Level: All
Works with: QuickPlace 3.0
Updated: 01-Aug-2002

Going more (Quick)Places

Interview by
Tara Hall

In this month's interview, the second with the QuickPlace development team, we dive deeper into some of the new features in QuickPlace 3.0. Specifically, we speak with Joe Russo, the administration project lead; Helen Dai, the developer responsible for PlaceTypes, Web cache, and migration; and Ken Hirata, the developer who works on Search Places. Be sure to take a look at the interview with QuickPlace product designer Charlie Hill, " ***Going (Quick)Places with Charlie Hill****," for more on QuickPlace 3.0 features.*

Joe Russo

In QuickPlace 3.0, the Admin Utility is being replaced by the QPTool. What is the QPTool and why is it replacing the Admin Utility?
The QPTool provides administrators with the functionality to manage the QuickPlace system. In this release, we introduce a new concept called the QuickPlace Service. The service comprises a catalog of QuickPlaces that you can distribute among various servers in a particular deployment. QPTool allows you to query the service, the QuickPlace servers, and other QuickPlaces, while providing administration functionality at various levels.

In release 2.0.8, we shipped the Admin Utility, a Domino application intended only for the NT platform. It tried to embrace the QuickPlace user interface concepts, but it didn't completely. Initially, we intended to deliver it for an out-of-cycle release, meaning not alongside a particular QuickPlace release. Instead, we shipped it as part of 2.0.8. As a result, it was built entirely separate from the QuickPlace server code, which presented us with a bunch of problems. For instance, any technology that we put into QuickPlace we also needed to reflect in this utility, which created a lot of work for no apparent gain. When we started release 3.0, we envisioned that QPTool would provide flexible functionality and would leverage what we had already built into the server as well as the new features for this release.

Our architecture leverages core components of the server to build functionality into QPTool. We expose the tool through a command-line interface, so administrators can run it like any other command-line tool. QPTool leverages anything that we built into the server. For instance, when we want to change user names, we engage in the directory code that we built into the server rather than having a separate application to mirror the directory code. Additionally, we adopted a model in which we embrace standard languages to build QuickPlace and QuickPlace functionality. What we designed in the QPTool was not only a command-line interface but also an XML interface, so you can feed this tool an XML file representing QuickPlace object streams with actions, and the QPTool command can do whatever you tell it to do.

The 2.0.8 Admin Utility can query a particular server for information about QuickPlaces on the server. For instance, you can query for all QuickPlaces that are greater than a specified size, but it's a very canned approach. Designing around a simplified UI restricted us in terms of the kind of selection logic you could use for QuickPlaces

© Copyright IBM

that are so old, for QuickPlaces that are so big, and so on. In 3.0, we provide a reporting tool that's part of QPTool. The reporting tool queries the service and generates an XML output matching your request. You can mimic the same functionality that you had in the Admin Utility, like show me all QuickPlaces greater than a specified size, show me QuickPlaces that haven't been accessed since a given date, and so on. But you can also build a much more robust report with QPTool.

You can also take the XML file output from this report and feed it to another QPTool command. For instance, if an administrator wants to remove QuickPlaces that haven't been used in 60 days, he can run a report listing those QuickPlaces. Then he can feed that output to the QPTool Archive command to archive those QuickPlaces. With the same XML, he can also delete those QuickPlaces.

We want to give administrators the tools to build their own applications to process whichever administrative functions they want rather than giving them a tool that dictates the application.

One of the other drawbacks to the Admin Utility is that it was built for NT only. But in this release, the QPTool supports IBM AIX, Sun Solaris, as well as the zSeries and iSeries—once they come out with their versions of this release.

You mentioned catalogs and this was something Charlie Hill had spoken about. What can you tell me about catalogs?
The catalog is one of the biggest features that we've added to this release. It has two purposes. One purpose is to provide a definition of the service, so an administrator can deploy a number of QuickPlace servers, register them in the service, and then have a coordinated deployment of QuickPlace. The second purpose is to provide a portal and UI for users who belong to many QuickPlaces.

