اکشن ها

- انواع اکشن ها
- آپدیت کردن صفحات بدون ایجاد صفحه جدید

انواع اکشن ها

اکشن ها دستوارت از پیش تعریف شده ای هستند که در صورت صدا کردن آن ها کار خاصی انجام میدهند.

شما میتوانید داخل بعضی از خواص المانها از اکشنها استفاده کنید که این المانها در فصل قبل بررسی شدند. همچنین میتوان با عملگرهای `->`، `;` و `^` چند اکشن را به صورت ترتیبی، موازی ای همراه با مسیر مدیریت خطا اجرا کرد و با پرانتز ترتیب اجرا را مشخص نمود.

زبان اپسان در حال حاظر از پانزده اکشن مختلف پشتیبانی می کند که بعضی از این اکشن ها مقدار ورودی دارند که با استفاده از " / " و سپس نوشتن ورودی مشخص می شوند.

ترکیب و کنترل جریان اکشنها

یک عبارت اکشن میتواند از چند اکشن و چهار ساختار زیر تشکیل شود:

مثال	رفتار	ساختار
`ac1->ac2`	اجرای ترتیبی: اکشن سمت راست فقط پس از پایان موفق اکشن سمت چپ اجرا میشود و نتیجه اکشن قبل را در `$data` دریافت میکند.	`->`
`ac1;ac2`	اجرای موازی: هر دو شاخه بدون منتظر ماندن برای دیگری شروع میشوند و به نتیجه یکدیگر دسترسی ندارند.	`;`
`ac1^handler`	مسیر مدیریت خطا: اکشن سمت راست فقط در صورت شکست اکشن سمت چپ اجرا میشود. تعریف این مسیر اختیاری است.	`^`
`(ac1;ac2)->ac3`	گروه‌بندی: ترتیب ارزیابی و محدوده اجرای موازی ای مدیریت خطا را مشخص میکند.	`(...)`

اولویت عملگرها

اولویت از بیشترین به کمترین به این ترتیب است: `()`، سپس `^`، سپس `->` و در آخر `;`. بنابراین عبارت `ac1;ac2->ac3` به معنی اجرای موازی `ac1` با زنجیره `ac2->ac3` است. برای اینکه `(ac1;ac2)->ac3` منتظر پایان هر دو اکشن بماند باید بنویسید `ac3`.

اجرای ترتیبی و انتقال نتیجه

در `ac1->ac2`، اکشن `ac2` تا پایان `ac1` اجرا نمی‌شود. اگر `ac1` نتیجه‌ای تولید کند، آن نتیجه از طریق `$data` در اختیار `ac2` قرار می‌گیرد.

```
<button text="Select image"
        onClick="imagepicker/gallery->script/image:showResult($data)" />
```

اجرای موازی و نتیجه گروه

در `ac1;ac2` هر شاخه ورودی اولیه یکسانی دریافت می‌کند، ولی نتیجه یک شاخه برای شاخه دیگر قابل مشاهده نیست. در `(ac1;ac2)->ac3`، اکشن `ac3` پس از پایان موفق هر دو شاخه اجرا می‌شود و نتایج را در `$data` دریافت می‌کند. ترتیب آرایه مطابق ترتیب نوشتن شاخه‌ها است و برای اکشنی که نتیجه ندارد مقدار `null` قرار می‌گیرد.

مدیریت خطا با `^`

در `ac1^ha1->ac2^ha2`، اگر `ac1` شکست بخورد، `ha1` اجرا می‌شود و `ac2` اجرا نخواهد شد. اگر `ac1` موفق باشد، `ha1` در صورت شکست `ac2` نیز اجرا می‌شود؛ خطای اکشن اصلی را به موفقیت تبدیل نمی‌کند و اکشن‌های بعدی زنجیره به `ac2` می‌رسد؛ اجرای موفق handler خطا `ha2` اجرا نمی‌شوند. برای یک زنجیره چندمرحله‌ای از پرانتز استفاده کنید:

```
ac1^(ha1->ha2)->ac3
```

اگر `ac1` شکست بخورد، زنجیره `ha1->ha2` اجرا می‌شود، اما `ac3` اجرا نخواهد شد. در `(ac1;ac2)^handler` ابتدا پایان همه شاخه‌های موازی بررسی می‌شود و سپس در صورت وجود خطا، `handler` فقط یک بار اجرا می‌شود.

