

SN经验浅谈

SN开发经验谈

零、快速参考

快捷键

帮助 Ctrl + H

搜索 Ctrl + F

下一个 Ctrl + G

上一个 Ctrl + Shift + G

注释 Ctrl + /

脚本协助 Ctrl + Space

通用代码片段例子

在表中创建一条新记录

```
1 var incidentGr = new GlideRecord('incident');
2 incidentGr.newRecord();
3 incidentGr.short_description = 'Hello!';
4 incidentGr.insert();
```

通过ID获取记录

```
1 var incidentGr = new GlideRecord('incident');
2 if (incidentGr.get('incidentSysID')) {
3     // Do something
4     gs.info(incidentGr.number + ': ' + incidentGr.short_description);
5 }
```

使用 GlideAggregate 统计活动记录

```
1 var incidentGa = new GlideAggregate('incident');
2 incidentGa.addQuery('active', 'true');
```

```

3 incidentGa.addAggregate('COUNT');
4 incidentGa.query();
5
6 var count = 0;
7 if (incidentGa.next()) {
8     count = incidentGa.getAggregate('COUNT');
9 }

```

将 JS 对象转换为 JSON

```

1 var jsonString = JSON.stringify(obj);

```

将 JSON 字符串转换为 JS 对象

```

1 var obj = JSON.parse(jsonString);

```

将 OR 条件添加到 GlideRecord 查询

```

1 var incidentGr = new GlideRecord('incident');
2 var orQuery = incidentGr.addQuery('state', '1'); // A or B or C
3 orQuery.addOrCondition('state', '2');
4 orQuery.addOrCondition('state', '3');
5 incidentGr.query();
6
7 while (incidentGr.next()) {
8     // Do something
9 }

```

常用的 GlideSystem 方法

```

1 var user      = gs.getUser(); // 当前用户对象
2 var userID    = gs.getUserID(); // 当前用户ID
3 var userName  = gs.getUserName(); // 当前用户的Name
4 var groups    = gs.getUser().getMyGroups(); // 当前用户所属组
5 var isMember  = gs.getUser().isMemberOf(current.getValue('assignment_group')); //
6 var isMember  = gs.getUser().isMemberOf('Service Desk'); // 判断是否为组成员（通过组
7 var hasRole   = gs.hasRole('itil'); // 判断当前用户是否有'itil'角色

```

常用的 AngularJS 指令

`ng-model`

`ng-repeat`

`ng-show/ng-hide`

`ng-if`

`ng-class`

`ng-change`

`ng-include`

服务门户页面客户端常用Api

`spUtil`

`recordWatch`

`$sp`

一、基本开发经验

在 ServiceNow 中实现客户的需求从编写高质量的代码开始。严格按照最佳实践做法来开发有助于确保良好的性能、减少潜在的Bug并简化调试过程。

0.写在前面——基本心态

- 当你的程序无法正常工作时，应进行调查和调试，之后如果仍然需要调试帮助，请联系其他开发人员寻求帮助
 - 利用 SN 的调试工具，例如User Criteria Diagnostics，Debug Security Rules和 JavaScript debugger
 - 使用 SN 社区或 ServicePortal.io 等网站进行调查
- 小心不要直接将在网上找到的代码复制粘贴到程序中。先通读一遍程序，明白你复制了什么
- 最好先在个人环境中尝试新的想法或概念，然后再在开发环境中实施
- 参考[开发者文档](#)，常读常新
- 不要一口气写几百行代码。相反，一次编写几行代码，并进行测试以确保它按预期工作
- 在网上提问时参考[如何提出好问题](#)
- 为你做的事编写易读的文档

1.易读的代码

开发时，请记住其他人未来也会使用你开发的代码。因此要确保你的代码易于阅读和理解。详细的代码规范请参考[阿里巴巴Java开发手册v1.4.0（详尽版）](#)。此处仅说明最基本的要点：

1. 精确简洁的注释
2. 令人赏心悦目的缩进
3. 避免炫技写法

2.小模块方法

如果你发现客户的需求有重复出现相同或相似的逻辑，那么你可以考虑在Script Include中写可复用的、小而美的方法。笔者认为，Script Include本质上是可以被SN其他的服务端脚本调用的方法库（Client Callable启用时客户端脚本也可以调用Script Include中的方法）。

举例来说，一个公司有多个部门，每个部门有部门负责人，客户需要获取某个特定部门的部门负责人，那么这个时候你可以写这样的方法：

```
1  /*
2  **
3  ** getDepDutyPersonExactly - 获取某个特定部门的部门负责人
4  ** @param department - 特定部门的sys_id
5  ** @param duty - 部门负责人职位名称
6  ** @return List - 部门负责人列表
7  **
8  */
9  getDepDutyPersonExactly: function(department, duty) {
10     var userList = [];
11     var dutyGr = new GlideRecord("u_user_department_duty");
12     dutyGr.addQuery('u_department', department);
13     dutyGr.addQuery('u_duty', duty);
14     dutyGr.addQuery('u_active', true);
15     // 不要忘记query(), 它发出查询并获取结果。
16     dutyGr.query();
17     while(dutyGr.next()) {
18         userList.push(dutyGr.getValue('u_user'));
19     }
20     return userList;
21 }
```

