PRETIUS APEX CLIENT SIDE VALIDATION

Author:	Bartosz Ostrowski
Position	Oracle APEX Developer
E-mail	bostrowski@pretius.com
Reviewer	Przemysław Staniszewski
Company	Pretius.com

Table of Contents

1. Intro

Pretius APEX Client Side Validation plugin was created due to lack of core functionality that allows user to run APEX defined validations **on the fly (live)** in APEX 4.x and 5.x.

The plugin is dynamic action plugin that allows user to validate form data according to defined APEX validations without submitting the page. APEX item validations can be executed "on the fly" (eg. after leaving the textfield) and do not require pressing submit button and reloading the page. You can find more information about standard APEX validations in APEX Documentation (https://docs.oracle.com/cd/E59726_01/doc.50/e39147/comp_val_proc002.htm#HTMDB29820).

Unlike other validation plugins, **Pretius APEX Client Side Validation** plugin does not require creating additional validations in Java Script. It is because the plugin executes APEX validations associated with APEX items. All you need to do is to create new dynamic action, choose affected items (by pointing APEX items or using jQuery selector) and create standard APEX validations. No JavaScript or jQuery knowledge required!

With the plugin you can choose three predefined styles for presenting validation result. Furthermore the plugin supports callback function (in JavaScript) on validation outcome. If you are brave enough, you can define your own validation result handling. Moreover you can track every step of validation process with jQuery events that are available through APEX dynamic actions.

What is important is the fact that one implemented dynamic action on **Magic Page 0** can handle APEX validations associated to all APEX items on all pages in your application. You are just few clicks from APEX live validation.

The plugin is available for free and works with APEX 4.x and 5.x.

2. Features at a Glance

- Live validation for APEX Items;
- Integrated with standard APEX item validations;
- Auto-executing validations on dependent APEX items;
- Easy to install and use (without single line of JavaScript);
- Easy to implement for entire application by dynamic action on Page 0;
- Ready to use templates for validation result handling;
- Possibility to define custom validation result handling through JS callback function;
- Possibility to track every step of validation through jQuery events;
- Runs on APEX 4.x and 5.x.

3. Roadmap

- Implementing debounce mechanism for "key release" event;
- New templates for validation result handling;
- Plugin attributes customization on application and page level;
- Handling APEX page validations (not associated with APEX items).

4. Feature Requests and Bugs

If you would like to see additional functionality or you have found a bug, please let us know - apex@pretius.com.

5. License

The **Pretius APEX Client Side Validation** plug-in is currently available for use in all personal or commercial projects under both MIT licenses. This means that you can choose the license that best suits your project and use it accordingly. Both licenses have been included with this software.

6. Legal Disclaimer

The program(s) and/or file(s) are supplied as is. The author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The author takes no responsibility assumes no liability for damages, direct or consequential, which may result from the use of these program(s) and/or file(s).

7. Installation and Configuration

7.1 Installation package

Installation package contains:

- plug-in installation file - **dynamic_action_plugin_pretius_apex_client_side_validation.sql;**
- this manual as PDF file;
- MIT license as text file.

7.2 Installation guide

In selected application:

1. navigate to **Shared Components > Plug-ins**,
2. click **Import** button,
3. choose installation file and proceed with installation steps.

During plug-in installation, you will be asked to confirm default value of **Global Error Template** attribute.

This attribute contains information how validation errors are displayed and is described in detail in section **Configuration > Plugin attributes > Application scope**. If you are not sure, just leave it as is.

7.3 After installation

Plug-in is ready to use!

Although you can use it "as is", we recommend to check configuration options and adapt it to your needs. In particular:

- Check your APEX Template and modify it as described in **Configuration > Template modifications**;
- Learn plug-in attributes in **Configuration > Plugin attributes**.