handler اطلاعات خطا را به صورت یک object در `$data` دریافت می‌کند:

```
{
  "expression": "ac1",
  "errors": [
    {
      "branchPath": [0],
      "action": "ac1",
      "type": "ExceptionType",
      "message": "Error message",
      "input": null
```

```
    }
  ]
}
```

پرانتز و حالت‌های توو در توو

- `ac1->(ac2;ac3)`: پس از موفقیت `ac1`، دو اکشن بعدی به صورت موازی اجرا می‌شوند.
- `(ac1;ac2)->ac3`: `ac3` منتظر پایان موفق هر دو شاخه می‌ماند.
- `(ac1;ac2)^handler`: خطای کل گروه پس از پایان همه شاخه‌ها به handler ارسال می‌شود.
- برای handler توو در توو باید پرانتز صریح به کار رود؛ نوشتن `ac1^ha1^ha2` معتبر نیست.

استفاده از کاراکترهای عملگر در ورودی اکشن

اگر یکی از کاراکترهای رزروشده جزئی از ورودی اکشن است، آن را با `\` escape کنید:

ورودی	مقدار دریافت‌شده توسط اکشن
`\;`	`;`
`\^`	`^`
`\(`	`(`
`\)`	`)`
`\->`	`->`
`\\`	یک backslash

پرانتزهای داخل ورودی یک اکشن، مانند `script/x:fn($data)`، به صورت خودکار جزئی از همان اکشن محسوب می‌شوند. فقط اگر خود اکشن باید با پرانتز باز شروع شود از `\(` استفاده کنید. توالی‌های ناشناخته مانند `\d` بدون تغییر باقی می‌مانند.

> عبارت‌هایی با پرانتز ناقص، عملگر بدون اکشن، گروه خالی یا handler زنجیرهای بدون پرانتز معتبر نیستند. در API قدیمی خطا ثبت می‌شود و هیچ بخش ناقصی از عبارت اجرا نخواهد شد.

---

command :

این اکشن دستوری را به سمت سرور ارسال می کند تا صفحه ای دیالوگ جدیدی را بارگیری (load) کند. مقدار ورودی ای که در سمت سرور نیز به شما می رسد پس از " / " در پایان command می آید.
به عنوان مثال:

```
<page>
    <button text="goto page 2" onClick="command/page2">
</page>
```

در این مثال دستور page2 برای سرور ارسال شده و سرور موظف است صفحه ی بعد را ارسال کند.
می توانید هر چیزی می خواهید باشد و دقیقا همان مقدار سمت سرور رفته و شما زمانی مقدار ورودی
صفحه ی بعد را با استفاده از آن مشخص و ارسال کنید.

• App :

این اکشن برای کنترل های درون برنامه استفاده می شود به طور مثال برای بستن برنامه.

ورودی های ممکن شامل موارد زیر است:

↵ end :

این کد ، اپسان شما را می بندد.

```
<page>
    <button text="exit the app" onClick="app/end">
</page>
```

↵ Back :

بدون در نظر گرفتن کدهای داخل المان ها به صفحه ی قبل بر میگردد و در صورت نبود صفحه ی
قبل برنامه را می بندد.

```
<page>
    <button text="return to previous page" onClick="app/back">
</page>
```

↵ BackKey :

عملیات باز گشت توسط کاربر شروع میکند و کاملا مشابه فشرده شدن کلید بازگشت از
است. در صورتی که در page خاصیت onback مشخص کرده باشید این خاصیت صدا می شود.

↵ closedialog :

عملیات بستن دایالوگی را که باز است انجام میدهد.

↵ closesnack :

عملیات بستن اسنک بار را که باز است انجام میدهد.

↵ result :

سپ ه ک دن ک ی م ص خ ش م ی ن و ن ک گ و ل ا ی د ای action دوش ی م ل ا س ر ا ه ج ی ت ن ن ا و ن ع ه ب ن آ ی ه د ن ن ا و خ ا ر ف
ی ا ر ب ا ر ی ا ه ج ی ت ن page رد ن آ ن د ش ه ت س ب ز ا

dialog/d1- ن ش ک ا ز ا ه د ا ف ت س ا ز ا س پ دوش ز اب app/result ع ب ا ت رد ه د ش ه د ا د ر ا د ق م
ن ش ک ا ب گ و ل ا ی د ک ی ر گ ا ل ث م (data$)script/s1:showResult>