这个getDepDutyPersonExactly()方法可以在服务端脚本中这样被调用。

```
1 new scope_name.nameofUtils().getDepDutyPersonExactly(department, duty);
```

小而简洁，功能有限。

易于理解逻辑、输入和输出。这使得未来使用这个方法的人更容易修改。

易测试。顺便一提，您测试代码时，请务必测试有效和无效输入，以确保健壮性。

这样做也不是完全没有缺陷的，例如，当你接手他人的项目或者是开发周期紧迫时，你可能不知道有前人已经写过某种方法，于是你又在Script Include中再次写了几乎一样的方法。因此，确保每个方法都有清晰的注释可以最大程度避免这种情况发生。

同时也要避免在一段代码中重复调用同一个方法，重复调用可能会导致性能问题。举反例来说：

```
1 if (gs.getUserID() == current.u_a ||
2     gs.getUserID() == current.u_b ||
3     gs.getUserID() == current.u_c ||
4     gs.getUserID() == current.u_d) {
5
6     //要实现的需求
7
8 }
```

这样写不仅重复调用了特定方法，而且看起来令人有些难受。写成这样会更好：

```
1 //gs.getUserID() - 获得当前登录用户的sys_id
2 var currentUser = gs.getUserID();
3 var isA = (currentUser == current.u_a);
4 var isB = (currentUser == current.u_b);
5 var isC = (currentUser == current.u_c);
6 var isD = (currentUser == current.u_d);
7
8 if (isA || isB || isC || isD) {
9
10     // 要实现的需求
11
12 }
```

3.验证值是否存在

请在使用变量和字段之前验证它们是否具有值。

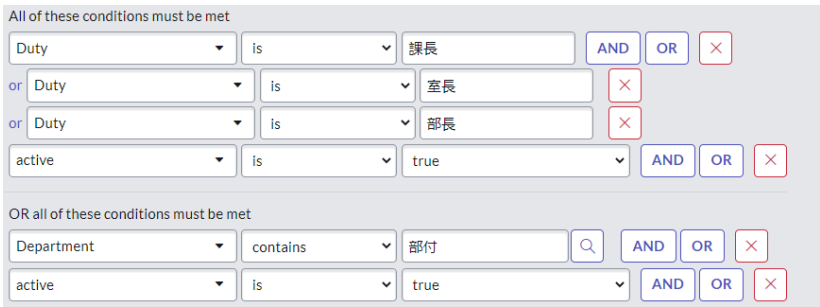
```
1 if (current.u_department)
2     var dep = current.u_department;
3 else
4     gs.info("current.u_department is undefined");
```

4.避免复杂的GlideRecord查询

有时与其创建一系列 addQuery() 和 addOrCondition() 调用来获取结果，不如使用 addEncodedQuery() 使查询更易于创建和维护。

对于复杂的 GlideRecord 查询，通过过滤器生成编码查询字符串并将该字符串与 addEncodedQuery 一起使用，可以更轻松地创建查询字符串。

随着客户需求变化，可以用列表过滤器创建新的查询字符串，并验证结果，再维护现有脚本。



举一个不太好的例子，如上图我们需要构建如此复杂奇怪的查询逻辑，此时用addQuery() 和 addOrCondition() 会稍微有些麻烦，但是可以如下图所示获取查询字符串并直接用 addEncodedQuery

All > Duty = 課長.or. Duty = 室長.or. Duty = 部長 > active = true > or Department Name contains 部付 > active = true

☐

Open new window

Copy URL

Copy query

```
1 var depGr = new GlideRecord('u_user_department_duty');
2 depGr.addEncodedQuery('u_duty=課長^ORu_duty=室長^ORu_duty=部長^u_active=true^NQu_');
3 depGr.query();
4
5 // 要实现的需求
6
```

此外，对于大型的数据表(百万级别)，要避免使用复杂的查询。原因是查询时间很长，并且会影响实例整体性能。

如果不得不需要查询大型数据表，目前笔者能想到的最好办法是

- 1.新建计划作业，让查询工作在非工作时间(比如深夜)进行。
- 2.根据实际需求使用setLimit(), 返回较少的记录比返回所有记录要快的多。

5.避免掉入编程陷阱

- 1. 验证代码是否有效时可以在个人实例试验或Scripts - Background中测试。
- 2. 不要试图一次写出数百行代码。一次编写几行代码并边做边测试以确保它按预期工作。
- 3. 不要点遍历到参考字段的 sys_id

```
1 var id = current.user_id.sys_id; // 错误的示范
2 var id = current.getValue('user_id'); // 正确的示范
```