When an administrator turns on the catalog, he has a QuickPlace server to define the service and can register one or more servers in the service. We encourage administrators to use an existing directory for QuickPlace so that they can: one, take advantage of all the tools that the directory offers, and two, leverage the existing directory. We also recommend having a dedicated Place Catalog server and a dedicated directory server.

Most organizations have an existing directory, so we can work with that. You connect your QuickPlace server to the directory running the QuickPlace Service.

Users can connect to the QuickPlace server and create places. They use their credentials from the directory to authenticate with the service. The catalog lists members of QuickPlaces to determine who belongs to which place. These are stored in the catalog as the distinguished names from the directory.

There are a lot of directory integration changes for 3.0. Can you give me an overview of those changes?
We found that we tried to do too many things in past releases of the product. In one of them, we had the concept of no directory, so QuickPlace provided total directory management for each place. We realized—and our customers told us—that it's not helpful to have that because you don't have a directory to manage each place. Customers preferred that we use an existing directory.

We had two choices for directories. One was Domino, meaning we use the Domino API to query the directory.

© Copyright IBM

That choice was obvious because of what our platform was built on. But we also used NT domains and standard NT networking in our directory. The problem with the latter is that it's not truly a directory. NT keeps track of users and passwords, and it can provide authentication, but it doesn't provide any other directory services. For instance, other contact information about the person—first name, last name, email address, phone number—and all the other things that you need that are stored in a true directory aren't available in NT.

The other choice was LDAP. It was obvious to us that LDAP was really the right directory choice because it provided the directory services we need. And it was also very good at leveraging the Domino Directory and NT domains. For instance, Microsoft Exchange and Active Directory leverage NT networking. We can use the same user name and password that the user wants to use (from their NT network) and take advantage of the other directory services. It became obvious that we should use LDAP for this and not use NT domains.

Domino provides an LDAP service, so we use that. It allows us to focus on providing directory features as opposed to trying to provide a lot of flavors of the directory. In this release, we focused strictly on LDAP. If you have a Domino Directory, we work with that, but we strongly encourage you to turn on the LDAP service. With Domino, that's not a complex thing to do—just load the LDAP service, then reconnect the QuickPlace servers through LDAP to the Domino server, and you're in business.

That's one of the big changes that we wanted to make—focusing on LDAP. But we also want to make our use of LDAP more robust. We've taken the LDAP service and built some more customizability into it. We built in schema mapping and filter mapping that allows you to extend QuickPlace's use of LDAP, that allows QuickPlace to work better in an LDAP environment, and that allows you to tailor a particular LDAP implementation to QuickPlace.

In addition to that, we support SSL for communicating between the LDAP server and the QuickPlace server through a secure communication channel to provide much better security for directory look-ups and authentication. Many customers told us that the method that we had for look-ups and the manner in which we approached the whole UI experience wasn't quite right, so we redesigned that for the service. When we originally built QuickPlace, we built it with a very standardized LDAP approach, so we hard-wired it for particular entries in a particular type of LDAP. But many customers had directories that didn't have the standard mapping between objects and attributes and QuickPlace objects and attributes. In this release, we added the ability for schema mapping. We lay out a bunch of values that QuickPlace needs from the directory—first name, last name, phone number, and so on—and then the administrator can build a table that says "when you want to get the first name for users in my directory use this value."

When we perform look-ups, we do them in two ways. First, when users add members to their place, they want to add either groups or people. LDAP has a language for performing look-ups, and we take the strings that users provide, apply a particular filter, and send the request off to the directory. We wanted the flexibility to enable an administrator to select or code a filter that works better for his directory and to tell QuickPlace to use that filter when we perform look-ups for adding people or groups to a place. Second, we perform automatic look-ups to find a particular user when he authenticates with a place, and again we'll make use of a standard or customized filter.