showResult ت س ا ت ف ا ی ر د ل ب ا ق

• Dialog :

id ن ش ک ا ن ی ا ی د و ر و. د ن ک ی م ا ر ج ا ر د ی ا ه ت ش و ن د و خ د ک ل خ ا د ا م ش ه ک (offline) ن ی ا ل ف آ گ و ل ا ی د ک ی
ت س ا م ش ر ظ ن د ر و م گ و ل ا ی ا د.

```
<page>
    <button text="opening a dialog" onClick="dialog/dialog4">
</page>

<dialog borderCornerRadius="3" id= "dialog4">
  <frame layout="linear_vertical" width="75" height="75" gravity="center">
    <image src="http://162.55.105.137:8080/api/public/dl/IPI8gq4o" width="match_parent"
selfgravity="center"></image>
  </frame>
  <frame layout="linear_vertical">
    <text renderas="html" textcolor="#E64724" text="<b>ب ه ا س پ ا ن خ و ش آ م د ی د<\b>" textsize="18"
textalignment="center"></text>
  </frame>
  <button text="د ی ا ت" borderCornerRadius="3" background="#E64724"
onClick="app/closedialog"></button>
</dialog>
```

• Page :

د ن ک ی م ا ر ج ا ر د ی ا ه ت ش و ن د و خ د ک ل خ ا د ا م ش ه ک ن ی ا ل ف آ ه ح ف ص ک ی.

و ر و د ی ا ی ن ا ک ش ن id ص ف ح ه د ر و م ن ظ ر ش م ا س ت.

```
<page title="اپسان" background="#FFFFFF" toolbarColor="#E64724" width="match_parent"
height="match_parent">
  <frame  width="match_parent" layout="linear_vertical" height="match_parent"
background="#FFF" >
      <button textcolor="#FFDDCE"  background="#E64724" borderCornerRadius="10"  width="250"
onClick="page/nextPage">صفحه بعد</button>
  </frame>
</page>
```

<page title="اپسان" background="#DDDDDD" toolbarColor="#E64724" width="match_parent" height="match_parent" id="nextPage" > <frame margin="center" width="wrap_content" height="770"> <text textcolor="E64724" textsize="20" margin_bottom="250" >"اپلیکیشن اپسان"</text> </frame> </page>

- Call :

با شماره ای که به عنوان ورودی دریافت میکند با خط تلفن همراه مخاطب تماس برقرار می کند.

```
<page>
    <button text="press to call" onClick="call/09xxxxxxxx">
</page>
```

- Share :

عملیات اشتراک گذاری را انجام میدهد.

```
<page>
    <button text="press to share" onClick="share/متن قابل اشتراک">
</page>
```

- ## View :

برای نمایش فایل ها داخل سیستم عامل با دیگر ام ای ها استفاده میشود.

```
<page>
    <button text="press to view" onClick="view/fileAddress">
</page>
```

آدرس فایل باید به صورت content://filepath باشد. در صورتی که فایل توسط مینی اپ شما دانلود شده باشد آدرس فایل پس از دانلود شدن آن باز میگردد. برای اطلاعات بیشتر به بخش دانلود مراجعه کنید.

-
- ## viewbase64 :

> این اکشن از ورژن `appsan:0.1.73` در دسترس است.

به توسعه‌دهندگان اجازه می‌دهد تا داده‌های کدگذاری شده با Base64 را به عنوان فایل باز کنند. این عملکرد از قالب `viewbase64` استفاده می‌کند که داده‌ها را به صورت Base64 به همراه نوع MIME مشخص می‌کند. در ادامه توضیحات این عملکرد آورده شده است:

نحوه استفاده از `viewbase64`

هنگامی که از عملکرد `viewbase64` استفاده می‌کنید، باید یک رشته داده معتبر را به عنوان ورودی به این عملکرد بدهید. این رشته باید شامل نوع MIME و داده‌های Base64 باشد.

مثال‌هایی از استفاده:

- داده‌های یک فایل PDF به صورت Base64:

```
<page>
<button text="show pdf"
onClick="viewbase64/application/pdf,JVBERi0xLjUKJdDUxdgK...">
</page>
```

نکات:

- در صورت بروز خطا در حین پردازش داده‌ها، این خطاها ثبت و اخطار داده شد اما عملکرد متوقف نمی‌شود.
- اگر نوع MIME مشخص نشده باشد، فایل به صورت `text/plain` ذخیره و باز می‌شود.