6. 硬编码前深思

少即是多，在你进行hard-coding的时候，请一直考虑这个问题：

环境变更后，硬编码的部分是否还能正常工作？

二、Business Rule经验

Business Rule(BR)是在显示、插入、更新或删除记录或查询表时运行的服务器端脚本。

1. 什么时候运行

Display:

用于提供客户端脚本对服务器端数据的访问。

Before:

用于更新当前对象的信息。

After:

用于更新需要立即显示的相关对象的信息，如 GlideRecord 查询，after使用频率较高。

Async:

用于更新不需要立即显示的相关对象的信息。

此外，BR的高级视图中有一个Condition字段，可以的话请详细写好运行的条件以减少不必要的运行。

2. 一些建议

- 如果可能，使用BR的条件
- 尽可能不要使用 `current.update()`
- 不要编写可能在运行时再次激活自己的BR
- 不要创建全局BR，最好使用 Script Includes 代替

- 了解需求，是否打算因运行此BR而激活其他BR。如果不是，请确保将 `.setWorkflow` 设置为 `false`
- 可以用BR来验证Client Script/UI Policy之上的输入，因为字段可以在字段外编辑

三、客户端脚本开发经验

客户端脚本是在客户端而非服务端上运行的 JavaScript 代码。精心设计的客户端脚本可以减少完成表单所需的时间并改善用户体验。但是，不太好的客户端脚本会显著减慢表单加载时间。除了 onCellEdit 客户端脚本之外，客户端脚本仅适用于表单。

1.只运行必要的客户端脚本

因为客户端脚本没有条件字段，所以每次加载特定表单时，onLoad() 和 onChange() 脚本都会完整运行。为避免不必要地运行耗时的脚本，请确保客户端脚本仅执行必要的任务。

1.onChange()脚本尽可能使用isLoading()

isLoading()是防止不必要的代码占用浏览器时间的最简单方法。它用在任何不需要在加载表单时运行的脚本的开头。无需在表单加载时运行此脚本，因为逻辑在上次更改字段时已经运行。

如果 onChange() 脚本必须在表单加载期间运行，请这样写客户端脚本：

```
1 function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2
3     if (isLoading) {}; // 保留isLoading()
4
5     // onChange逻辑实现部分
6
7 }
```

2.考虑添加newValue != oldValue检查

要让脚本对表单加载后更改的值做出反应，请考虑使用 newValue != oldValue 检查（非强制要求的做法）。可以参考下面的例子：

```
1 function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2
3     if (isLoading)
4         return;
5
6     if (newValue != oldValue) {
7         // 实现功能
8     }
9 }
10
```


2.最小化对服务端的查询

客户端脚本使用客户端可用的数据或从服务器检索的数据。 尽可能直接使用客户端数据，以减少耗时的服务器查找需求。 从服务器获取信息的主要方法是 `g_scratchpad` 和异步 `GlideAjax` 查找。

这两个方法之间的主要区别在于 `g_scratchpad` 在加载表单时发送一次（信息从服务器发送到客户端），而 `GlideAjax` 是在客户端从服务端请求信息时动态触发的。

~~其他方法，`GlideRecord` 和 `g_form.getReference()` 回调也可用于检索服务器信息。但是，由于它们对性能的影响，不再推荐使用这些方法。当大多数情况下只需要一个字段时，这两种方法都会检索所请求的 `GlideRecord` 中的所有字段。`GlideRecord` API 不适用于作用域应用程序。~~

① `g_scratchpad` 例

`g_scratchpad` 对象将信息从服务端传递到客户端。

举例来说明，如果你有一个需要访问字段 `u_department` 的客户端脚本，但是这个字段不在表单上，那么。 有一种典型解决方案是将字段放入表单，然后始终使用客户端脚本或 UI Policy 将此字段隐藏。 虽然这种解决方案配置起来非常方便，但实际可能带来执行时间的增加。

如果你知道客户端在加载表单之前需要从服务端获取哪些信息，则 `Display` 的业务规则可以创建 `g_scratchpad` 属性来保存此信息。 `g_scratchpad` 对象在请求表单时发送到客户端，使其可用于所有客户端脚本方法。但是只能在加载表单时以这种方式加载数据。 无法动态触发。

② 异步 `GlideAjax` 例

请直接参考例子代码来写。

客户端脚本：

```
1 function onChange(control, oldValue, newValue, isLoading) {
2     if (isLoading || newValue == '') {
3         return;
4     }
5
6     var ga = new GlideAjax('asu_GetLocationData');
7     ga.addParam('sysparm_name', 'getCampus');
8     ga.addParam('sysparm_buildingid', g_form.getValue("u_building"));
9     ga.getXMLAnswer(updateCampus);
10 }
11
12 function updateCampus(answer) {
13     var clearvalue;
14     if (answer) {
15         var returneddata = JSON.parse(answer);
16         g_form.setValue("campus", returneddata.sys_id, returneddata.name);
```

```

17     } else {
18         g_form.setValue("campus", clearvalue);
19     }
20 }

