ORACLE APPLICATION EXPRESS: TALES FROM THE TRENCHES

Dr. Paul Dorsey, Dulcian, Inc.

Oracle Application Express (APEX) gains ground every year with the Oracle development community and has received a lot of attention in the IT press and beyond. But how are people really using the tool? What are they using it for? How happy are they? This paper summarizes the experiences of leading APEX users and provides an overview of their usage and perceptions of the APEX development environment for building and maintaining production systems.

APEX is largely perceived as being an easy-to-learn, easy-to-use, productive, capable environment. However, not all users are satisfied, nor are all projects successful. For this paper, I reached out to interview a group of active APEX users and asked them about their experiences. This paper is intended to give new and prospective APEX users a view into what awaits them in the APEX development environment.

Overall, APEX users are very happy. Even former Oracle Forms users are satisfied with the product. The advances that the product has made over the years have only increased user satisfaction. That said, APEX is not nearly as simple to learn or as straightforward to use as is popularly thought.

LEARNING APEX

The responses about the time needed to become productive with APEX were mixed. The background of the persons learning APEX had a lot to do with the ease with which the tool could be learned.

Those with existing PL/SQL skills and experience with Oracle Designer, Forms, Reports, etc. were able to learn the tool more quickly and become rapidly productive. It was also mentioned that some knowledge of HTML, CSS, JavaScript and AJAX was useful in getting up to speed with APEX. The main challenges mentioned seemed to revolve around problems making the conceptual switch from the traditional database master/detail or Visual Basic design approach to a web environment.

Learning APEX is not nearly as automatic or easy as one would hope. Learning to use the basics of the tool seems to require about three months, although some users reported that the time required to get up to speed could be measured in days or weeks. Becoming an expert takes a year or more. There are two reasons for this:

1. APEX's use of a point and click, repository-based IDE means that every new feature set requires a new screen in the IDE. Where early versions of APEX were easy to learn but only included limited functionality, as the tool has evolved, it has become more capable as well as more complicated. There are currently over 2000 screens in the IDE. This fact overstates the tool's complexity since most screens are easy to find or in wizards, but it does illustrate the fact that APEX is no longer a compact, simple product.
2. Very few users simply stick to the APEX IDE. Almost everyone goes "outside of the box." APEX is simply not deep enough to satisfy most developers' desire for control. Hence, expert users are forced to learn to use the APEX APIs, JavaScript and HTML.

One of the most disturbing findings was that some old Oracle Forms users were unable to make the transition to APEX. Even after a year of trying, they were unable to get the hang of the tool. It may be that those particular developers would be unable to easily transition to any other tool because some people have trouble learning new things, but this does seem to indicate that organizations moving to APEX should not expect that everyone will easily make the transition.

Using APEX

APEX's flexibility of supporting custom JavaScript and HTML templates is an important extension to the tool but has also provided the tool with a great propensity for abuse. Development teams that do not stay within the confines of what APEX is good at, but instead force the tool to do exactly what they want it to usually end up writing lots of custom code that is very complex and ultimately may lose all of the advantages of using the APEX environment. Perhaps most disturbing are the teams that are clearly not using APEX to its best advantage and are completely unaware that they are working against the tool. The most common excuse is that "our users are very particular and will not allow us any latitude in the UI design".

Planning ahead is essential for any APEX team. APEX is not simply a "code and go" kind of tool where your prototypes nicely evolve into your production system. Careful, detailed analysis of technical requirements must be completed prior to finalization of the technical architecture for a project. Otherwise, major refactoring of the code may be required.

The core problem is that APEX has multiple ways of implementing business rules in the product. Development can be done in a number of ways:

1) In the IDE
2) Using the APIs
3) In JavaScript
4) In HTML templates

What gets done where needs to be decided early on in the project. An example will help to illustrate the problem. The code for enabling and disabling button is done within the IDE during development. Late in the development process, it is decided that button enabling must be contingently based upon complex security requirements that go outside of the APEX IDE capability. All of the impacted rules that were placed in the IDE would now have to be refactored into API calls so an extra IF THEN statement could be wrapped around the code.