QuickPlace also has the ability to register users within places. When you install a QuickPlace server out-of-the-box, you can create local accounts in a particular place. It's up to the administrator to connect the directory to the service. The administrator can choose to allow local users to be created within places or to disallow this function, so all users must come from the directory. The reason for that switch is that we found customers occasionally want to invite outside people to participate in a particular place. That's much easier for them to do because the management of the company's infrastructure does not allow someone who created a place to create an account in the corporate directory. Asking administrators to create accounts is a much more intensive process in terms of content and people that own the directory service. Corporations aren't apt to add external users to their places. In that case, the administrator of a place can create local accounts. The administrator can go to his place, add a member who's local, put in some credentials for that member, and tell that person that he can come to this place to do these things with this account.

For those who might be running a mixed-release environment even temporarily, how difficult will it be to connect their existing directory to QuickPlace 3.0?
The directory connections, in theory, haven't changed at all, so it shouldn't present a problem. But how QuickPlace names entities in a place has changed. In past releases of the product, when we added users from the directory, we created specific names for them in the place. We call that scoping them to the place. So, someone can have directory credentials like Joe Russo/IBM, but when an administrator adds the person to a particular place, his distinguished name in the place might be Joe/QuickPlace. That particular user may be in a multitude of QuickPlaces, and he would have different names for each place he's in.

We wanted to represent the correct way for those users to be reflected in places, so we adopted native directory

© Copyright IBM

names in QuickPlace. If your name is Tara Hall/ Westford/IBM and you belong to twelve places, that's your name in all twelve places. There's no confusion—everybody knows who you are, and you know which places you belong to. However, if you have release 2.0.8, you've got the old naming convention. We built release 3.0 to be backward compatible for this name change. So we allow 2.0.8 created places to work with the new 3.0 release, even before the place has been upgraded. In this release, we provide upgrade functionality that takes an existing QuickPlace that's release 2.0.8 or earlier and fixes up all names, so they can work in release 3.0 with the new native directory names.

For instance, let's say you're using an LDAP directory in your release 2.0.8 environment. You keep the same LDAP directory infrastructure in place and install release 3.0. After the install process is done, your server is back up and running, and your users can still access their servers using their old names. That won't change until you upgrade those QuickPlaces on your 3.0 server. Any new places that are created and any new members who are added use the new names with the new representations. The users in the old state have to be upgraded, but the beauty of that is that you can pick and choose when it happens.

One problem we had with scoped names is the way we scope them. We take a person's distinguished name (DN), which in my case is Joe Russo/Westford/IBM. We strip out the common name, which is Joe Russo. We create the QuickPlace DN from that name string so that my name in a particular QuickPlace, like My Place, is JoeRusso/my place/QPcertifier. If there is another Joe Russo that I want to add to that place, I can't add him. I can't add another person with the same name. You can use groups to workaround this limitation, but that's tricky and a real pain in the neck to do.

Using native name format, I can have a place with fifty people with the same first and last name, and they're all uniquely identified because the native directory identifies them uniquely. We do what directories do to distinguish between users.

How has installing QuickPlace on Domino changed the administration of QuickPlace?
We separated the application from the supporting structure of the Web server and made it clear to users that Domino provides the Web service and storage model. QuickPlace is the application that you lay over that. A standalone QuickPlace server is a quick way to get users started on a QuickPlace server, but it isn't a good choice for the long run.The problem with administering a standalone server is that you need Domino to properly administer it. Rather than re-invent the wheel in terms of providing the Domino UI, it's much better for us to work in the Domino world.

In this release, again because we are dedicating our resources to provide real features to the customer, it's just silly for us to provide two identical releases. We spend more thinking power and muscle behind how we work around the problem of administering a standalone server. It's much easier if we install on top of Domino. Now customers know how QuickPlace really works and is deployed.