```

服务端脚本：

```

1 var asu_GetLocationData = Class.create();
2 asu_GetLocationData.prototype = Object.extendObject(AbstractAjaxProcessor, {
3     getCampus: function () {
4         var buildingid = this.getParameter('sysparm_buildingid');
5         var loc = new GlideRecord('cmn_location');
6         if (loc.get(buildingid)) {
7             var campus = new GlideRecord('cmn_location');
8             if (campus.get(loc.parent)){
9                 var results = {
10                     "sys_id": campus.getValue("sys_id"),
11                     "name": campus.getValue("name")
12                 };
13                 return JSON.stringify(results);
14             }
15         } else {
16             return null;
17         }
18     }
19 });

```

3.设置参考字段的值时的注意点

在参考字段上使用 setValue() 时，请尽可能将显示值(display value)包含在值 (sys_id) 中。如果你设置的值没有显示值，SN会执行同步 Ajax 调用来检索你指定的记录的显示值。这种额外的服务器往返可能有出现性能问题的风险。举个例子说明：

```

1 var id = 'iku1145148101919386iiyokoiiyo';
2 var name = '開発部';
3
4 g_form.setValue('dep', id, name); // 这种情况下没有执行同步Ajax调用

```

4.通用的写客户端脚本时的注意点

1.避免写出全局客户端脚本

全局(Global)客户端脚本没有表限制； 因此它们将加载到系统中的每个页面上，这会使得页面打开速度变慢

2.不要去做DOM操作

避免文档对象模型 (DOM) 操作。 当浏览器更新时，它可能会导致可维护性问题。

5.验证用户输入

在曾经验证客户输入时，基本依赖于onChage()脚本，如例：

```
1 function onChange(control, oldValue, newValue, isLoading) {
2     if (isLoading || newValue == '') {
3         return;
4     }
5
6     var string = g_form.getValue('variableName');
7     var regex = /D/g;
8     string = string.replace(regex, '');
9     g_form.hideFieldMsg( variableName, 'true');
10    g_form.showFieldMsg( variableName, 'Not number: '+newValue, 'error', false);
11    g_form.setValue('variableName', string);
12
13 }
```

但是在Madrid版本发布后，引入了一个不错的新功能：“Variable Validation Regex”。 简而言之，你可以通过从参考字段中选择记录来添加对（单行文本）变量的验证。 显示正则表达式选项的参考字段。 通过这种方式你可以轻松添加验证、重复使用验证，不必再一遍又一遍地编写Catalog Client Script，举个例子如下图。

The screenshot shows the 'Variable' configuration page for '支払金額 (円)'. The 'Type' is 'Single Line Text'. The 'Validation Regex' dropdown is open, showing a list of options: 'Currency', 'None --', 'Currency', 'Email', 'IP Address', 'Number', 'URL', and 'US Zip Code'. The 'Currency' option is selected. The 'Variable Width' is set to 'System Default Width (50) %'. The 'Variable attributes' section is also visible.

*支払金額（円）を入力してください

壹壹肆伍壹肆

有効な金額を入力してください

▲ 次のフィールドにエラーが含まれています: 支払金額（円）を入力してください

*支払日を入力してください
2023-04-26

*支払金額（円）を入力してください
壹壹肆伍壹肆
有効な金額を入力してください

*予算項目のうち、いずれかを選択して下さい

一時保存 送信

将验证正则表达式应用于变量，验证就会在字段更改时激活。如果该字段填写不正确，则会在变量下方显示一条错误消息。且不必添加额外的来防止表单提交格式不正确的变量。如果点击提交按钮，屏幕顶部会出现一条错误消息。

不幸的是，这项功能并不完美。有一个讨厌的bug，当对强制（Mandatory）变量应用正则表达式变量验证时，用户在尝试提交目录项时会卡住。对于非强制变量，则不存在bug。

ServiceNow 将此列为PRB1355430。

到目前版本(Utah犹他)依旧没有针对性的修复。

请参考此[解决方案](#)。

除此之外也可以新建一个onSubmit()的Catalog Client Script，指定 `g_form.submit()` 方法的参数：

```
1 g_form.submit('submit');
```

