# ابزارهای تولید و ویرایش متنهای xml (WorkXMLUtils): 1- تولید متن xml

در توابع و متدهای برنامه نویسی سیستم مدیریت فرایندها، در موارد زیادی باید داده های ساختاریافته را به فرم xml و به عنوان پارامتر به این توابع پاس کنیم. برای مثال یکی از پرکاربردترین این موارد، جستجوی کارهاست که با استفاده از تابع SearchWork انجام می شود. به یک نمونه از این تابع توجه کنید:

function WorkActionRepSupplier() {

var data='<\_RQST WorkflowCode="1" Status="15" >';

data+='<Term ScriptName="supplier" LOp="AND" Op="=" Value="' + WorkState1.suplier + '" />';

data+='</\_RQST>';

var list=WorkUtils.SearchWork("", data);

return JSON.stringify(list);

}

همانطور که ملاحظه می کنید، عبارتها و عملگرهای جستجو به شکل ساختاریافته و در قالب یک متن xml بنام data به تابع SearchWork پاس شده است. این مثال اگرچه یک جستجوی بسیار ساده است اما در همین مثال ساده نیز پیچیدگیهای متن xml از لحاظ رعایت علائم و کاراکترهای ‘, “”, <, > تا حدودی مشخص است. یک اشتباه کوچک در عدم رعایت قواعد استفاده از این علائم و یا عدم استفاده از یک فاصله خالی، اجرای تابع را با خطا مواجه می کند ضمن این که ردیابی و رفع این خطاها بسیار خسته کننده و زمانبر می باشد. لذا برای تولید سیستماتیک فایلهای xml یک Object با نام WorkXMLUtils پیاده سازی شده است که با استفاده از متدها و ویژگی (Property) های این Object ، می توانید یک فایل xml را تولید کنید و یا عناصر (Element) و مشخصه (Attribute) های آن را ویرایش کنید. این متدها بطور اتوماتیک، علائم و دستور (Syntax) استفاده از این علائم را رعایت می کنند و به این ترتیب، فایل xml به شکل صحیح و بدون خطا تولید و یا ویرایش می شود. در این مستند تنها به تولید متنهای xml پرداخته شده است و بحث خواندن و ویرایش این متنها در مستندات دیگر توضیح داده خواهد شد.

برای شرح دقیق تر توابع تولید xml ، در ادامه مطلب نحوه استفاده از این توابع در چند مثال، بصورت عملی توضیح داده شده است.

## مثال 1: تولید متن xml برای یک جستجوی ساده در کارهای گردش کار

فرض کنید، می خواهیم کارهای فعال گردش کار 1 (خرید کالاها) را بازیابی کنیم با این شرط که نام تامین کننده آنها محتوای فیلد تامین کننده (supplier) در کار جاری باشد. این مثال یک جستجوی ساده و دارای یک شرط روی یک فیلد است. برای این کار باید یک متن xml به شکل زیر تولید کنید:

<\_RQST WorkflowCode="1" Status="15" >

<Term ScriptName="supplier" LOp="AND" Op="=" Value="' + WorkState1.suplier + '" />

</\_RQST>

## مراحل تولید متن xml

برای تولید متن xml مثال بالا باید پاراگراف زیر را در محیط برنامه نویسی درج کنید:

**var xmlStr="<\_RQST/>";**

**var xmlDoc=WorkXMLUtils.CreateDocument();**

**xmlDoc.LoadXML(xmlStr);**

**var rootNode=xmlDoc.DocumentElement;**

**rootNode.SetAttribute ("WorkflowCode", 1);**

**rootNote.SetAttribute ("Statue", 15);**

**var node1=xmlDoc.CreateElement ("Term");**

**rootNode.AppendChild ("node1");**

**node1.SetAttribute ("ScriptName", "supplier");**

**node1.SetAttribute ("LOp", "AND");**

**node1.SetAttribute ("Op", "=");**

**node1.SetAttribute ("Value", WorkState2.orders );**

**var data=xmlDoc.XML;**

**var list=WorkUtils.SearchWork("", data);**

این پاراگراف برای آشنایی کلی و امکان مراجعه کاربران در اینجا درج شده است. برای آشنایی بیشتر با این توابع و متدها، در ادامه مطلب، مراحل مختلف این پاراگراف به ترتیب و سطر به سطر توضیح داده شده است:

### ایجاد متن xml

در این مرحله، با تعیین عنصر ریشه (Root Element) ، متن xml مورد نظر خود را به شکل حداقلی ایجاد می کنیم. عنصر ریشه (Root Element) در مثال ما \_RQST است که در بالا با رنگ قرمز مشخص شده است. برای این کار کافی است یک رشته حرفی به شکل زیر بسازیم:

**var xmlStr="<\_RQST />";**

در واقع در این مرحله ما سطح پایه یا ریشه متن xml را ایجاد می کنیم. مشخصه های ریشه و تمام عناصر و مشخصه های بعدی از طریق برنامه نویسی و به شکل فرزند (Child) برای این عنصر انجام خواهد شد.

**توضیح**: در اینجا برای راحتی کار و پرهیز از اشتباهات سهوی دستوری از فرمت خلاصه <\_RQST /> استفاده شده است. این عبارت معادل </\_RQST><\_RQST> است.

### ایجاد یک DOMDocument

همانطور که می دانید یک فایل xml یک فایل متنی با ساختار درختی با لایه ها یا سطوح متعدد است که از یک سری عنصر Element که سرشاخه های درخت هستند به اضافه یک سری مشخصه Atrribute که مشخصات Element ها هستند تشکیل می شود. حال برای امکان دسترسی و ویرایش متن xml از طریق برنامه نویسی، نیاز به یک مدل واسط بنام DOM(Document Object Model) داریم که از عناصر و مشخصه های موجود در xml یک Object تولید می کند و به این وسیله نقش یک API را برای امکان دسترسی به ساختار xml فراهم می کند. در ابزارهای برنامه نویسی سیستم مدیریت فرایندها، متدی که برای این کار استفاده می شود، CreateDocument است. بنابراین دومین سطر برنامه به شکل زیر خواهد بود:

**var xmlDoc=WorkXMLUtils.CreateDocument();**

حال ما یک متغیر بنام xmlDoc داریم که در واقع یک Object است که متن xml در مراحل بعدی، در این Object، فراخوانی خواهد شد.

### خواندن متن xml در xmlDoc

حال که متن xml ایجاد شده است، باید این متن توسط xmlDoc خوانده شود تا امکان ویرایش و توسعه آن را داشته باشیم:

**xmlDoc.LoadXML(xmlStr);**

### تعیین مشخصه (Attribute) های عنصر ریشه

اگر به مثال مراجعه کنید، می بینید که عنصر ریشه دارای دو مشخصه WorkflowCode=”1” و Status=”15” است. برای تعیین مشخصه های فوق، از متد SetAttribute به ترتیب زیر استفاده می کنیم:

**var rootNode=xmlDoc.DocumentElement ;**

**rootNode.SetAttribute ("WorkflowCode", 1);**

**rootNote.SetAttribute ("Statue", 15);**

با توجه به اینکه عنصر ریشه DocumentElement نام دارد لذا در سطر اول از این بخش، عنصر ریشه در یک متغیر بنام rootNode قرار می گیرد و در سطرهای دوم و سوم، مشخصه ها یک به یک به این عنصر افزوده می شود.

### تعریف یک عنصر با نام Term برای وارد کردن شرط جستجو

مقادیر مورد نظر فیلدها برای جستجو و نیز عملگرهای جستجو بصورت مشخصه های یک عنصر با نام Term تعیین می شود بنابراین باید ابتدا عنصر Term را ایجاد کنیم. ایجاد عنصر توسط متد CreateElement و به شکل زیر انجام می شود:

**var node1=xmlDoc.CreateElement ("Term");**

**نکته مهم:** ایجاد یک عنصر (Element) توسط متد CreateElement به معنی افزوده شدن آن به متن xml نیست و افزودن آن در مرحله بعد انجام می شود.