You mentioned My Places several times. Can you tell me a little bit more? Is that a new feature in 3.0?
Today with a 2.0.8 server, we don't offer any kind of aid to the user to keep track of which places he belongs to and what's going on in particular places. We have notification through emails and the "What's New" feature that sends you email when new things happen, but it's up to you to either keep bookmarks or keep track of the places you belong to. We wanted to provide a portal feature in which you come to a Web page that lists all the places you belong to.

Rather than go to a particular place, which you can still do, you might instead go to a My Places page where you log in. Obviously, this works only for people from the directory who are connected to the place. You use the directory credentials to find out which groups you belong to. Then, My Places queries the catalog to find all the places that you are a member of and provides links to those places.

You can find information about the last accessed time to find what's changed, what size the places were when they were last accessed, and which places are active. That's your portal to various places. When you're inside a place, you can get back to the My Places page. That's been a huge feature that customers have been asking for for a long time. It's part of broadening what we mean by place deployment with this concept of “service” that we're introducing.

There's a lot of XML being used in this particular release. You mentioned that the reporting feature will report back in XML. Are you using Domino XML or are you writing your own XML?
We're writing our own XML. We want to build our own framework because many of the features that we want to implement require a change to the underlying infrastructure. Today, the server and client don't communicate using standard methods because when we first built QuickPlace, there was no such thing as XML. Going forward, we want to build on Web-standard languages and technologies so that third-party developers can develop their own

© Copyright IBM

tools. If we provide them with proprietary technologies, they can't develop their tools, or it would be very expensive. If we give them standard technologies, then anyone who knows these languages can easily develop add-ons. We also want to make the place flexible enough to fit into existing deployments. The world is moving to J2EE, and we want to embrace that technology as well. But we can't throw everything out and rebuild it. Instead, we have to migrate there. In this release, we built an underlying XML structure so that we can do what we need to do moving forward.

We've got J2EE, we've got XML, and we're still relying on Domino for our storage. Domino provides us with a lot of functionality, but we want to migrate this functionality to the new world. Ideally, we'll embrace both in some way—maybe abstracting more from the storage in the background, so we can work in a variety of storages and work flexibly enough with these other tools to do what we have to do. Since we started doing that, we've realized that a lot of the functionality you want to build today uses the technology that we have built into the product. It also has given us the capability to provide simple autonomous functionality for users, but they can build more complex applications using XML.

In this re-architecturing of the server, there are several layers. Can you describe them for me?
At the core, we still have the C/C++ server that has many of the objects we built since release 1.0. We leverage that as much as we can. And we've got a layer that is the interface between Java and C++. We built a Java object model to enable us to move to Web services. The Java object model represents the whole QuickPlace object model from service to objects within a place. On top of that, we built a Java API. That API works with XML to construct Java objects, so it can run the functions that it needs to perform any task.

This layer mimics what you get with Web services. Right now, we're not implementing the other part of Web services, but we want to provide that functionality now without exposing this object layer other than through this API. Again, it's not ready for prime time, but we want to start moving in that direction. Some of the server functionalities are implemented through the Java API. This model gives us more flexibility to build the technology we need to build and to leverage an existing framework that a customer might have.

Our Java API is very straightforward and will allow itself to fit into a Web service implementation without complication. We want to show people who buy the product that this is where we're going. We want to give some level of functionality in this release. But we also wanted to get the release out the door and into people's hands. So we can't implement all of the Web services functionality. But this is a really good step in the right direction.

Helen Dai

What is a PlaceType and how has it changed in Release 3.0?
A PlaceType is a mechanism for software reuse in QuickPlace. You can take a snapshot of a QuickPlace and make it a template or PlaceType for users to create new QuickPlaces. Also, you can control the design and content of a child QuickPlace by refreshing it with updates from the PlaceType. To create a PlaceType, first you edit the PlaceType options of a QuickPlace to allow creation of the PlaceType from that place. Then you make the PlaceType available to users. To update the child QuickPlace, you update the change from the source place to the PlaceType first, then from the PlaceType to the child place.