- If possible, use conditions in Business Rules

四、XML导入导出经验

要将 **数据** 从一个实例迁移到另一个实例，一般可以使用XML 文件。

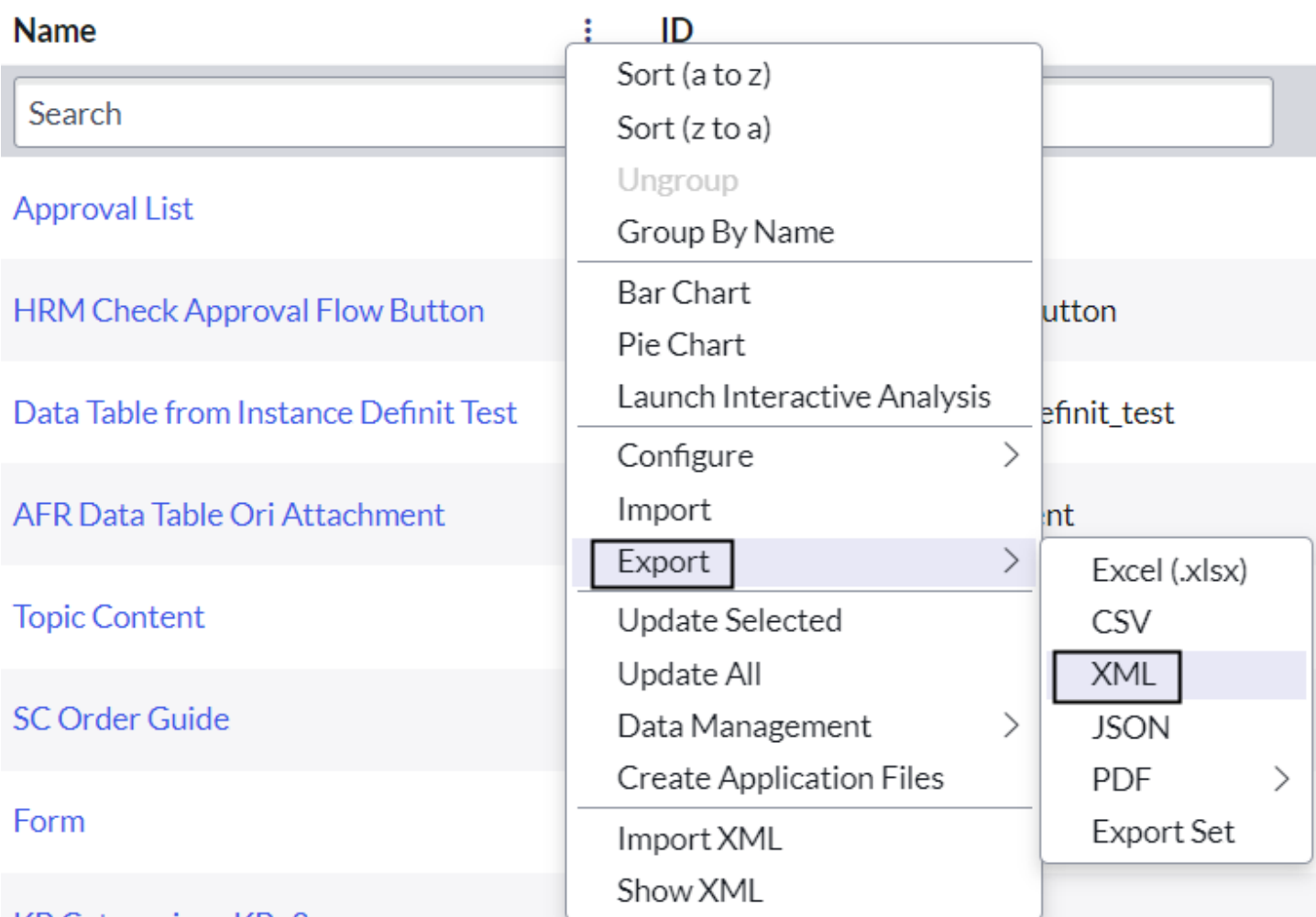
将单个记录（例如事件、用户、配置项 (Configuration Item) 或计划的作业）从一个实例导出到另一个实例，一般可以用Export XML和Import XML。例如，如果用户在正式中遇到问题，而你希望在不影响用户的情况下对开发环境进行深入诊断，你可以将特定的记录导出为 XML 文件，并将其导入到开发环境中。

实际上这是一个很简单的操作，但是它很实用。

将记录导出为 XML：

1.登录到包含源数据的实例。

- 2.导航到要导出的表。
- 3.根据你的实际需要，过滤此表
- 4.右键单击表的标题并选择Export > XML（需要admin角色）。
- 5.根据浏览器的不同，可能会出现一个对话框可能会提示你保存文件，或者浏览器可能会自动将 XML 文件保存到浏览器首选项中指定的下载文件夹中。



将XML导入进实例：

- 1.登录到要导入数据的实例。
 - 2.导航到SN中的任何列表。这是一个小技巧，可以使用任何列表，因为 XML 文件包含记录的目标表名称。
 - 3.右键单击列表标题并选择Import XML
 - 4.在导入屏幕中，单击选择文件并选择之前导出的 XML 文件并单击上传。
- 注意：执行XML导入时，请务必记住，XML导入只会导出当前记录，而不是与其相关的记录。例如，在导出用户记录（sys_user）时，它所属的Group和Role不会成为 XML 文件的一部分。在这种情况下，可能有必要将描述这些关系的记录导出到单独的 XML 文件中。按着这个例子来看，用户角色表（sys_user_has_role）和组成员（sys_user_grmember）表也需要导出。