### افزودن عنصر Term به شکل فرزند (Child)برای عنصر ریشه

در این مرحله با استفاده از متد AppendChild می توانیم عنصر Term را بصورت فرزند یا زیرشاخه عنصر ریشه، به متن xml اضافه کنیم:

**rootNode.AppendChild ("Term");**

**توضیح:** هر عنصر Term برای یک شرط بکار می رود و لذا برای جستجوهای پیچیده تر و ترکیبی که شامل شرطها و فیلدهای متعدد هستند باید Termهای متعدد ایجاد کنید. این مورد در مثالهای بعدی توضیح داده شده است.

### افزودن مشخصه ها به عنصر Term

در این مرحله مشخصه های لازم برای جستجو در مقادیر فیلد تعیین و اضافه می شود. این مشخصه ها عبارتند از :

ScriptName : نام در برنامه فیلد

LOpعملگر بین خطوط:

Opعملگر جستجو در سطر:

Valueمقدار فیلد :

در مرحله شماره 5، که عنصر Term ایجاد شد، این عنصر به شکل یک WorkDOMNode در یک متغیر بنام node1 قرار گرفت. لذا از همین متغیر می توانیم برای افزودن مشخصه ها استفاده کنیم.

**node1.SetAttribute ("ScriptName", "supplier");**

**node1.SetAttribute ("Lop", "AND");**

**node1.SetAttribute ("Op", "=");**

**node1.SetAttribute ("Value", WorkState2.orders );**

با این ترتیب تولید این xml پایان می یابد.

**توضیح**: همانطور که در این مثال هم نشان داده شد، پارامتر دوم در تابع SetAttribute (name, value) برای تعیین مقدار مشخصه ها بکار می رود، اگر این پارامتر رشته حرفی باشد باید بین دو کاراکتر نقل قول (“) قرار بگیرد در غیر این صورت و برای مقادیر عددی و منطقی و نیز برای مقدار فیلدهای قابل تعریف، نیازی به این کاراکتر نیست. این قابلیت یکی از مزایای استفااده از این توابع می باشد.

### استخراج متن xml و پاس کردن آن به عنوان پارامتر

و درنهایت باید محتوای xml را از xmlDoc استخراج کنیم و این متن را به عنوان پارامتر به تابع مورد نظر (در این مثال SearchWork) پاس کنیم. برای این کار باید سطر زیر را به برنامه اضافه کنیم:

**var data=xmlDoc.XML;**

**var list=WorkUtils.SearchWork("", data);**

توضیح این که یکی از ویژگی (property) های xmlDoc ویژگی XML است و خروجی آن، محتوای متنی xml می باشد که به این ترتیب می توانیم این متن را استخراج کنیم و با قرار دادن در یک متغیر مثل data ، آن را به تابع مورد نظر پاس کنیم.

## مثال 2: یک جستجوی ترکیبی در کارهای گردش کار

همانطور که در مثال قبل گفته شد، بخشی از شرطهای مورد نظر برای جستجو، در قالب مشخصه های عنصر Term در متن xml درج می­شود. بنابراین برای جستجوی ترکیبی باید این عنصر را تکرار کنید. برای مثال فرض کنید جستجوی قبلی را می خواهید محدودتر کنید و مثلا کالاهای با نام تامین کننده خاص و قیمت بیش از 10000 ریال را جستجو کنید. در این مثال، شرط قیمت بیشتر از 10000 به شکل زیر به متن xml اضافه می شود:

<\_RQST WorkflowCode="1" Status="15" >

<Term ScriptName="supplier" LOp="AND" Op="=" Value="' + WorkState1.suplier + '" />

<Term ScriptName="price" LOp="AND" Op=">" Value="10000" >

</\_RQST>

حال در برنامه گردش کار برای افزودن یک عنصر فرزند دیگر با نام Term به عنصر ریشه، اضافه کنید. به این ترتیب، برنامه به شکل زیر تغییر می کند:

**var xmlStr="<\_RQST/ >";**

**var xmlDoc=WorkXMLUtils.CreateDocument();**

**xmlDoc.LoadXML(xmlStr);**

**var rootNode=xmlDoc.DocumentElement;**

**rootNode.SetAttribute ("WorkflowCode", 1);**

**rootNote.SetAttribute ("Statue", 15);**

**var node1=xmlDoc.CreateElement ("Term");**

**rootNode.AppendChild (node1);**

**node1.SetAttribute ("ScriptName", "supplier");**

**node1.SetAttribute ("LOp", "AND");**

**node1.SetAttribute ("Op", "=");**

**node1.SetAttribute ("Value", "شرکت الف" );**

**var node1=xmlDoc.CreateElement ("Term");**

**rootNode.AppendChild(node1);**

**node1.SetAttribute ("ScriptName", "price");**

**node1.SetAttribute ("LOp", "AND");**

**node1.SetAttribute ("Op", ">");**

**node1.SetAttribute ("Value", 10000);**

**var data=xmlDoc.XML;**

**var list=WorkUtils.SearchWork("", data);**

سطرهایی که در بالا به رنگ آبی است برای افزودن شرط دوم جستجوی ترکیبی به برنامه افزوده شده است. به این ترتیب، متن xml به شکل زیر تولید می شود:

<\_RQST WorkflowCode="1" Statue="15">

<Term ScriptName="supplier" LOp="AND" Op="=" Value="شرکت الف"/>

<Term ScriptName="price" LOp="AND" Op="&gt;" Value="10000"/>

</\_RQST>

**نکته مهم:** اتفاق بسیار جالبی که در اینجا افتاده این است که متد SetAttribute بطور اتوماتیک کاراکتر بزرگتر (<) را تبدیل به &gt; کرده است. علت آن هم این است که کاراکتر (<) در ساختار xml معنی دار است و برای بستن Tag ها استفاده می شود لذا استفاده از این کاراکتر برنامه را با خطا روبرو خواهد کرد. البته این قابلیت محدود به علامت بزرگتر نیست و برای تمام کاراکترهای خاص این تبدیل انجام می شود.

### تبدیل کاراکترهای خاص در xml

توابع تولید xml بطور اتوماتیک کاراکترهای خاص زیر را به عبارتهای معادل (entity reference) آنها تبدیل می کند:

|  |  |
| --- | --- |
| تبدیل شده در متن xml | کاراکتر |
| &lt; | Less than (<) |
| &gt; | Greater than (>) |
| &quote; | Double quote (") |
| &apos; | Apostrophe (ˋ) |
| &amp; | Ampersand (&) |

این قابلیت یکی دیگر از مزایای بسیار مفیدی است تولید متنهای xml شامل این کاراکترها را بسیار تسهیل می کند. این مورد بخصوص برای کاراکتر نقل قول (“) بسیار برجسته است. برای مثال فرض کنید در یک مشخصه با محتوای حرفی مثلا خلاصه کتابها، بعضی از عبارات برای تاکید بین دو علامت نقل قول قرار گرفته باشد در این صورت سیستم این کاراکترها را تشخیص می دهد و به شکل اتوماتیک تبدیل به &quote; می کند. در غیر این صورت، علامت نقل قول محتوای این مشخصه را با اختلال روبرو خواهد کرد.

## مثال 3: جستجو در فیلدهای از نوع آرایه، در کارهای گردش کار

همانطور که می دانید فیلدهای از نوع آرایه، شامل تعدادی فیلد زیرمجموعه هستند که اجزای آرایه نامیده می شوند. به این ترتیب، در متن xml، اجزای آرایه به شکل فرزند فیلد آرایه درج می شود. برای مثال فرض کنید فیلد تامین کننده ها (suppliers) به شکل آرایه و دارای یک جزء بنام supplierName است و می خواهیم جستجو را در این فیلد انجام دهیم به این شکل که تمام کارهای خرید کالا که نام تامین کننده آن **شرکت نمونه** است، بازیابی شود. در این صورت متن xml باید به شکل زیر تولید شود:

<\_RQST WorkflowCode="1" Statue="15">

<Term ScriptName="suppliers" LOp="AND">

<Term ScriptName="supplierName" LOp="AND" Op="=" Value=" "شرکت نمونه />

</Term>

</\_RQST>

برنامه تولید این xml به شکل زیر خواهد بود:

**var xmlStr="<\_RQST/ >";**

**var xmlDoc=WorkXMLUtils.CreateDocument();**

**xmlDoc.LoadXML(xmlStr);**

**var rootNode=xmlDoc.DocumentElement;**

**rootNode.SetAttribute ("WorkflowCode", 1);**

**rootNote.SetAttribute ("Statue", 15);**

**var node1=xmlDoc.CreateElement ("Term");**

**rootNode.AppendChild ("node1");**

**node1.SetAttribute ("ScriptName", "suppliers");**

**node1.SetAttribute ("LOp", "AND");**

**var node2=xmlDoc.CreateElement("Term");**

**node1.AppendChild (node2);**

**node2.SetAttribute ("ScriptName", "supplierName");**

**node2.SetAttribute ("LOp", "AND");**

**node2.SetAttribute ("Op", "=");**

**node2.SetAttribute ("Value", "شرکت نمونه" );**

**var data=xmlDoc.XML;**

**var list=WorkUtils.SearchWork("", data);**

## مثال 4: ایجاد یک کار جدید با فیلدهای دلخواه (StartWork)

فرض کنید در یک گردش کار خاص می خواهیم یک کار جدید از طریق برنامه ایجاد کنیم. در سیستم مدیریت فرایندها ایجاد کار جدید با استفاده از متد StartWork انجام می شود. پارامتری که به شکل یک متن xml به این تابع پاس می شود دارای الگوی زیر است:

<\_RQST WorkflowKey="xxx" WorkflowCode="xxx"

StartTaskKey=”xxx” StartTaskCode=”xxx”

DOAddWizardKey="xxx" DOAddWizardCode="xxx"

DODescription="xxx"

ContentKindKey="xxx" ContentKindCode="xxx"

StartTaskNote="xxx">

<WorkflowFieldInfs>

<WFI ScriptName="xxx" FieldValue="xxx" />

:

</WorkflowFieldInfs>

</\_RQST>

به یک مثال در این مورد توجه کنید:

<\_RQST WorkflowCode="12" DOAddWizardCode="18" DODescription="اصلاح اطلاعات مشتریان"<

<WorkflowFieldInfs>

<WFI ScriptName="customer" FieldValue="شرکت الف" />

<WFI ScriptName="address" FieldValue="تهران-خیابان عارف" />

<WFI ScriptName="tel" FieldValue="76543210" />

</WorkflowFieldInfs>

</\_RQST>

با استفاده از برنامه زیر، می توانید این xml را تولید کنید:

**var xmlStr="<\_RQST/ >";**

**var xmlDoc=WorkXMLUtils.CreateDocument();**

**xmlDoc.LoadXML(xmlStr);**

**var rootNode=xmlDoc.DocumentElement;**

**rootNode.SetAttribute ("WorkflowCode", 12);**

**rootNode.SetAttribute ("DOAddWizardCode", 18);**

**rootNode.SetAttribute ("DODescription", "اصلاح اطلاعات مشتریان")**

**var node1=xmlDoc.CreateElement ("WorkflowFieldInfs");**

**rootNode.AppendChild (node1);**

**var node2=xmlDoc.CreateElement("WFI");**

**node1.AppendChild (node2);**

**node2.SetAttribute ("ScriptName", "customer");**

**node2.SetAttribute ("FieldValue", "شرکت الف" );**

**var node2=xmlDoc.CreateElement ("WFI");**

**node1.AppendChild (node2);**

**node2.SetAttribute ("ScriptName", "address");**

**node2.SetAttribute ("FieldValue", "تهران-خیابان عارف" );**

**var node2=xmlDoc.CreateElement ("WFI");**

**node1.AppendChild (node2);**

**node2.SetAttribute ("ScriptName", "tel");**

**node2.SetAttribute ("FieldValue", "76543210");**

**var data=xmlDoc.XML;**

**var wKey=WorkUtils.StartWork("", data);**

با این چهار مثال، مدل کلی تولید xml توسط توابع برنامه توضیح داده شد. موارد دیگر نیز کم و بیش شبیه به این مدل هستند و برای این موارد می توانید از این مثالها الگوبرداری کنید.