In QuickPlace 2.0.8, PlaceType follows a cookie-cutter model because of its limited refresh capability. In the new release, we add the ability to perform a central refresh of all places that are based on a given PlaceType. All design or content changes in the source place can be pushed to all child places in a single operation through the central refresh mechanism. In addition, we are providing two levels of refresh—refresh and replace, or forced refresh, for more flexibility and robustness.

Let me give you an example of PlaceType use. Let's say a company wants their QuickPlaces to have the same company look-and-feel. In addition, the company has a common library for all the company QuickPlaces as well. The administrator can create a QuickPlace that has the company look-and-feel and that contains the common library, and then create a PlaceType from this QuickPlace and put the PlaceType on a server where other users create QuickPlaces from it. All the child QuickPlaces have the same company look-and-feel, so there is a commonality in the company's applications. Later on, if the look-and-feel changes because for some reason the company decides to use different themes or the common library content changes, the administrator can refresh all the child places through central refresh. That provides the administrator or the service provider a mechanism to control, maintain, track, and synchronize all child QuickPlaces.

One other thing I want to mention is that PlaceType will support basic clustering in this new release. In release 2.0.8 and earlier, you have to copy the PlaceType to other servers in a clustered environment. With this new release, you can refresh your PlaceType on one server in the clustered environment, and it will replicate to other servers. So you no longer need to copy the PlaceType to other servers.

© Copyright IBM

Does the refresh happen automatically? Can it be scheduled to happen?
In the existing version (2.0.8), it doesn't happen automatically. An administrator can create a PlaceType from a QuickPlace and make it available. The user who creates a child QuickPlace from that PlaceType has to refresh the QuickPlace throughout the process to receive updates. There is no mechanism to automatically refresh the child QuickPlace on schedule. But in the new release, central refresh can run on-demand or as a scheduled server task.

Is there any way for an administrator to determine which content goes out to which child QuickPlaces? For instance, if the administrator wants to secure certain documents, can he prevent them from being pushed out to any of the child QuickPlaces?
Administrators can selectively pick QuickPlaces to refresh, but they can't selectively pick certain documents to update. In the new release, a user can lock off the refresh for a child QuickPlace, so when an administrator runs central refresh, updates aren't pushed into the child place.

It's like a big switch in the QuickPlace. If you don't want to receive updates, you can switch off the central refresh for your QuickPlace. There is no such switch on the individual documents in the QuickPlace. This refresh model in the new release is a more flexible and thorough update model than the existing one. For instance, in the new release, the refresh model has two levels of refresh—refresh and replace, which is a forced refresh. If a document in a child QuickPlace is modified by its user, then under the first level of refresh, the document won't be updated during refresh. It will only be updated under the second level of refresh, which is forced refresh.

During a refresh, can a source QuickPlace inherit changes from the child place?
No, inheritance works only one way; the child inherits from the source. If you could push your changes from the child place to the source place, later on, your changes may be pushed to other child QuickPlaces. It's one thing that you don't want to happen.

How does the refresh and replace feature handle deletions?
The inheritance of deletion was a difficult feature to implement because deletion can cause data loss, and in most cases, there is no way to recover from that. It's tricky determining when deletions should be inherited during refresh. For instance, if a room is deleted in the source place, what should happen to this inherited room in the child place when it's refreshed? Should the room in the child QuickPlace be deleted? Maybe, but what if the room contains documents that have been modified by the QuickPlace user, or what if it contains documents that are not inherited from the PlaceType, but created by users in this QuickPlace instead? You don't want to lose that data. The implementation is difficult because there are lots of details and scenarios that we need to carefully cover.