Things that bother me but do not bother the users interviewed

There are some weaknesses in APEX that do not seem to bother the majority of the development community. A brief description is included here.

No Model Layer

There is no real model layer in APEX. As a result, users cannot commit whole transactions at once. All interaction is done one screen at a time (unless the developer does hand coding). Paging on a grid involves a requery each time a new page is requested, so users run the risk of never even seeing a particular record when paging through a grid.

Weak Layout Model

Without lots of hand coding, developers cannot really control the screen layout. Developers lack options for manual layout, positioning fields by X/Y position, and the ability to nest panels. Users either go outside of the IDE to make the screens they want, or live within the limitations of the layout model.

Inefficient Database Usage

Behind the screens, APEX is not nearly as efficient as it could be. A great deal of unnecessary dynamic SQL is being executed, with many queries being re-executed, etc. The Oracle database is so strong and the APEX thick database approach is so much better than the way in which most JavaEE architectures are built that no one seems notice the inefficiency.

How Is APEX Being Used?

I did not find APEX being used for any COTS or very large production systems. But it is being used to deliver serious larger applications and not just being used for prototypes or small, non-mission critical systems. Systems with thousands of screens and hundreds of users are clearly within scope of the tool.

budget and/or manpower required

It was interesting to note that most of the projects involved relatively small teams and budgets of $500,000 or less. The longest project cited took four years, but only used one consultant. Most of the others required less than one year with teams ranging in size from 2-15 people.

Reporting

Reporting seems to be the elephant in the room. No one claims to have a good solution to generate printed reports.

User Feedback

The following are some of the "more or less" raw responses from the questions posed to the users whom I interviewed. These are not specifically identified and provide a more in-depth picture of the product usage. The responses are provided these with no editorial comment.

1. What types of projects is APEX being used for?

The interviewees for this paper were asked to briefly describe the projects for which they were using APEX. The responses varied greatly:

- Creating a central "Web Portal" to service multiple lines of businesses in order to provide reporting and automated business function(s)/processes. The project was initially scoped to consolidate legacy reporting processes (including multiple uses of spreadsheets and databases scattered in different locations and based multiple versions of the same dataset) into one centralized system in order to improve critical decision making and to provide a consistent foundation for the data such decisions were based upon.
- Project for a state department that involved 100 users. There were approximately 4090 screens, 100 tables and the desired output was PDF or HTML. Dynamic SQL was used along with hand coded XSL stylesheets and database packages.
- Financial management system
- A real time monitoring system for a manufacturing test facility where multiple tests run continuously. The APEX portion of the project sets up the tests and provides a user-friendly interface for starting and stopping the machines. The actual starting/stopping is performed by other software and presented graphically in the APEX front-end.
- Replacement of ERP-like legacy logistics system containing order entry, shipments, planning, financial information based on an Oracle database into which all of the business rules were coded. The old

fashioned front-end needed to be replaced due to lack of knowledge and support. An automatic conversion tool was used for this process.

- Functional Migration project for a large bank, to move 60 Oracle Forms dialogs and 100 Microsoft Access dialogs to APEX 3.2.1. For reporting purposes (25 PDF reports), SAP Business Objects was integrated into the system.
- Permit system for a state government with 300-400 internal users. The system required external web publication of permit data and replacement for an existing system plus new functionality.
- Lending and Signing Authority system for a large US bank with 4000 users to replace an existing system and add functionality. This system had no external facing pages. It was designed to track lending and signing authority for loan underwriters, certification and re-certification training courses for each underwriter, recommendations, and approvals for changing lending authority levels for an underwriter.
- Simple system for internal use and utility used to gather new electronic product build information and past purchasing history to develop vendor quotes and proposal estimates. Project requires tracking and justification for government and customer review.
- Medical diagnostics firm suite of six applications with 700 users each but only about 100 simultaneous users. This system also included some quality control applications. Some of the information was gathered, some uploaded, some direct data load. The system needed to handle items of interest, with 2-5 levels to be retrieved, enter comments, and attach documents.
- Large enterprise application used across the United States with 3000 users, averaging 100 concurrent, 100 tables with 2 million records in the largest table. The project consisted of converting three systems into a single integrated platform.