五、更新集经验

更新集在SN开发、迁移中是十分重要的部分，你在创建更新集时需要详细描述这个更新集做了什么工作。

简单说明更新集相关的注意点。

1.顺序！

更新集迁移时，最好有一个排序表，详细说明更新集迁移时的顺序。

举个例子，如果更新集Rococo_HRC_P1_S1创建了一个新表，而更新集Rococo_HRC_P1_S2更改了该表的Form Layout，则首先迁移更新集Rococo_HRC_P1_S2会导致错误。它无法将新的Form Layout应用于表，因为表尚未创建。

2.额外的迁移

不是所有更改都会被更新集捕获的。因此你在完成开发工作后需要检查更新集中的内容，未被包含在内的更新请使用XML文件迁移。XML相关内容请看第四部分。

六、小部件经验

笔者自己对于小部件的了解也不是很深入，因此参考这部分时请注意可能存在错误。附：官方对于自定义小部件的[教程文档](#)(Utah)。

0.什么是小部件？

小部件是可重用的组件，它构成门户页面的功能并允许用户与服务门户交互并显示信息。小部件有六个部分：HTML、CSS、Client Script、Server Script、Link Function、Options。不过目前笔者接触较多的是HTML、Client Script、Server Script以及CSS。

HTML

HTML 接受和显示数据，并为小部件提供基本结构。这将是用户在最终产品和浏览器中看到的内容。

CSS

CSS 或层叠样式表允许对 HTML 进行样式设置并自定义页面上的内容。这包括边框、字体、颜色、可见性、大小、定位等等。

Client Script

小部件的客户端脚本定义了 AngularJS 控制器。它定义了小部件的行为并将data model传递给HTML。

Server Script

服务器脚本处理 ServiceNow 实例中的任何交互。您可以引用脚本包含、从表中获取数据、执行更新等。在本教程中，我们将从 ServiceNow 中的表中检索数据。

1.在HTML中使用Bootstrap

Bootstrap，是一款来自Twitter的前端CSS框架，它基于HTML、CSS和JavaScript，是目前最受欢迎的前端框架之一。它简洁灵活、规范优雅，可快速实现Web开发，它是一个随service portal一起开箱即用的组件库。允许你快速设计响应式小部件。Bootstrap有大量的内置类，请参考此[文档](#)。

以下是一些用法举例，推荐你将其复制到实例的小部件中，查看效果，并且微调代码，查看变化。

```
1 <div class="jumbotron">
2     <h1>基本用法举例</h1>
3     <p>Jumbotron（超大屏幕）会创建一个大的灰色背景框，里面可以设置一些特殊的内容和信息</p>
4     <p>请注意它如何根据窗口的大小做出反应</p>
5 </div>
```

```
1 <div class="container-fluid bg-primary">
2     <p>这是一个将根据窗口分辨率调整大小的容器</p>
3 </div>
```

```
1 <div class="container-fluid bg-primary">
2     <div class="row">
3         <div class="col-sm-4 text-center">
4             <h2>肯德基</h2>
5             <p>吮指原味鸡很好吃</p>
6         </div>
7         <div class="col-sm-4 text-center">
8             <h2>金拱门</h2>
9             <p>双层鳕鱼堡很好吃</p>
10        </div>
11        <div class="col-sm-4 text-center">
12            <h2>汉堡王</h2>
13            <p>三层皇堡很好吃</p>
14        </div>
15    </div>
16 </div>
```

肯德基

吮指原味鸡很好吃

金拱门

双层鳕鱼堡很好吃

汉堡王

三层皇堡很好吃



```
1 <button type="button" class="btn">基础按钮</button>
2 <button type="button" class="btn btn-primary btn-lg">有色彩的按钮</button>
```

```
1 <div class="container">
2   <div class="dropdown">
3     <button
4       type="button"
5       class="btn btn-primary dropdown-toggle"
6       data-toggle="dropdown"
7     >
8       下拉框按钮
9     </button>
10    <div class="dropdown-menu">
11      <li><a class="dropdown-item" href="esc" target="_blank">员工服务中心</a></li>
12      <li><a class="dropdown-item" href="sp" target="_blank">服务门户</a></li>
13      <li><a class="dropdown-item" href="kb" target="_blank">知识库</a></li>
14    </div>
15  </div>
16 </div>
```

2.AngularJS

AngularJS 是一个 JavaScript 框架。它是一个以 JavaScript 编写的库。AngularJS 是以一个 JavaScript 文件形式发布的，可通过 script 标签添加到网页中。AngularJS 通过指令扩展了 HTML，且通过表达式绑定数据到HTML。在这里记录了一些常用的AngularJS指令。

1.ng-if

它判断一个表达式，如果为真，则显示 DOM 元素。举个例子，仅当表达式isTrue为True时才会显示 div。

```
1 <div ng-if="isTrue">要显示的内容：{{data.name}}</div>
```


2.ng-repeat

ng-repeat 用于在存在项目列表时遍历循环。举个简单例子：

```
1 <ul>
2   <li ng-repeat="person in people | orderBy: 'age'">
3     {{person.name}}
4   </li>
5 </ul>
```

```
1 api.controller=function($scope) {
2   /* widget controller */
3   var c = this;
4
5   $scope.people = [{
6     name: 'A',
7     age: 11
8   },{
9     name: 'B',
10    age: 4
11  },{
12    name: 'C',
13    age: 51
14  },{
15    name: 'D',
16    age: 48
17  }]
18 };
```

