before moving on to the next line of code. The **await** keyword can only be used within an **async** function.

Without the **async** keyword, the function would not be able to use the **await** keyword, and the function would not work as expected. The function would return immediately without waiting for the AJAX request to complete, and the **result** variable would not be assigned the expected data.

In short, the **async** keyword is necessary to ensure that the function will wait for the AJAX request to complete and that the **result** variable will be assigned the expected data.

public static void SaveWebMethodResult

Táto metóda s názvom "SaveWebMethodResult" slúži na ukladanie výsledkov webových volaní (webových služieb) v XML formáte.

Metóda má niekoľko vstupných parametrov, ktoré sú:

* "result" - premenná, ktorá obsahuje výsledok webového volania. Táto premenná je typu "dynamic", čo znamená, že môže mať ľubovoľný typ.
* "methodName" - reťazec, ktorý obsahuje názov webovej metódy, ktorá bola volaná.
* "parameters" - pole reťazcov, ktoré obsahuje parametre, s ktorými bola volaná webová metóda.
* "path" - reťazec, ktorý obsahuje cestu k priečinku, kde sa má výsledok uložiť.

Metóda začína kontrolou, či priečinok, do ktorého sa má výsledok uložiť, existuje. Ak neexistuje, tak sa vytvorí.

Ďalej sa vytvára názov súboru na základe aktuálneho dátumu a času, názvu webovej metódy a parametrov, s ktorými bola volaná. Názov súboru je v tvare "yyyyMMddHHmmss\_methodName\_parameter1\_parameter2\_...\_parameterN.xml".

Potom sa vytvára koreňový element XML dokumentu s názvom "Root".

Metóda "AddToXml" slúži na pridávanie dát do XML dokumentu. Metóda má tri vstupné parametre:

* "parent" - rodičovský element, do ktorého sa majú dáta pridať.
* "obj" - objekt, ktorý sa má pridať do XML dokumentu.
* "parentName" - reťazec, ktorý obsahuje názov rodičovského elementu.

V metóde "AddToXml" sa vytvára nový element s názvom "parentName" a pridáva sa ako potomok rodičovského elementu. Potom sa prechádza cez vlastnosti objektu a ak je vlastnosť hodnota typu primitívneho typu alebo typu "string", tak sa pridá do elementu ako nový potomok. Ak je vlastnosť hodnotou typu "IEnumerable" (teda môže obsahovať viacero hodnôt) a nie je typu "string", tak sa pre každú hodnotu v danej vlastnosti zavolá metóda "AddToXml" s aktuálnym názvom vlastnosti ako názvom rodičovského elementu. Ak vlastnosť obsahuje iný objekt, tak sa tiež zavolá metóda "AddToXml" s aktuálnym názvom v

last sentence: lastnosti ako názvom rodičovského elementu a objekt sa rekurzívne pridá do XML dokumentu.

Potom sa v hlavnej metóde "SaveWebMethodResult" kontroluje, či výsledok webového volania obsahuje viacero hodnôt (typ "IEnumerable") a nie je typu "string". Ak áno, tak sa pre každú hodnotu v danej premennej zavolá metóda "AddToXml" s koreňovým elementom ako rodičovským elementom. Ak je výsledok webového volania iba jedna hodnota, tak sa zavolá metóda "AddToXml" s touto hodnotou ako parameterom.

Nakoniec sa vytvorí úplná cesta k súboru a XML dokument sa uloží pomocou metódy "Save" na danú cestu.

Táto metóda teda slúži na konverziu výsledkov webových volaní na formát XML a ich uloženie do súboru. Vďaka tomu môžu byť tieto výsledky neskôr ľahko spracované a analyzované pomocou rôznych nástrojov a programov.

This is a method in C# that takes in a result from a web method, method name, parameters, and a file path. It serializes the result to an XML file with the current date and time, method name, parameters, and data. Let's take a closer look at the implementation.

csharpCopy code

public static void SaveWebMethodResult(dynamic result, string methodName, string[] parameters, string path)

This method is public, static, and returns void. It takes in four parameters: a dynamic result, a string representing the method name, a string array of parameters, and a string representing the path where the file should be saved.

luaCopy code

if (!Directory.Exists(path)) { Directory.CreateDirectory(path); }

If the directory where the file is supposed to be saved does not exist, this code will create it.

vbnetCopy code

string fileName = DateTime.Now.ToString("yyyyMMddHHmmss") + "\_" + methodName + "\_" + string.Join("\_", parameters) + ".xml";

This code creates a file name based on the current date and time, the method name, and the parameters passed in. The format of the file name is **yyyyMMddHHmmss\_methodName\_parameter1\_parameter2\_...\_parameterN.xml**.

javaCopy code

XElement root = new XElement("Root");

This code creates a new **XElement** object with the root element name "Root".

csharpCopy code

void AddToXml(XElement parent, object obj, string parentName="data")

This code declares a new method called "AddToXml" that takes in an **XElement** parent, an object, and an optional string called "parentName" that defaults to "data".

javascriptCopy code

var element = new XElement(parentName);

This code creates a new **XElement** object with the name passed in as "parentName".

csharpCopy code

foreach (var prop in obj.GetType().GetProperties()) { var value = prop.GetValue(obj); if (value != null) { if (prop.PropertyType.IsPrimitive || prop.PropertyType == typeof(string)) { element.Add(new XElement(prop.Name, value)); } else if (value is IEnumerable enumerable1 && !(value is string)) { foreach (var item in enumerable1) { AddToXml(element, item, prop.Name); // pass current property name as parent name } } else { AddToXml(element, value, prop.Name); // pass current property name as parent name } } }

This code loops through each property of the object and creates an **XElement** object for each one. If the property is a primitive or a string, it creates a new **XElement** object with the name of the property and its value. If the property is an **IEnumerable** (such as an array or a list), it recursively calls the **AddToXml** method with the current **XElement** object as the parent and the item in the enumerable as the object, passing the property name as the parent name. If the property is neither primitive nor an **IEnumerable**, it recursively calls the **AddToXml** method with the current **XElement** object as the parent and the property value as the object, passing the property name as the parent name.

scssCopy code

parent.Add(element);

This code adds the current **XElement** object to its parent.

csharpCopy code

if (result is IEnumerable enumerable && !(result is string)) { foreach (var item in enumerable) { AddToXml(root, item); } } else { AddToXml(root, result); }

This code checks if the result is an **IEnumerable** (such as an array or a list)

and if it is, it loops through each item in the enumerable and recursively calls the **AddToXml** method with the **root** element as the parent and the item as the object. If the result is not an **IEnumerable**, it simply calls the **AddToXml** method with the **root** element as the parent and the result as the object.

luaCopy code

string fullPath = Path.Combine(path, fileName); root.Save(fullPath);

This code combines the file path and file name to create the full path to where the XML file should be saved. It then calls the **Save** method on the **root** element, which saves the element and all of its child elements to an XML file at the specified path.

Overall, this method is useful for serializing a web method result to an XML file for later use. It handles primitive data types, arrays, lists, and objects with nested properties. The resulting XML file is named in a standardized way that includes the date and time, method name, parameters, and data, making it easy to identify and manage. The method also creates the necessary directories if they do not already exist.