2. How hard was it to learn APEX and become proficient with the tool?

The time taken in order for the various consultants/development teams to become productive with APEX ranged from 2-3 months to 1 year or more.

- APEX was fairly new during the beginning of the implementation therefore, the team relied on the Oracle forums, online tutorials, and leveraging existing PL/SQL skill sets. The development team became proficient with the tool in approximately three to four months.
- Existing skills in PL/SQL, Forms, Reports - Learning Curve ~ 2 applied months to become proficient.
- Learning the APEX environment was not difficult at all. In fact, the declarative nature of the development eliminated a tremendous amount of the learning curve. However, it is helpful to have a background in web technologies along with web development experience. The time to produce a base application in the project took less than three months.
- Some resistance from users who didn't think APEX was capable of handling larger applications. Took about 12 months to train
- It doesn't take that long to just start building your first screens, reports etc. More days than weeks. (Have a thorough background in SQL, PL/SQL, Designer, Developer and that helps a lot). But still learning new "tricks" or solutions every day.
- This depends on what you want and how fast you are able to learn things. To become a basic APEX developer using the wizards, you will need a few weeks. To become an APEX expert you will need multiple years.
- Becoming **proficient** with the APEX IDE is a multi-step process:

1. Work through the 2-day developer tutorial and you are solidly on first base. At this point, a junior developer can do productive CRUD work.
2. Build one or two sand box systems where you play with the "out of the box" GUI features. This takes about 2 or 3 person-weeks. At this point you are on second base and capable of producing an "acceptable" GUI.
3. Becoming **expert** (third base and home) with APEX depends on your knowledge of APEX's supporting technologies like SQL, PL/SQL, HTML, CSS, DOM, JavaScript, AJAX, BI Publisher, OS scripting tools, Oracle database features etc. If you are expert in the supporting technologies then it is not too hard to figure out how they are integrated into the APEX environment. The time estimate before you can user the moniker of "expert" is probably a year or two (assuming that you are already proficient with the supporting technologies). By the way, there is now a large body of sample APEX tips and tricks floating about the web.

- One difficulty I had with APEX was getting rid of my Visual Basic design approach. For quite a while, I would think of how I would design a GUI in Visual Basic and then look for the equivalent in APEX. A better approach is to use APEX in the way the APEX authors meant it to be used. Learning from the APEX development tool is a good start. I suspect that many other current APEX developers suffer from the "how would I do this in my previous tool" syndrome which could explain why some developers use JavaScript instead of the native APEX tool.
- Trained two users, one with Forms background, PL/SQL SQL took about 3 months. No HTML, CSS or Java Script The second Forms developer could not make conceptual switch from master/detail to HTML environment – wanted to do a lot of customization Failed after 8-9 months trying to build a small allocation management system
- ADF team failed twice – Java team too slow - $500,000 budget
- As an early user of APEX (HTMLDB/Project Flow), at the start it was more limited and to get up to speed took me a couple of days. Obviously I gained more experience in the next months and know better how to do it. But depending upon previous knowledge (PL/SQL etc) it's a matter of days or weeks, but certainly not months.

3. How much of the system was built out of the APEX IDE? How much custom coding was needed and why?

Most people responded that they tried to stay within the APEX IDE as much as possible. Doing this greatly reduced development time and cost of the projects. The APEX version used also impacted the amount of hand coding needed since many improvements were made to the 4.0 version.