这将为指定数组中的每个项目输出 标签。您可以访问对象 people 并将每个项目提取到 person 中。作为一个小插件，你还可以添加一个过滤器来对列表进行排序。在这种情况下，按年龄递增排序。

3.ng-click

ng-click用来指定一个元素被点击时调用的方法或表达式，该指令类似于onclick，但它更强大。可以将它连接到客户端脚本以触发功能。举个例子：

```
1 <button ng-click="onClick('hello')">请点击按钮</button>
```

```
1 api.controller=function($scope) {  
2   $scope.onClick = function(name) {  
3     console.log(name);  
4   }  
5 }
```

4.ng-class

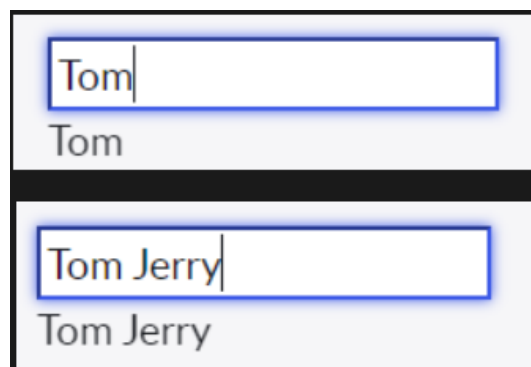
该指令允许你通过绑定表达式在元素上动态设置 CSS 类。该指令的用法请参考此[文档](#)。

5.ng-model

该指令允许你将文本元素双向绑定到变量。见例子，效果如图：

```
1 <div>  
2   <input ng-model="name" type="text" id="name"/>  
3 </div>  
4 <div>  
5   {{name}}  
6 </div>
```

```
1 api.controller=function($scope) {  
2   $scope.name = "TadoKoro";  
3 };
```



ng-model绑定到在客户端脚本中找到的name。

div中的文本将始终与输入框内的内容相同。

6.ng-bind

用于将客户端脚本中的变量附加到 html 页面。

3.从客户端到服务端的通信

如何在客户端和服务端之间进行通信？
提交操作时需要通知客户端，并且应该将该数据发送到服务端进行处理。



首先需要了解这两个对象：input和data。概括来说：
input是客户端用来与服务端通信的对象。
data 是服务端用来与客户端通信的对象。
小部件上的服务端脚本在加载时执行，可用于将数据对象(Data Object)从服务端传到客户端。
在客户端，可以调用方法（函数）来更新服务器。一般用的是：`server.get();` 和 `server.update();`。
在小部件客户端控制器上使用时这么写 `this.server.update();` 或 `var c = this; c.server.update();`。
当然还有 `spUtil.update($scope);` 下文会具体说明。
你问有什么区别？`this.server.get();` 允许你传入一个对象，而 `this.server.update();` 获取ng-models绑定的对象。
它们本质上做同样的事情，因此请使用你认为适合需求的一个。

4.CSS最佳实践

1.可读性

一个好的CSS应该是易读的
你可以这样写你的 CSS，所有内容都在一行中：

```
1 .example {
2     background: red; padding: 2em; border: 1px solid black;
3 }
```

这很难读。最佳实践是将每种样式写在自己的行中，如下所示

```
1 .example {  
2     background: red;  
3     padding: 2em;  
4     border: 1px solid black;  
5 }
```

2.自上而下

你可以采用这样的自上而下的结构来组织你的CSS:

1. 通用元素优先(`body` , `a` , `p` , `h1` 等)
2. `.header`
3. `.nav-menu`
4. `.main-content`
5. `.footer`

3. “共享” CSS

为了避免重复，相同式样的选择器可以写在一起，比如这样：

```
1 h1,  
2 h2,  
3 h3 {  
4     font-family: Impact;  
5     color: #d1a3f5;  
6 }
```

4.使用适当的命名约定

CSS 允许你将元素的样式和设计与其内容分开。你只需更改 CSS 即可更改小部件的设计，而无需更改 HTML。

CSS 元素的名称基于它们是什么，而不是它们给人的印象。例如，最好使用这样的名称 `.article-title`，而不是 `.article-large-font`。

5.避免内联样式

内联样式是指在 HTML 中声明 CSS。例如，

```
1 <p style="background:#ccc; color:#000; border: solid black 1px;"></p>
```

如果不得不只使用内联样式，小部件将很快变得臃肿并且很难维护。这是因为内联样式必须应用于你希望它们出现的每个元素。

因此，最佳做法是将 HTML 和 CSS 代码分开。

6. 简写属性

简写属性是可以让你同时设置好几个 CSS 属性值的 CSS 属性。使用简写属性，Web 开发人员可以编写更简洁、更具可读性的样式表，节省时间和精力。例如

```
1 img {  
2     margin-top: 5px;  
3     margin-right: 10px;  
4     margin-bottom: 5px;  
5     margin-left: 10px;  
6 }
```