Snackbar :

یک اسنک بار با آیدی مشخص را نشان می‌دهد.

```
<page>
    <button text="show snackbar" onClick="snackbar/welcomeSnackbar">
</page>
```

Navigate :

موقعیت جغرافیایی با فرمت lat,long را ورودی می‌گیرد که lat و long به ترتیب طول و عرض جغرافیایی به صورت عدد اعشاری هستند.

این اکشن مقدار ورودی را روی نرم افزار نقشه نصب شده روی گوشی کاربر نمایش می‌دهد.

```
<page>
    <button text="show on map" onClick="navigate/35.8029459">
</page>
```

---

vStream :

یک پنجره تمام صفحه برای پخش فیلم باز کرده و لینکی که در ورودی می‌گیرد را پخش می کند.

```
<page>
    <button text="opening media" onClick="vstream/http://yourURL">
</page>
```

Update :

همانند اکشن command دستوری است که سرور می فرستد که سرور اب استفاده از آن مقداری ورودی از برای

• Web :

مقدار ورودی (URL) را از داخل مرورگر گوشی باز می‌کند.

در صورت در دسترس نبودن لینک عبارت "لینک مورد نظر یافت نشد" نمایان می‌شود.

```
<page>
    <button text="open URL" onClick="web/http://example.com">
</page>
```

• Script:

برای اجرای یک تابع در script های نوشته شده ی شما استفاده می شود.

```
<page>
    <button text="call script" onClick="script/myscript:myFunction()">
    <script id="myscript">
    function myFunction() {
      console.log("Hello Appsan");
        }
    </script>
</page>
```

> توضیحات تکمیلی اسکریپت نویسی در بخش استفاده از جاوا اسکریپت در اپسان نوشته شده است.

• appstorage:

برای ذخیره داده در shared storage در اندرویدو در local storage در وب ذخیره می شود و قابل دسترسی با bind است

برای ذخیره ی مقدار یک کلید از پس از \ قرار داده و مقدار مورد نظر برای ذخیره از پس از : قرار دهید.

```
<page>
    <button text="save text" onClick="appstorage/myKey:my text to save">
</page>
```

- location:

> این اکشن از ورژن `appsan:0.1.74` در دسترس است.

اکشن `location` برای مدیریت و نمایش تنظیمات موقعیت مکانی کاربر مورد استفاده قرار می‌گیرد. این اکشن دو حالت مختلف دارد: نمایش دیالوگ تنظیمات موقعیت مکانی و باز کردن تنظیمات موقعیت مکانی در سیستم عامل. با استفاده از این اکشن ها به همراه bind های مربوط به location می توانید در صورت خاموش بودن موقعیت مکانی درخواست روشن کردن آن را بدهید.

1. `location/showdialog`

این اکشن دیالوگ تنظیمات موقعیت مکانی را به کاربر نمایش میدهد تا وی بتواند تنظیمات موقعیت مکانی دستگاه را فعال کند. این اکشن به دقت بالایی برای موقعیت‌یابی نیاز دارد و از APIهای گوگل برای نمایش دیالوگ استفاده می‌کند.

مثال استفاده:

```
<button text="Show Location Dialog" onclick="location/showdialog"/>
```

این دکمه دیالوگ تنظیمات موقعیت مکانی را به کاربر نمایش میدهد.

2. `location/openSettings`

این اکشن صفحه تنظیمات موقعیت مکانی را مستقیماً در سیستم عامل باز میکند، جایی که کاربر میتواند موقعیت مکانی دستگاه را فعال یا غیرفعال کند.

مثال استفاده:

```
<button text="Open Location Settings" onclick="location/openSettings"/>
```

این دکمه صفحه تنظیمات موقعیت مکانی دستگاه را باز میکند.

Bind

با استفاده از bind می توانید یک variable را به مقادیر مختلف وصل کنید مثلا با گذاشتن "bind="os/version ورژن اندروید کاربر را دریافت کنید.

بایند های تعریف شده:

os:

- os/screenwidth
- os/screenheight
- os/screenorientation
- os/layoutdirection
- os/version
- os/type

Appsan:

- appsan/version
- appsan/client

Element:

این bind اتریبیوت های المان را به شما تحویل می دهد ، به عنوان مثال برای گرفتن متن text از المان تکستی با آی دی txt1 این variable را داخل کد قرار دهید.:

```
<var id="textValue" bind="element/txt1:text"></var>
```

به طور کلی ساختار bind برای المان ها به صورت زیر است:

```
element/id:attr
```

که id ، آی دی المان مورد نظر و attr ، اتریبیوت آن المان است.