- 90% of the screens in the application were built using the APEX IDE. The remaining 10% include additional technologies to improve the user experience. The technologies integrated into the application consist of JavaScript, AJAX, and JQuery. The project also modified the CSS templates to customize the look and feel in order to meet the customer needs. The project strategy maintained the practice of using PL/SQL for complex business logic and data intensive processing. JAVA was also used for integrating with existing enterprise applications such as the corporate content management system.
- 90% - avoided customization because of upgrading. Most customization involved opening PDFs, calling procedures. Missing some IDE elements: cascading LOV, enable/disable tabular forms, four regions 1 master/detail, 2 tabular forms (hand coded), Use APEX API, build PL/SQL to process
- The system was built mostly by using the IDE. However, some bits were hand coded:

    - **Dynamic Page Structure:** Some pages were constructed using the APEX API to build dynamic pages that captured data in PL/SQL collections before committing the changes to the database. This was done because the users wanted to see the data presented in a non-APEX way.
    - **Navigation:** The users wanted to be able to jump directly from one context to another. To do this, hierarchical drop-down menus were built in JavaScript. They were invoked by clicking on cells in a report.
- The system uses JavaScript to almost fully rewrite the DOM. The client was an early adopter of APEX and has between 200-300 APEX systems in production. The JavaScript rich interface was likely developed to overcome deficiencies in the earlier versions of APEX. Another reason for the heavy JavaScript is to make the APEX applications look and work like existing non-APEX applications. The initial cost of the JavaScript development has been amortized over 200 to 300 APEX applications that are used by thousands of users. The JavaScript is nicely packaged so the developers plug them in and do not have to worry about it.
- With APEX 3.2, this felt like 70%. With APEX 4.0 more is possible declaratively, maybe 80%. With APEX 3.2, a lot of "corrections" in the user interface were needed using JQuery.
- Usually use an external IDE (SQL Developer or similar) to build the tables and the PL/SQL stuff. You can do that in APEX as well, but SQL Developer is more appropriate and handy. JavaScript, CSS etc is all written inside the IDE (using the Dynamic Action features and Plugins). In production the JS and CSS is stored on the webserver. For large pieces of JavaScript (very very rare) I usually use a real JavaScript editor.
- Less than 20% - otherwise lose rapid development advantage. Push users toward native functionality
- Initially the whole system was built using the APEX IDE. After reaching some of the limitations of the built in CSS and JavaScript libraries, our team ventured into creating additional scripts and modules that better fit the end user requirements for interactivity and faster response within the application. Currently, the development has evolved into the APEX API in order to utilize the power of APEX and cascade some of the API wrappers to empower some end users that may be in a management role in which they needed more control or flexibility.
- Project is mostly data management and handling. Extraction and cataloging of historical data for retention requirements, cataloging of assumptions made for estimates. Most of this type of coding was done through PL/SQL - API construct. The design of the project was broken into a "use case" where data construction was managed separate from the user interface with event calls used when interaction was needed. The user interface had more to do with gathering assumptions the data collection process would use, reporting results, requesting quotes (API to Oracle eBusiness) and making adjustments to those results and recording the justification for the adjustments.
- The project started with an out-of-the box screen, but was enhanced with AJAX, JavaScript, PL/SQL, etc. With APEX 4, we try to re-engineer to use Dynamic Actions so we have less JavaScript as with APEX 3.x.

4. WHAT ARE THE GREATEST STRENGTHS OF THE PRODUCT?

There was agreement that the ability to develop and deploy quickly were the most significant strengths of APEX.