7.4 First steps with the plugin

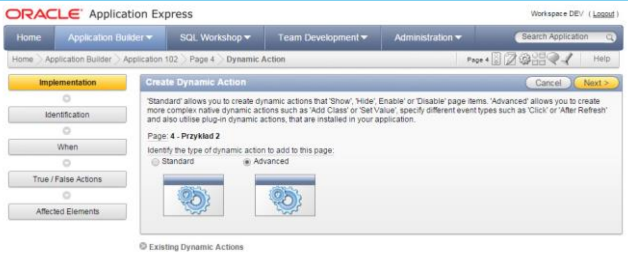
Before proceeding with this instruction please, be sure you have modified label template described in section Configuration > Template modifications.

Assuming you have APEX page with created APEX text items (eg. P1_CUSTOMER_ID) and standard APEX validations associated to the items (eg. P1_CUSTOMER_ID must be numeric and not null), proceed with following steps:

1 **Add Dynamic Action by clicking Create button and follow creator steps.**

2 **Dynamic Action – Implementation**

1. Choose type of dynamic action - **Advanced**
2. Click **Next** button



'Standard' allows you to create dynamic actions that 'Show', 'Hide', Enable' or 'Disable' page items. 'Advanced' allows you to create more complex native dynamic actions such as 'Add Class' or 'Set Value', specify different event types such as 'Click' or 'After Refresh' and also utilise plug-in dynamic actions, that are installed in your application.

Page: **4 - Przyklad 2**

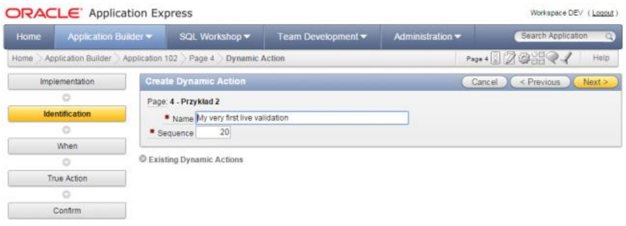
Identify the type of dynamic action to add to this page:

Standard [✓ Advanced]

Existing Dynamic Actions

2 **Dynamic Action - Identification**

1. Enter DA name, eg. **My very first live validation**
2. Click **Next** button



Page: **4 - Przyklad 2**

* Name: My very first live validation

* Sequence: 20

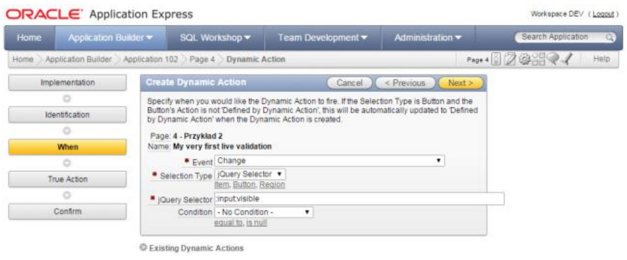
Existing Dynamic Actions

3 Dynamic Action - When

1. For **Selection** type - select **jQuery Selector**
2. For **jQuery selector** - type in **:input:visible** (it affects every visible text item)
3. For **Condition** - select **- No Condition -**
4. Click **Next** button

If you want you can also run live validation just for specific items:

1. For **Selection** type - select **Item(s)**
2. For **jQuery selector** - type in your APEX items (eg. P1_CUSTOMER_ID)
3. For **Condition** - select **- No Condition -**
4. Click **Next** button



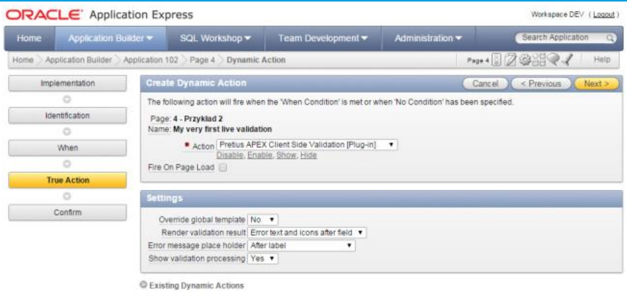
Specify when you would like the Dynamic Action to fire. If the Selection Type is Button and the Button's Action is not 'Defined by Dynamic Action', this will be automatically updated to 'Defined by Dynamic Action' when the Dynamic Action is created.