User:

این bind امکان گرفتن یک id منحصر به فرد از کاربر وارد شده را ایجاد می کند که می تواند برای شناسایی کاربر وارد شده استفاده شود.

مثال:

```
<var id="identity" bind="user/identifier"></var>
```

appstorage:

این bind امکان گرفتن مقدار ذخیره شده با appstorage را فراهم میکند. برای گرفتن مقدار کلید مقدار را بعد از / قرار دهید.

```
<var id="identity" bind="appstorage/myKey"></var>
```

component:

امکان دریافت مقادیر تعریف شده در component را فراهم می کند

```
<var id="myValue" bind="component/myKey"></var>
```

Deeplink:

با `deeplink/key` ت صورت زیر وجود دارد که به داده های دیپ لینک باندینگ برای دسترسی کلید های زیر در دسترس است.

Available:

بررسی می کند که آیا کاربر از طریق یک لینک عمیق به جلسه فعلی رسیده است یا خیر.
برمی گرداند: "true" (رشته) اگر داده های لینک عمیق وجود داشته باشد، در غیر این صورت "false" (رشته).

Path:

بخش مسیر اصلی URL لینک عمیق را استخراج می کند.
برمی گرداند: رشته مسیر (مثال "/products/1234") یا `null` در صورت نیافتن داده های لینک عمیق.

Query:

پارامترهای کوئری درون لینک عمیق را جدا می کند.
برمی گرداند: رشته ای که نشان دهنده پارامترهای کوئری است. (مثال "?color=blue&size=large")
یا `null` در صورت نیافتن داده های لینک عمیق.

queryjson:

یک رشته json از پارامتر ها ی داخل deeplink را بر می‌گرداند.

Host:

بخش نام میزبان (hostname) در URL لینک عمیق را بازیابی می کند.
برمی گرداند: رشته نام میزبان (مثال "www.example.com") یا `null`
در صورت نیافتن داده های لینک عمیق.

Port:

شماره پورت مشخص شده در URL لینک عمیق (در صورت وجود) را استخراج می کند.
برمی گرداند: رشته شماره پورت (مثال "8080") یا `null` در صورت نیافتن داده های لینک عمیق.

Fragment:

شناسه قطعه (fragment identifier) در URL لینک عمیق را که اغلب برای ناوبری درون صفحه ای استفاده می شود، بازیابی می کند.
برمی گرداند: رشته قطعه (مثال "#section3") یا `null` در صورت نیافتن داده های لینک عمیق.

Pathsegments:

بخش های جداگانه مسیر لینک عمیق را به صورت یک آرایه ارائه می دهد.
برمی گرداند: رشته ای حاوی بخش های مسیر که با کاما جدا شده اند (مثال "products,1234") یا `null` در صورت نیافتن داده های لینک عمیق.

Queryparameters:

بخش پارامترهای کوئری در لینک عمیق را به یک نمایش رشته ای کاربرپسند تبدیل می کند.
برمی گرداند: رشته ای با قالبی که پارامترهای کوئری را نشان میدهد (مثال "{color: blue, size: large}") یا `null` در صورت نیافتن داده های لینک عمیق.

location:

> این باندن از ورژن `appsan:0.1.74` در دسترس است.

location/on:

Bind `location/on` برای بررسی فعال یا غیرفعال بودن موقعیت مکانی (GPS) بر روی دستگاه کاربر استفاده میشود. این Bind به توسعه‌دهندگان امکان میدهد تا به سادگی وضعیت GPS دستگاه را بررسی کرده و نتیجه آن را دریافت کنند.

مقدار بازگشتی:

- اگر GPS فعال باشد، مقدار `"true"` بازگردانده میشود.
- اگر GPS غیرفعال باشد، مقدار `"false"` بازگردانده میشود.

آپدیت کردن صفحات بدون ایجاد صفحه جدید

اگر در خواصی که از اکشن پشتیبانی می کنند از اکشن update استفاده کنید، پیامی با اطلاعات داخل صفحه (به فصل "مقادیر ارسالی برای سرور" مراجعه کنید) برای میکروسرویس شما ارسال میشود. داخل این پیام کلید update وجود دارد که با استفاده از مقدار ورودی آن که در کد xml خود پس از “ / ” وارد کردید میتوانید تغییرات مورد نیاز را مشخص کنید.