- Deployment speed
- Tight interactivity with the database

- The ability to extend the platform with API's, Plug-ins and the use of utilities that are native to the Oracle database.
- Ease of use
- Development speed (mentioned numerous times)
- Level of productivity – easy to demonstrate to client
- Can fix and roll out changes into production in 30 minutes
- No performance issues – some DB tweaking needed
- Capitalize on PL/SQL experience
- Easy to demo to users and make changes on the spot.
- Rapid Application Development. APEX let's you do a lot more in less time. We did a comparison APEX vs. Google Webtoolkit. Outcome: with the same functional specifications, you will only need 1/3 of the total development time when using APEX.
- The tool is written in PL/SQL which is tried, true, debugged, tested, stable and optimized. The PL/SQL core is one the big reasons that APEX appears to be rock solid.
- It's easy to get started.
- The "out of the box" configurations have very few moving parts.
- Project cost estimates and schedule estimates can be accurate for "out of the box" configurations.
- The "out of the box" themes produce an "acceptable" GUI.
- Fast to get up and running
- Easy to create prototypes to show to users
- Easy to create and install a bunch of PL/SQL packages
- Can change things in production without too much difficulty
- Version 4 includes plug-ins
- Nice Web features - IDE more flexible – easier to change
- The APEX toolset provides huge benefits by offering rapid development for data driven applications, increased efficiencies with an intuitive development environment, and offered a road map leveraging existing corporate PL/SQL development skill sets. The other strength consists of deploying changes very quickly thereby delivering a quick turnaround.
- Simplicity of architecture

5. WHAT ARE THE GREATEST WEAKNESSES OF THE PRODUCT?

For those who stayed within the "box," few weaknesses were cited. Most difficulties occurred when trying to do something that APEX was not explicitly designed to do. Because of the myriad screens and elements contained in APEX, some cited problems finding the appropriate elements to use or difficulties when debugging.

- The project experienced challenges when trying to use the delivered IDE components and still deliver a rich client experience. Complicated requirements drove the team to depend on other technologies to finalize the end requirement. The second disadvantage was the authentication integration as the team learned how vendor lock in could present obstacles. Nevertheless the team overcame the challenge and utilized database technologies that complemented APEX.
- None if you stay within the limits of the tool.

- Need some knowledge of CSS, JavaScript, HTML
- Sometimes difficult to find the correct element
- Can't redesign entire page
- One page, one data source
- HTML interface
- Managing the source code is awkward.
- Debugging is not intuitive.
- It is restricted to data oriented web applications.
- Version Control, Configuration Management and Large scale (many developers) development. You can do all that, but you have to follow some strict procedures (exporting, locking etc). This is all due to the fact that there is no "source code", but only "data".
- APIs are not well documented.
- It is hard to find qualified APEX developers.
- Limited reporting capability
- Quirky issues with some data types (dates) and the inability to actually get into the nuts and bolts of the extra features such as Interactive Reports.
- No real documentation for the Flow_API that would enable us to program more complex business functions and connect to Oracle database utilities.
- Distance between the wizard and custom requirements. Writing HTML output through PL/SQL is a major drag.
- Styling - but I don't think that is unique to APEX.
- Reporting is one of the major weaknesses of APEX. BI Publisher works really well with APEX but the price tag is too rich for many shops.

6. Compare developing with APEX to Oracle Forms, JavaEE or other environments

For former Oracle Forms and Reports developers with a PL/SQL background, the APEX environment seems to be a bit easier to learn than for those with other skills.

Difficulty in becoming proficient

- Forms vs. APEX - Both are very database centric. Forms had the complexity of regions and windows that needed to be managed. This was a steep learning curve. Also, working in the eBusiness environment, you had all of the pre-defined rules and regulations to deal with and learn in order to be effective. APEX, as a blank slate has a lot less of that type of definition. Because the tool is the "rules and regulations", it does not seem so much of an "extra" hindrance.
  The key fact that the deployment of APEX applications does not come close to the complexity of forms makes the difficulty to become proficient less.
- APEX is much easier than JDeveloper, Forms or J2EE.
- Not skilled enough (anymore) to compare thoroughly and it is hard to compare a web technology with e.g. Forms. Not necessarily all Forms applications could be APEX. But APEX learning curve is definitely easier than both others