Page: **4 - Przyklad 2**
Name: **My very first live validation**

* Event	Change
* Selection Type	jQuery Selector (Item, Button, Region)
* jQuery Selector	:input:visible
Condition	- No Condition - (equal to, is null)

4 Dynamic Action - True Action

1. For **Action** - select **Pretius APEX Client Side Validation [Plug-in]**
2. For **Fire On Page Load** - unchecked
3. In **Settings** section set field as listed below:
    1. Override global template = No
    2. Render validation result = Error text and icons after field
    3. Error message place holder = After label
    4. Show validation processing = Yes
4. Click **Next** button



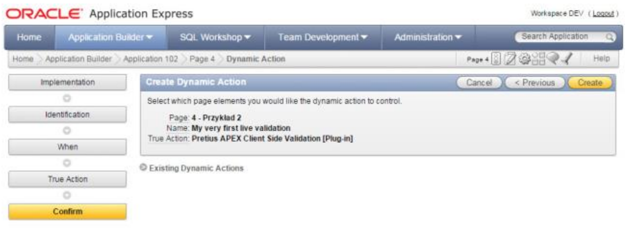
The following action will fire when the 'When Condition' is met or when 'No Condition' has been specified.

Page: **4 - Przyklad 2**
Name: **My very first live validation**

* Action	Pretius APEX Client Side Validation [Plug-in] (Disable, Enable, Show, Hide)
Fire On Page Load	☐
Override global template	No
Render validation result	Error text and icons after field
Error message place holder	After label
Show validation processing	Yes

5 Dynamic Action - Confirm

1. Finalize creating dynamic action based on the plugin with **Confirm** button



Select which page elements you would like the dynamic action to control.

Page: **4 - Przyklad 2**
Name: **My very first live validation**
True Action: **Pretius APEX Client Side Validation [Plug-in]**

Run APEX page and test APEX live validation.

7.5 Configuration

7.5.1 Template modifications

Important note

Label template is one of template types provided by APEX Themes. It describes the way the items labels are displayed in your applications and how your application displays errors referring to APEX items.

Modification of label template is recommended to avoid duplication of validation message. It can happen after page submit when you can see standard validation messages and plugin wants to show its dynamic validation messages in the same place.

The plugin executes APEX validations on-fly when chosen event (eg. click, item focus etc.) occurs without submitting the page. After submitting page the same APEX validations are executed by APEX engine.

Because of the fact, that APEX (in pre-installed templates) supports validation error template, the plugin needs to know where APEX displays error messages within DOM structure. If the plugin does not have reference to validation error message it can not remove validation message created by APEX engine. No reference to error message results in validation message duplication after executing on-fly APEX validation by the plugin.

To avoid validation message duplication, it is recommended to do simple modification in label templates. Please proceed with simple steps described in following sections for APEX 4.x and APEX 5.x.

Label template modification in Oracle APEX 4.x

Go to **Shared Components > Templates > Label templates**. In section **Error Display** are defined two attributes with default values like below

```
<div class="t1InlineError">
```

and

```
<br/>#ERROR_MESSAGE#</div>
```

The #ERROR_MESSAGE# substitution string is displayed as plain text after label and it can not be identified without knowing surrounding DOM objects. Because of that we need to add extra HTML tag with class around #ERROR_MESSAGE#.

To make the change just modify **Error Display > On Error After Label** attribute (example below) in every label template used in application.

```
<span class="apexBuildInErrorText"></br>#ERROR_MESSAGE#</span></div>
```

*Please keep in mind that, the code above is used to display validation error message after the label. To prepare label template with validation error message before label you need to move **<br>** tag after #ERROR_MESSAGE# but within span tag.*

Of course you can use your own tags and classes - the example is plugin default. When you want to change it you should also change the value of plugin Global Error Template attribute. The value of this plugin attribute and value of "On Error After Label" must be same!