علاوه بر مقدار ورودی کلید update دیتای صفحه ای که کاربر مشاهده می کند نیز برای شما ارسال می شود که میتوانید از آن برای شناسایی تغییرات مورد نیاز استفاده کنید. (به فصل "مقادیر ارسالی برای سرور" مراجعه کنید)

برای مشخص کردن تغییرات شما باید کد JSON به شکل زیر بسازید:

```
{
    widgetdata: {
        [id]:{
            [attr]:[value]
            color:red,
            text:"hello"
            override
        }
    },

    variabledata:{
        var1:3
    }

}
```

> همان طور که در مثال مشخص شده است، تغییرات اعمالی به دو دسته تقسیم می شوند :
>
> 1. widgetdata : برای مشخص کردن تغییرات المان ها استفاده می شود.

2. variabledata : برای تغییری در variable ها مورد استفاده قرار میگیرد.

- widgetdata :

تغییرات المان ها داخل این کلید قرار میگیرند.

برای مشخص کردن المانی که تغییرات روی آن اعمال میشود id آن المان را نوشته و آن را برابر با یک جیسون دیگر قرار دهید و داخل آن از نام خاصیت به عنوان کلید و مقدار جدید به عنوان مقدار کلید استفاده کنید.

برای مثال همان طور که در کد بالا دیده میشود با گذاشتن text به عنوان کلید و متن "hello" به عنوان مقدار متن نمایش داده شده روی دستگاه کاربر به "hello" تغییری میکند و این اتفاق بدون باز شدن صفحه جدید رخ می دهد.

- variabledata :

تغییرات variable ها در این کلید قرار می گیرد. ساختار جیسونی که داخل این کلید قرار میگیرد به صورت نام متغیر به عنوان کلید و مقدار جدید آن به عنوان مقدار خواهد بود.

مقادیر ارسالی برای سرور

با هر بار صدا شدن میکروسرویس شما مقادیری برای شما داخل یک JSON ارسال میشود این مقادیر به صورت زیر است:

- Inputs :

مقادیر المان هایی که برای آن ها id تعریف کرده اید در این کلید جیسون برای شما ارسال می شود. این مقادیر برای هر المان متفاوت است به عنوان مثال مقدار المان text متن داخل آن و مقدار المان textInput متن ورودی کاربر میباشد.

تمام این مقادیر به صورت یک json برای شما ارسال شده که در یک طرف id المان به عنوان کلید و مقدار المان به عنوان مقدار جیسون داده میشود.

برای مثال با ارسال شدن کد xml زیر:

```
<page>
    <text id="text 1" text="enter your phone number"/>
```

```
    <texstInput id="phone" hint="phone number"/>
    <button text="goto page 2" onClick="command/page2"/>
</page>
```

مقدار جیسون بازگردانده شده پس از انتخاب دکمه goto page 2 برابر با مقدار زیر خواهد بود:

```
{
    "text1": "enter your phone number",
    "phone": "01234566789"
}
```

که “0123456789” مقدار وارد شده توسط کاربر است.

تمامی المان هایی که id داشته باشند داخل این JSON نمایش داده می شوند. همین چنین متغیرهایی (variable) که تعریف می شوند نیز به همین صورت نمایش داده می شوند.

- Command :

در صورتی که از اکشن command استفاده کنید پس از اجرا شدن اکشن توسط کاربر این کلید برای شما ارسال میشود که مقدار ورودی آن داخل کلید میباشد شما میتوانید با استفاده از این مقدار صفحه ای که باید به کاربر نشان دهید را مشخص کنید.

- Update :

در صورتی که از اکشن update استفاده کنید پس از اجرا شدن اکشن توسط کاربر این کلید برای شما ارسال میشود که مقدار ورودی آن داخل کلید میباشد شما میتوانید با استفاده از ورودی تغییرات مورد نیاز را شناسایی و بدون ساخت صفحه جدید مقادیر صفحه را تغییر دهید.

- sdp_userId :

id کاربر را نمایش می دهد. این آیدی منحصر به فرد است و شما می توانید از آن به عنوان اعتبار سنجی کاربر استفاده کنید.