- Forms as we now know it seems simple, but initially, it was very complex. All those triggers firing at some point it time.... APEX is far easier.
- Forms: Not difficult as it is 100% PL/SQL and not object oriented; Java: more difficult to first understand the oracle database
- APEX provides the same ability to create dynamic prototypes, fast development, and scalable Oracle applications as a Forms application. The programmer's productivity with meeting customer demands are about equal however, APEX has the upper hand with delivering a fast, robust, web interface.
- The ability to become proficient with APEX does depend on the programmer skillset. Java developers with minimal database knowledge tend to tackle business problems with a different mindset. This project overcame the transition by leveraging the skills of the database programmers to increase productivity in the beginning stages. After the team realized the power of two technologies the performance of the application and the team were superior. The quality of the application is improved because APEX can benefit from J2EE developer fundamentals like write once, create extensibility, and then make use of the database when appropriate.
- The J2EE application developer was able to see how a shared development environment can improve productivity and performance. Since most Java developers in our enterprise work in a local sandbox the APEX platform improves the team synergy and shared resources.

Productivity

- Forms vs. APEX: Higher productivity with APEX - as reporting is part of the integration. It is easy to add APEX reports without the complexity of Forms having to generate new windows/regions in forms. Multiple page layouts for web development flow more easily in APEX than in Forms.
- Event driven - this is one place where Forms was simpler to work with. It was all event-driven. This provided more flexibility. This also bodes well for the .NET style framework. That type of direct interaction and event behavior is easy for a human to understand and work with.
- Productivity does not always mean quality.
- Need some CSS and JavaScript knowledge
- APEX is more productive than Forms. Our experience is in APEX you need 1/3 of the time compared to Forms.
- Java is not productive; Forms is more productive; APEX is a bit more productive
- Very productive – can create 1 screen or more per day
- Productivity the same, unless it's a huge application maybe. I've found APEX having a better performance than the two others (Web Forms and BC4J). Compared to native Forms desktop application it is not a fair comparison and different use case.
- As with any development environment the mindset and skillset of the developer will depend on the performance, productivity, and quality of the application. Nevertheless, the J2EE developers were able to adapt to the environment with minimal impacts. Furthermore, the developers were able to capitalize on the fundamentals of Java like security, which allowed for complex interface integration necessary outside the database footprint.

Performance

- Interpreted as: How does the tool preform relative to the users? - I don't see any major drawbacks in this area to any tool. At the end of the day - the results should be the same. The question is: how long did it take to get the same results with the same performance? I believe all the tools have ways of managing performance issues to produce ideal results. When dealing with the database, I believe that the closer you are to the database, the easier it will be to accomplish higher speed results. In this case APEX is the closest to the database out of the Forms and JavaEE pool.
- No performance issues (repeated several times)
- APEX needs way less resources than Forms. With the same server configuration, you can serve 10 times more users than Forms.
- Java uses a lot main memory; Forms needs an application server as well; APEX runs 100% in the oracle database and performs very well
- APEX is a wee bit slower because the page refreshes hit the database

Functional Quality of Applications

- Quality is in the development design and approach. I do not believe the tools (Forms, Java, or APEX) can be held responsible for quality.
- In Forms you could do pixel perfect positioning. You can't do that in APEX (or you don't do that). But until now, I haven't encountered a functional requirement I couldn't build. Especially with the new Dynamic Action and Plug-in features you can do a lot of tricks (AJAX calls etc) a great lot easier than in the 3.x version.
- For data driven web applications, APEX should be the first choice; for client/server apps used by internal employees, Forms is in the lead; for other scenarios or special requests, Java can be considered.
- This is a wash, some things are better in APEX, some worse.
- Quality depends on the developer. It's not because APEX is "easy" to learn, you are a good developer with high quality. So I think it depends the developer whatever technology they use

7. Are you happier with APEX than with other development environments YOU HAVE used?

All of the interviewees expressed satisfaction with the APEX development environment. They liked the ability to rapidly develop applications that can be shown to clients and easily modified to meet user requirements.