Label template modification in Oracle APEX 5.x

In case you want to use the plugin with APEX 5 (Universal Theme) the only change needed is setting proper template in plugin global attribute scope. To make the modification proceed with following steps:

1. Go to Shared Component;
2. Go into details for used label templates;
3. Copy value from attribute **Error Template** in section **Error display**;
4. Paste copied value into plugin application attribute **Global Error template**.

If you use Universal Theme without any modifications, then default value you need to set is:

```
<span class="t-Form-error">#ERROR_MESSAGE#</span>
```

In case you use custom theme you need to proceed with steps 1 to 4, and if the template does not support direct #ERROR_MESSAGE# identification through CSS class you need to modify the template like in APEX 4.x.

*Please keep in mind that if yours DA overrides **Global Error template** in page scope, you need to set plugin attribute **Error template** to value from step 3*

7.5.2 Plugin attributes

The table below is summary of all available plugin attributes. Detailed description is presented in subsequent sections.

SCOPE	LABEL	TYPE	REQUIRED	DEPENDS ON	DEFAULT VALUE
Application	Global Error template	Text	Yes		<span class="apexBuildInErrorText"> #ERROR_MESSAGE# </span>
Component	Override global template	Select List	Yes		No
Component	Error template	Text	Yes	Override global template	<span class="apexBuildInErrorText"> #ERROR_MESSAGE# </span>
Component	Error message place holder	Select List	Yes		After label
Component	Show validation processing	Select List	Yes		Yes
Component	Render validation result	Select List	Yes		Error text only
Component	Callback function	Textarea	No	Render validation result	

7.5.3 Application scope

Global Error template

This attribute contains default error message template which is displayed after validation failure. Validation messages are rendered with this template when the plugin attribute **Override global template** defined on page is set to **NO**.

7.5.4 Dynamic action scope

Override global template

When this attribute is set to **Yes**, the **Error template** is used to render validation message. Otherwise validation message is rendered from **Global Error template** plugin attribute.

You can use this attribute if validations on specific pages should be displayed in different way.

Error template

This attribute contains validation message template, which is used to render validation message after performing validation on the fly.

This attribute is available when **Override global template** is set to Yes.

Error message place holder

This attribute defines where validation message should be located.

In APEX 4.x validation message is allowed before or after item label.

In APEX 5.x validation message is allowed before or after item label and before or after item.

Show validation processing

When this attribute is set to **Yes**, the image indicator is displayed after item field during validation process.

Render validation result

This attribute defines how validation result should be displayed. Available option are described below:

OPTION	DESCRIPTION
Error text only	Only the validation message is displayed after validation failure.
Error text and icons after field	Validation message and status image indicators are displayed after validation.
Error text and highlight field	Validation message is displayed after validation. Text field is highlighted. Highlight color depends on CSS classes *pretius_highlight_success* and *pretius_highlight_error.*
Custom callback function	This option allows developer to define custom validation result handling with JavaScript. When this option is selected, attribute **Callback function** is accessible. **Callback function** attribute is described in this document.

Callback function

This attribute defines custom validation result handling with Java Script function implementation. Function declaration is defined as below:

```
function(triggeringElement, itemName, validationResult, errorHTML, errorClass)
```

Function arguments:

ARGUMENT	TYPE	DESCRIPTION
triggeringElement	jQuery object	jQuery reference to validated item field
itemName	String	Validated APEX item name
validationResult	JSON object	JSON object containing information about validation result. The structure of JSON object is described under this table
errorHTML	HTML string	Validation error message rendered with error template (attribute **Global Error template** or **Error template**)
errorClass	String	jQuery selector to validation message

validationResult description:

OBJECT ATTRIBUTE	TYPE	DESCRIPTION
item	String	Name of validated APEX item (e.g. P1_CUSTOMER_ID)
message	String	Rendered Error Message if validation failed (plain text)
passed	Boolean	Validation result (true/false)
revalidate	String	List of APEX Items that will be revalidated after current validation ends (eg. P1_CUSTOMER_NAME,P1_CUSTOMER_ADDRESS)
time	JSON object	Execution time of PL/SQL validation process (object attributes are "ms", "seconds" and "minutes")
logs	Array of JSON	Each object of the array contains information about performed validations

validationResult.logs.logs describes each validation that was performed for APEX item. Description of JSON object in array:

OBJECT ATTRIBUTE	TYPE	DESCRIPTION	EXAMPLE
passed	Boolean	validation result	true
validationCode	String	APEX validation code	ITEM_NOT_NULL
validationCondition	String	Information about validation condition result	Condition [PLSQL_EXPRESSION] passed.

validationMsg	String	Validation message defined in APEX validation	Validation "P4_NAME not null" [ITEM_NOT_NULL] failed
validationName	String	Validation name that was executed for specific item	P4_NAME not null

Events

The plugin supports tracking each steps of validation process. All events can be bound through Dynamic Action (**Dynamic Action > When > Event > Component events**).

EVENT NAME	JQUERY EVENT NAME	DESCRIPTION
validation started	pretius_validation_init	DA is triggered when validation process begins
validation ended	pretius_validation_ended	DA is triggered when validation process ends
validation not found	pretius_validation_not_found	DA is triggered when validation for item is not defined
validation not passed	pretius_validation_failed	DA is triggered when validation result is failure
validation passed	pretius_validation_success	DA is triggered when validation result is success

Events in table are presented in execution order.

If dynamic action is defined as **Execute JavaScript Code** action, the JSON object is available to use:

```
this.data.validationPlugin
```

Attributes of object this.data.validationPlugin:

ATTRIBUTE	TYPE	DESCRIPTION
errorHTML	String	Validation error message rendered with error template (attribute **Global Error template** or **Error template**)
itemId	String	Validated APEX item name
label	jQuery object	jQuery reference to APEX item label
labelText	String	Label display text
message	String	Validation message defined in APEX validation
passed	boolean | null	**true** - success | **false** - failure | **null** - no validations defined for APEX item

7.6 FAQ

How to create validation on "Magic Page 0"?

Create dynamic action and use proper selector. For example *:input:visible*. For details please refer to **First steps with plugin** and recreate it on **Page 0**.

How does the plugin work?

Pretius APEX Client Side Validation starts on browser side, validates APEX items in PL/SQL and then returns result of validation back to user browser. Detailed steps are listed below:

1. Defined event occurs on specific APEX item (eg. change or click event).
2. AJAX call is initialized for APEX item.
3. Database contains information about validations, validation conditions, validation rules, APEX item settings and depending APEX items.
4. If APEX validation condition is matched the APEX validation is being executed (exactly like APEX engine does for standard APEX validations).
5. The result of executed validation is returned through the plugin API to GUI.
6. Validation result is handled according to the plugin settings.
7. Depended APEX items are validated (starting from Step 1).

If there are dynamic actions based on plugin events (eg. "Validation not found"), they are triggered when specific event occurs. For more information about plugin event please refer to section **Configuration > Events**.

Does the plugin work in APEX 5?

Yes! The plugin is compatible with APEX 4.x and APEX 5.x.

Does the plugin execute validation that is not associated with any APEX item?

Not yet, but we are working on it (check Roadmap).

So far your validation must be associated with some item.

What event is most suitable to execute validations?

It depends on the business expectations. For some items it is required to use **change** event, for some more suitable is **on lose focus**. Because the plugin can be assigned to group of APEX items, you can choose what event is most suitable to meet business requirements.

How to execute validation only on browser (GUI) side?

Due to integration with standard APEX validations, all validations are executed twice - on browser and on server side. We do not recommend to change this behavior (because of security reasons), but you can if you want.