和这个CSS是相同的

```
1 img {  
2     margin: 5px 10px;  
3 }
```

7. 尽可能不要用 `!important`

当在一个样式声明中使用一个 `!important` 规则时，此声明将覆盖任何其他声明。使用 `!important` 是一个坏习惯，应该尽量避免，因为这破坏了样式表中的固有的级联规则，使调试找 bug 变得更加困难了。

一些经验法则：

- 一定要优先考虑使用样式规则的优先级来解决问题而不是 `!important`
- 只有在需要覆盖全站或外部 CSS 的特定页面中使用 `!important`
- 永远不要在你的插件中使用 `!important`
- 永远不要在全站范围的 CSS 代码中使用 `!important`
- 与其使用 `!important`，你可以：
 - a. 更好地利用 CSS 级联属性
 - b. 使用更具体的规则。在您选择的元素之前，增加一个或多个其他元素，使选择器变得更加具体，并获得更高的优先级。

什么情况下可以使用 **!important**（请根据实际情况来考虑）

七、Workflow基础（编写中）

dev16788.service-now.com/esc?id=sc_cat_item&sys_id=da073a1e4fff0200086eed18110c72b

YouTube 有道 有道翻译 有道 有道翻译 文本、文... 地图 翻译 forgrt SN Pro Tips yisu DeepL翻译: 全世... qita Urged Mails Send... Workflow Editor ~... personal environ... VMware Horizon Synolog

Big Data Analysis

Request big data analysis environment

Service Overview:

Engage with the Data Service team to instantiate and provide Big Data Analysis environment and analysis. We provide expertise in a wide range of technology platforms and services ranging from data to in-depth analysis. Platforms include Hadoop, Pentaho and Flume.

Services Include:

- Data Modeling, Algorithm development
- Develop map reduce code, transformations
- Data Integration Services
- Data Quality and Metadata management
- Reports/Visualizations, Analytics

*Who is the primary contact?

System Administrator

*What cost center will be billed?

IT

*What is the business purpose

☐ KTLO

☐ Special Projects

☒ Testing

☐ Other

*Please describe your needs

Data Modeling, Algorithm development

Quantity: 1

Delivery Time: 2 Days

Order Now

这是OOTB中的某一Catalog Item（目录项，一些开发环境中可能是Record Producer），当你/用户填写了表单必需的内容，点击 Order Now（或者是Submit等）之后，系统会创建一条（多条）新记录。你可以看到目录项有一字段名为“WorkFlow”，因此笔者在这里会演示新建一个Workflow，以使用户提交时，一些自动化工作会进行。

写在最后的杂谈

这一部分是还未系统性分类或者是不便于分类的经验。

1. Flow Designer V.S. Workflow

Flow Designer 是 ServiceNow 提供的较新的流程工具。优点是它的代码更少，使最终用户能够创建和维护流程，缺点是它能实现的功能也少。

Workflow 是 ServiceNow 中使用的传统 Flow 引擎。仍然有许多功能使用工作流。它更类似于 BPMN(Business Process Model and Notation)，并且与 Flow Designer 相比更需要一些技术知识。

ServiceNow 在过去几年中一直在朝着低代码的方向发展。这个方向不仅适用于 Flow Designer，也适用于 AppEngine Studio。

ServiceNow 一直强烈建议使用 Flow Designer 而不是新开发的 Workflow 。与每个新版本的 Workflow 相比，更多新功能被添加到 Flow Designer 中。

此外，Flow Designer 可以与 Workflow 集成。

这就像大多数人已经转向更新更强大的语言（如 Java 和 Python）时仍在使用 C 语言一样。用 C 语言编写应用程序仍然是可能的，C 语言使开发人员能够更好地控制内存使用和解决性能问题，但需要更多的时间来开发和维护。因此，大多数开发人员已经转向提供更高效率的语言，并且只在需要更好性能的部分使用 C 语言。

C 语言仍在使用，因此它没有消亡。同样，Workflow 不会消亡（因为有很多客户依旧在使用 Workflow），只是会用的少而已。

2、EC 经验

EC 即 Employee Center，员工中心。

1. 配置 Header



一般来说，你可以在 ESC 画面看到这样的 **Global Header**，你可以通过修改 **Instance with Menu-Employee Center Menu** 的 Additional options, JSON format 来启用或禁用特定的 Header。

如果你想增加新的 Header，你可以复制并修改 EC Theme 中的 Employee Center Header 小部件。

特别要注意的是 Tours 这一 Header，要禁用它，请打开 sys_properties 表，并检索 com.snc.guided_tours.sp.enable 这一记录，将它的 Value 改为 false 即可。

（顺便一提，Footer 也是基本一样的操作，一般客户会要求隐藏整个 Footer（存疑？），导航至 **All > Employee Center > Employee Center Footers**，将特定记录的 Active 改为 false 即可实现）