- Happier (three replies)
- Have only used APEX environment – very happy with it.
- A lot happier (two replies)
- Eager to upgrade to APEX 4.0 – utilize new features/plug-ins, search feature. Would never go back to Oracle Forms
- Compared to Forms – easier to train
- Can create a template and copy pages and elements
- More happy and only do that now (used Forms, Java, etc before)

- The team is happier to have another tool in the toolbox. The ability to have multiple options to better satisfy the customer improves competitiveness; technological skill sets, and creates opportunity with improving the enterprise.

8. How hard is it to make APEX build the kind of applications you want?

- If the business challenges involve multiple platform and non-database driven requirements, APEX may not be the ultimate software solution. This is why having additional frameworks and technologies allow an enterprise to respond to business challenges appropriately. APEX cannot solve all business needs but can reduce time and complexity with data driven applications.
- Need to stay within the tool – do as much as possible in the database.
- Once you get your head around the way APEX works, it is not hard to build dynamic GUIs by using the APEX API together with PL/SQL collections.
- Not hard, as long as you follow the guidelines and has some experience with the product.
- Not that hard. As it is in fact an open framework, you can build whatever you like. You can incorporate Java applets (I incorporated Oracle Forms inside APEX pages), tons of graphical stuff, maps, pictures etc without a lot of effort.
- Took 3 weeks to turn an old MS Access program into a robust APEX application
- It's not necessarily APEX, but rather the amount of creativity placed into a project. In general, APEX doesn't limit us beyond the typical limitations presented in the web environment.
- You have to design the application in an APEX style. If you don't, then you lose a lot of productivity, but still it can be done, but it's probably not what you really want to do.
- WANT = Requirements needed for our business
  Fairly Easy. Most requirements are data manipulation and reporting. We are not selling anything to the public, so want normally comes with a business need. Business needs to record, analyze, and report.

9. How hard are the debugging and modification processes in APEX?

- Debugging APEX applications in the beginning versions was very challenging. The team takes advantage of other tools to improve the testing and debugging capabilities. With each new version and as more products become available, debugging becomes easier. For example Firefox offers FireBug which provides insight into the web portion and the integration with SQL Developer proves to be a life saver for the applications. There is no silver bullet tool to cover all areas of debugging. Web 2.0 applications rely on so many different technologies, so APEX is not as easy to refactor as a J2EE application.
- There are a lot of ways to debug an APEX application. I find it very convenient to quickly analyze problems without having to deploy my application over and over again and again and again.
- The answer has to do with standards. Where to code what. There are more ways to meet some requirements. So you have to set up some standards around that. Once you've done that, it isn't that hard. For instance, we use as few PL/SQL blocks as possible inside APEX. Inside APEX, we just use calls to package procedures or functions. That makes it a lot easier to code, test and re-use.
- Not difficult at all, we have even built some customized debugging tools that work along with the built in functions of APEX.
- With APEX being stateless, it is pretty easy to see what events are happening when. If the application is written with basic utility in it, narrowing down an event is pretty straight forward.

Debugging

- **Out of the Box:** This is fairly easy to do.
- **Supporting Technology:** You need external tools (e.g. Firebug) to debug JavaScript. Some of the commercial APEX training companies offer explicit courses for debugging the APEX environment. It is a matter of having a good set of tools to manage and debug the various environments (SQL, PL/SQL, JavaScript, AJAX etc).

Modification

- **Out of the Box:** This is fairly easy to do. It is simple to delete a page and rebuild it from scratch.
- **Supporting Technology:** If you program well and centralize your code in PL/SQL packages and JavaScript files, then you are in good shape. If you shot-gun your code throughout the individual APEX pages, you are lost.