To execute validations only on the fly, you must change settings for buttons and validations on selected page:

1. Buttons that submit the page - change **Execute Validation** attribute value to **No**;
2. Validations - change **Always Execute** attribute value to **No**. If you set this attribute to **Yes**, the validation is executed server side.

How to use callback function?

In this example callback function will be used to:

- color the text in input field to red when validation ends with failure
- display error message after label.

Validation execution point - **Change** event.

1. Create APEX input text field;
2. Create APEX validation assigned to created item - eg. rule alphanumeric.

Next step is setting dynamic action based on the plugin.

1. Create advanced dynamic action;
2. Choose from **Event** select list value **Change**;
3. Choose from **Selection Type** select list value **Item(s)** or **jQuery Selector**;
4. Choose created item if Selection Type is Item(s) or type in jQuery selector *:input:visible* if Selection Type is jQuery Selector;
5. Proceed to next step of dynamic action creator;
6. For **Action** choose **Pretius APEX Client Side Validation [Plug-in]**

7. In Settings section set attributes as below:

ATTRIBUTE	VALUE
Override global template	No
Render validation result	Custom callback function

Callback function	`var label = $('[for='+itemName+']');`
	`triggeringElement.parent().find( errorClass ).remove();`
	`label.parent().find( errorClass ).remove();`
	`if ( validationResult.passed == true) {`
	`  triggeringElement.css('color', 'black');`
	`}`
	`else if ( validationResult.passed == false) {`
	`  triggeringElement.css('color', 'red');`
	`  label.after( errorHTML );`
	`}`
	`else {`
	`  null;`
	`}`

Show validation processing	Yes

8. Finish creating dynamic action;

9. Test it!

Which types of validation are supported?

Supported

- [APEX item validation] Item is required
- [APEX item validation] Item is datepicker
- [APEX item validation] Item is number field
- Exists
- Function Returning Boolean
- Function Returning Error Text
- Item / Column specified is NOT NULL
- NOT Exists
- PL/SQL Expression
- SQL Expression
- Item / Column specified is NOT zero
- Item / Column specified contains no spaces
- Item / Column specified is NOT NULL or zero
- Item / Column specified is alphanumeric
- Item / Column specified is numeric
- Item / Column specified is a valid date
- Item / Column in Expression 1 equals string literal in Expression 2
- Item / Column in Expression 1 does NOT equal string literal in Expression2

- Item / Column in Expression 1 matches Regular Expression in Expression 2
- Item / Column in Expression 1 is contained in Expression 2
- Item / Column in Expression 1 is NOT contained in Expression 2
- Item / Column in Expression 1 contains at least one of the characters in Expression 2
- Item / Column in Expression 1 contains only characters in Expression 2
- Item / Column in Expression 1 does not contain any of the characters in Expression 2

Not yet supported

- PL/SQL Error
- Item / Column specified is a valid timestamp

Which validation conditions are supported?

Supported

- Exists (SQL query returns at least one row)
- PL/SQL Function Body Returning a Boolean
- Value of Item / Column in Expression 1 Is NOT NULL
- NOT Exists (SQL query returns no rows)
- PL/SQL Expression
- SQL Expression
- Value of Item / Column in Expression 1 != Zero
- Value of Item / Column in Expression 1 Contains No Spaces
- Value of Item / Column in Expression 1 Is NOT null and the Item Is NOT Zero
- Value of Item / Column in Expression 1 Is Alphanumeric
- Value of Item / Column in Expression 1 Is Numeric
- Value of Item / Column in Expression 1 = Expression 2
- Value of Item / Column in Expression 1 != Expression 2
- Not yet supported
- Always
- Value of Item / Column in Expression 1 Is Not Numeric
- Value of Item / Column in Expression 1 Is NULL
- Value of Item / Column in Expression 1 Is NULL or Zero
- Value of Item / Column in Expression 1 = Zero
- Never
- Value of Item / Column in Expression 1 Is Contained within Colon Delimited List in Expression 2
- Value of Item / Column in Expression 1 Is NOT Contained within Colon Delimited List in Expression 2