In addition to PlaceTypes, you're also responsible for upgrade. Are there any guidelines that you have for administrators who are upgrading? Can you tell me a little about what people from previous releases are going to have to do to upgrade to 3.0?
In earlier releases, the install and the upgrade happened at the same time. In other words, your QuickPlace server and all QuickPlaces on it are upgraded when you install a later version of QuickPlace. In release 3.0, we separated the install, server setup, and the upgrade processes. The biggest benefit of doing that is minimized

© Copyright IBM

server downtime. Administrators bring down the server only to install QuickPlace 3.0. After the installation is completed, they can bring the server up again. The server and places/PlaceTypes upgrades can be done without bringing down the server. In addition, upgrade of places and PlaceTypes can be staged to manage server access and resources. For example, if you have 1,000 places on the server, you don't have to upgrade all 1,000 places at the same time. You may selectively upgrade some of the places at one time. Old places (releases 2.0.6, 2.0.7, 2.0.8) can run on a 3.0 server; however, you won't get all the new features in an old place until you upgrade the place to 3.0. Standalone servers are not supported in release 3.0, so you must transition all standalone configurations to the overlay configuration if you upgrade to 3.0.

We will provide detailed documentation on how to make the transition. There are more steps for upgrading a server in this release than in previous releases, but the server downtime is minimized, and you can better manage server access and resources.

Have you run into any interoperability issues between QuickPlace 3.0 and previous releases of QuickPlace in the same domain?
Release 3.0 features are designed to be backward compatible, and like I said already, old places (2.0.6, 2.0.7, 2.0.8) are supported on a 3.0 server. However, in a server cluster, you can't have a mix of 3.0 servers and pre-3.0 servers. The same applies to servers in the same domain; we don't support a mix of versions in the same domain.

You also work on Web cache. Can you tell me what that is and how it works in release 3.0?
Web cache is a way to optimize server performance. It's a new feature of QuickPlace in release 3.0. When a request comes in, it goes to the QuickPlace and through processes that render a page. You need to open many notes in a place to render a complete page for a user. Web cache caches the page in the file system, so the next time the user requests the same page, instead of rendering this page again from the place, the QuickPlace server displays the cached page on the file system, so long as the file copy is still there and valid.

The tricky thing is how do you validate this file copy on the server to make sure it's up-to-date? Web cache looks at a place to see what has changed in the place. It may need to go to one or more places to see if information has changed. It compares the time stamp on the page in the file system with the modification time of the related places. If a file copy is outdated, the server renders the page from the QuickPlace again and overwrites the stale copy with fresh content.

What about disk space requirements? Will administrators need more space on their servers to use Web cache?
You need proper disk space in order to benefit from the Web cache feature. The more space you have, the bigger the cache you can have. If your disk space is too small, then Web cache may remove valuable file copies that would have resulted in a cache hit. You can have a lot of disk space, but the benefits of adding space won't be significant after some point. To determine an optimized disk space size for your server, you need to know how much traffic your server gets. A good rule of thumb is your cache should take at least three days to fill up.

There is a default size of 50 MB set in the QuickPlace, but you can change the size of the cache. You also can specify where you want the cache on your file system directory. And you can also set the time intervals to clean up the cache. In addition, you can also determine who you want the cache to be available for. For instance, you can turn on the cache for anonymous users only.

Ken Hirata

What search changes have you made in QuickPlace 3.0?
The search capability in 2.0.8 only allows you to search within one place, so if you’re a member of more than two places and want to search documents from multiple places, you can't. So we introduced Search Places for release 3.0. It uses Domino Domain Search to extend the search boundary beyond a single place. We take advantage of Domain Search's use of access controls to manage security. So like Domain Search, search results contain only pages to which a user has access.

One important thing to note about Search Places is that the user has to be an external user. A local user cannot find documents from other places even if his user name and password are the same because the local user's identification is bound to one place.