Refactoring

- **Out of the Box:** This can be really tedious. For example, if you use APEX validations that exist on each page, then to change something like the way you validate dates, you need to open each page and repeat the same mouse clicks again and again and again. In theory, you could export your application, make the changes in the export file and then re-import the application. But god help you if you code a syntax error, you could seriously blow up your APEX instance.
- **Supporting Technology:** Same comment as above. If you program well and centralize your code in PL/SQL packages and JavaScript files, then you are in good shape. If you shot-gun your code throughout the individual APEX pages, you are lost.
- If you know how to debug, it's not that hard, but it can be complex in a web application as you have to debug JavaScript, HTML, CSS, PL/SQL backend, SQL etc.

10. How do you handle reporting in your APEX applications?

Responses to the issue of reporting in APEX systems were diverse. Several respondents mentioned the prohibitive expense of BI Publisher.

- Reporting in the APEX sense is any information on the page. In many cases, the information relevant to a job or decision can be displayed on the screen. Our project required exported PDF documents. We purchased BI Publisher when it was available in standalone version for our business. This has proven to be a powerful and useful tool for printed and exportable document generation. We utilize the integration of the BI Publisher tool with APEX to generate those types of reports.
- Technically speaking, the integration of other Oracle database modules such as the DBMS_Scheduler and the use of stored procedures have enabled the ability to create timed reports that meet reporting timetables and deadlines for daily, weekly, and monthly reporting. The integration with other Oracle reporting engines such as BI Publisher has enabled advance creations of customized report templates and distribution models like report bursting via fax, email and FTP. Additional toolsets provided by Oracle such as the Microsoft Word and Excel plug-ins enable a greater efficiency in creating customized report templates that ensure a consistent look and feel for each report.
- 4000 lines of code in one report
- Reporting in the legacy system was done using Crystal Reports and it was decided to keep it that way. We access/run the reports using the Crystal server (all URL based).
- APEX includes an option to download the data in Excel. Some simple reports are done in HTML. We don't and won't use BIP. If we need any more sophisticated reports we will probably use a "cheap" alternative like Apache/FOP or Jasper Reports.
- Use a series of pages – query with options. PL/PDF standard FOP – no BI Publisher
- The users are happy with the CSV dumps that APEX provides. Form letters have been hand-coded.

- Most of the reporting has been farmed out to another MIS department that handles reporting for the entire organization. I think they use Crystal Reports. Form letters will be generated using BI Publisher.
- In general, the APEX users have been bugging Oracle's BI Publisher group to release a "BI Publisher Lite" for APEX that is much less expensive than the current $40,000 price tag. So far, the pleas have fallen on deaf ears.

About the Author

Dr. Paul Dorsey is the founder and president of Dulcian, Inc. an Oracle consulting firm specializing in business rules and web based application development. He is the chief architect of Dulcian's Business Rules Information Manager (BRIM®) tool. Paul is the co-author of seven Oracle Press books on Designer, Database Design, Developer, and JDeveloper, which have been translated into nine languages as well as the Wiley Press book *PL/SQL for Dummies*. Paul is an Oracle ACE Director. He is President Emeritus of NYOUG and the Associate Editor of the International Oracle User Group's SELECT Journal. In 2003, Dr. Dorsey was honored by ODTUG as volunteer of the year, in 2001 by IOUG as volunteer of the year and by Oracle as one of the six initial honorary Oracle 9*i* Certified Masters. Paul is also the founder and Chairperson of the ODTUG Symposium, currently in its eleventh year. Dr. Dorsey's submission of a Survey Generator built to collect data for The Preeclampsia Foundation was the winner of the 2007 Oracle Fusion Middleware Developer Challenge and Oracle selected him as the 2007 PL/SQL Developer of the Year. Paul can be contacted at paul_dorsey@dulcian.com