Not supported

- Text in Expression 1 != Expression 2 (includes &ITEM. substitutions)
- Text in Expression 1 = Expression 2 (includes &ITEM. substitutions)
- Text in Expression 1 Is Contained within the Text in Expression 2
- Text in Expression 1 Is NOT Contained within the Text in Expression 2
- Inline Validation Errors Displayed
- No Inline Validation Errors Displayed
- Current Page Is in Printer Friendly Mode
- Current page is NOT in Printer Friendly Mode

- Current Page != Page Submitted (this page was not the page posted)
- Current Page = Page Submitted (this page was posted)
- Client Browser: Microsoft Internet Explorer 5.5, 6.0 or higher
- Client Browser: XHTML / CSS capable browser
- Client Browser: Mozilla, Netscape 6.x/7x or higher
- Client Browser: Other browsers (or older version)
- Current Language = Expression 1
- Current Language Is Contained within Expression 1
- Current Language != Expression 1
- Current Language Is NOT Contained within Expression 1
- When CGI_ENV DAD_NAME = Expression 1
- When CGI_ENV DAD_NAME != Expression 1
- When CGI_ENV HTTP_HOST = Expression 1
- When CGI_ENV HTTP_HOST != Expression 1
- SQL Reports (OK to show the forward button)
- SQL Reports (OK to show the back button)
- Request = Expression 1
- Request Is Contained within Expression 1
- Request != Expression 1
- Request Is NOT Contained within Expression 1
- When CGI_ENV SERVER_NAME = Expression 1
- When CGI_ENV SERVER_NAME != Expression 1
- User is Authenticated (not public)
- User is the Public User (user has not authenticated)
- Value of User Preference in Expression 1 = Expression 2
- Value of User Preference in Expression 1 != Expression 2

8. About Pretius and the Authors

8.1 About Pretius

Pretius is a software development and IT consulting company operating in Poland. Our company was founded with a vision that corporate architectures and business solutions must be simple and user-friendly to be able to add value instead of being a costs center.

We engage in lasting relations with our clients; this helps us to even better match the software to their needs. We deliver IT solutions to corporate customers present in telco, energy, media and finance markets. Our customers include internationally acclaimed brands and leaders in respective verticals in the polish market. We build innovative business solutions and tools supporting work of multiple units in big organisations, e.g.: sales, sales support, back office, call center. Our solutions include: CRM, Sales Commission, KPI Dashboard, Workflow, Sales Partner Portal.

For IT departments we deliver solutions such as Enterprise System Service Bus, Operational Data Store and Machine to Machine.

As an Oracle Gold Partner we use Oracle stack technologies as base for our solutions (Oracle Application Express, Java, Oracle database).

CONTACT US	FIND MORE
Pretius Sp. z o.o. Sp. K. Okopowa 56/1 01-042 Warsaw, Poland Email: sales@pretius.com WWW: http://pretius.com/	@PretiusSoftware facebook.com/pretius youtube.com/c/pretius

8.2 About Pretius APEX Department

Pretius APEX Department is a team of APEX specialists supported by database and frontend developers.

Pretius is a pioneer in Poland in building solutions based on Oracle Application Express. We have developed APEX applications since 2008, starting from APEX 3.0.

We provide custom applications and two APEX based ready-to-deliver products. Our APEX applications are located in the middle of our clients business processes, and very often are crucial in fulfilling their business needs.

8.3 About the Author

AUTHOR	POSITION	E-MAIL
Bartosz Ostrowski	Oracle Application Express, PL/SQL & Frontend developer and IT consultant. Experienced in developing database and web applications using Oracle Application Express. Oracle APEX enthusiast, Oracle APEX plugin developer and active community member. Specialties: Oracle Application Express (Oracle APEX), javascript, PL/SQL, Web based applications.	bostrowski@pretius.com