Because Search Places utilizes Domain Search, can Search Places search Domino databases in the same domain?
If the domain consists of Domino databases and QuickPlaces and the domain has only one domain index, then Domain Search searches the Domino databases and QuickPlaces. Search Places filters out Domino documents from the search results, but Domino doesn't filter out the QuickPlace results from a Domain Search. So, if you

© Copyright IBM

perform a Domain Search from a Notes client, you will see QuickPlace pages in the results. But QuickPlace is not designed to be accessed from a Notes client, so you cannot open a QuickPlace page from a Notes client.

Do you have any recommendations for implementing Search Places?
We recommend having either a single dedicated domain for the QuickPlace servers or a single dedicated Domain Catalog server for QuickPlace. If you have one domain for your QuickPlaces and Domino databases, you might want two Domain Catalog servers: one for Domino Domain Search and the other for Search Places. This will prevent the Notes client from receiving QuickPlace results from Domain Search.

When you implement a dedicated Domain Catalog server for Search Places, you must install a QuickPlace server on the Domain catalog server, and you may want to disable creating places on the Domain Catalog server so that the server is used only for indexing and generating Search Places search results.

You can configure a single QuickPlace server for both QuickPlace and the Domain Catalog server, but you may experience performance degradation depending on the size and the number of places on the server. If you have just one QuickPlace server, you can install it on the Domain Catalog server. You can implement Search Places with a single server first, then move to a dedicated Domain Catalog server when you experience performance degradation by installing a new dedicated Domain Catalog server and by disabling indexing on the QuickPlace server. To implement Search Places, you must install a QuickPlace server on a Domain Catalog server.

Do you have any disk space recommendations for having the QuickPlace server and Domain Catalog server on one machine?
Index size will be same with a single QuickPlace server and dedicated Domain Catalog server, but it depends heavily on the contents of database, like large bitmaps, (Microsoft) Office documents, and so on.

Will Search Places support only QuickPlace 3.0 or will it support a mixed-release environment?
If you set up Search Places in a mixed-release environment, you will be able to find a document from previous versions of QuickPlace. In this mixed-release environment, you will also find documents from removed rooms or places if the index is not properly updated. To resolve this issue, you will need to rebuild the index.

Are there any other changes in search?
You can scope a search to a folder, room, or place to limit your search results.

Like Domino, can you customize Search Places?
If you want to customize Search Places, you can use the Search Places API. You can also customize the appearance of the QuickPlace, which customizes the look of Search Places.

QuickPlace 3.0 uses XML to return search results. How does that work?
We defined an XML schema for QuickPlace objects, such as server, place, document, user, and so on. In Search

© Copyright IBM

Places, we generate search results in XML, then transform the results to HTML using XSLT. But you can get search results in raw XML format using the Search Places API.

ABOUT JOE RUSSO
Joe Russo is the Administration Audience lead developer for QuickPlace 3.0. He has a seldom used Electrical Engineering degree from SUNY at Buffalo where he graduated too many years ago to mention. He spent the early part of his career in the imaging and graphics world and then hopped onto the Internet development bandwagon. He lives with his wife and three sons somewhere in Massachusetts, in seclusion from his admiring fans. He hopes one day to write a book that people (other than friends and relations) would be willing to purchase.

ABOUT HELEN DAI
Helen Dai joined Iris/IBM in mid-2001. This is the first QuickPlace release that she is involved in. Helen is responsible for the development of PlaceTypes and Web cache features as well as upgrade. Prior to working at Iris/IBM, she worked for General Electrics Corporate Research and Development. In another life, she trained in Mechanical Engineering and holds a Ph.D from M.I.T.

ABOUT KEN HIRATA
Ken Hirata is an advisory software engineer in the QuickPlace group. His primary responsibility is the globalization of QuickPlace. Before joining the QuickPlace team, he worked for SmartSuite and various Domino-based applications in Lotus Development Japan. Prior to Lotus, he worked on the Japanization of OpenVMS at Digital Equipment Corporation Japan.

© Copyright IBM