Appian Tips & Tricks Webinar:
Data Design for Success

August 27th, 2013

Cindy Cheng
Director of Product Marketing

Gabriel Borges
Senior Consultant

Michael Chirlin
Consultant

Appian

Agenda

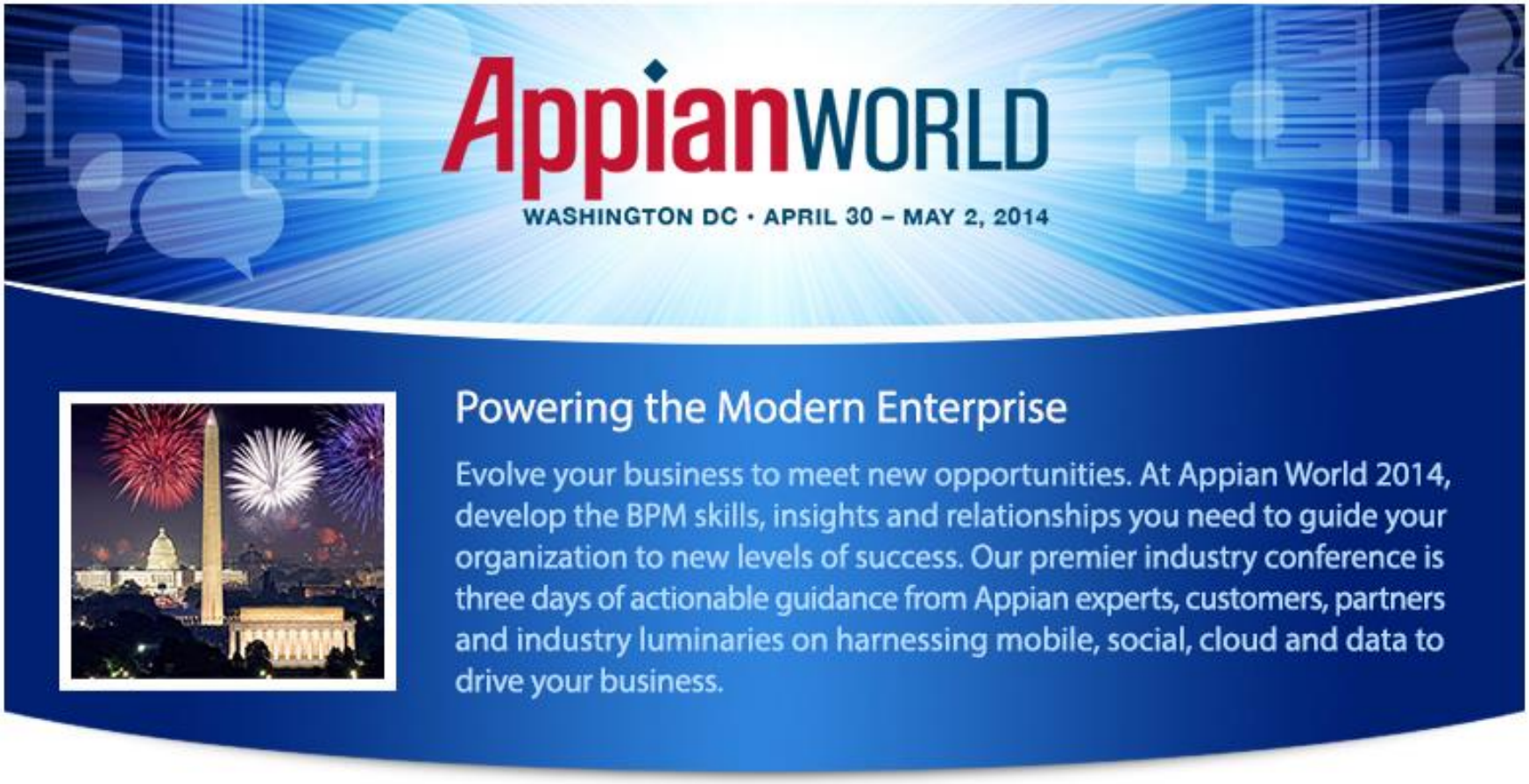
Introductions & Updates – Cindy

Data Design for Success – Gabriel and Michael

- Basic Data Roundtrip
- Data Relationship Examples
- Common JPA Annotations
- Design Considerations
- Demo
- Common Issues
- Relevant Resources

Questions & Answers

Marketing: Save the Date for Appian World 2014



www.AppianWorld.com

at the Grand Hyatt in Washington, DC

Recruiting: Appian is Still Growing & Hiring!

Cloud and Community Services

- Cloud Engineer
- Senior Cloud Engineer

Engineering

- Platform Engineer
- Senior Software Engineer
- Software Engineer (iOS)
- Software Engineer (Technical Lead)
- Software Engineer II
- Software Engineer in Test
- Software Engineer Manager
- Technical Writer
- UX Designer

Information Technology

- Associate IT Engineer

University Recruiting

- Associate Consultant
- Software Engineer
- Software Engineer - Intern

Professional Services

- BPM Consultant
- BPM Senior Consultant - EMEA
- Corporate Technical Trainer
- Senior Consultant
- Senior Consultant - APAC
- Technical Delivery Manager
- Technical Delivery Manager - Financial Services
- Technical Delivery Manager - Insurance
- Training Coordinator

Sales

- Account Executive - Broad Markets (Great Lakes Region)
- Account Executive - Financial Services (Boston)
- Account Executive - Financial Services (UK)
- Account Executive - Melbourne
- Account Executive - Pharmaceuticals (New York or Nationwide)
- Sales Engineer - London - UK
- Sales Engineer - New York
- Sales Engineer - North East Broad Markets
- Solutions Leader- Financial Services

Training: Upcoming Opportunities

Appian

HOME | ABOUT US | PRODUCT | SOLUTIONS | CUSTOMERS | SERVICES | BPM RESOURCES

- Professional Services
- Training Services
- Customer Support
- BPM Partners

BPM Services / Professional Services / Overview

+ Share This

Professional BPM Services

Appian Records and Reports	August 28	2	Online
Partner Bootcamp	September 16	5	Reston, VA, USA
Appian Process Design	September 23	5	Reston, VA, USA
Partner Bootcamp	October 7	5	Reston, VA, USA
Business Essentials	October 7	1	Online
Advanced Appian Process Data Design	October 8	2	Online
Appian Records and Reports	October 10	2	Online
Appian Process Design	October 21	5	Reston, VA, USA

Professional Services: Appian Labs

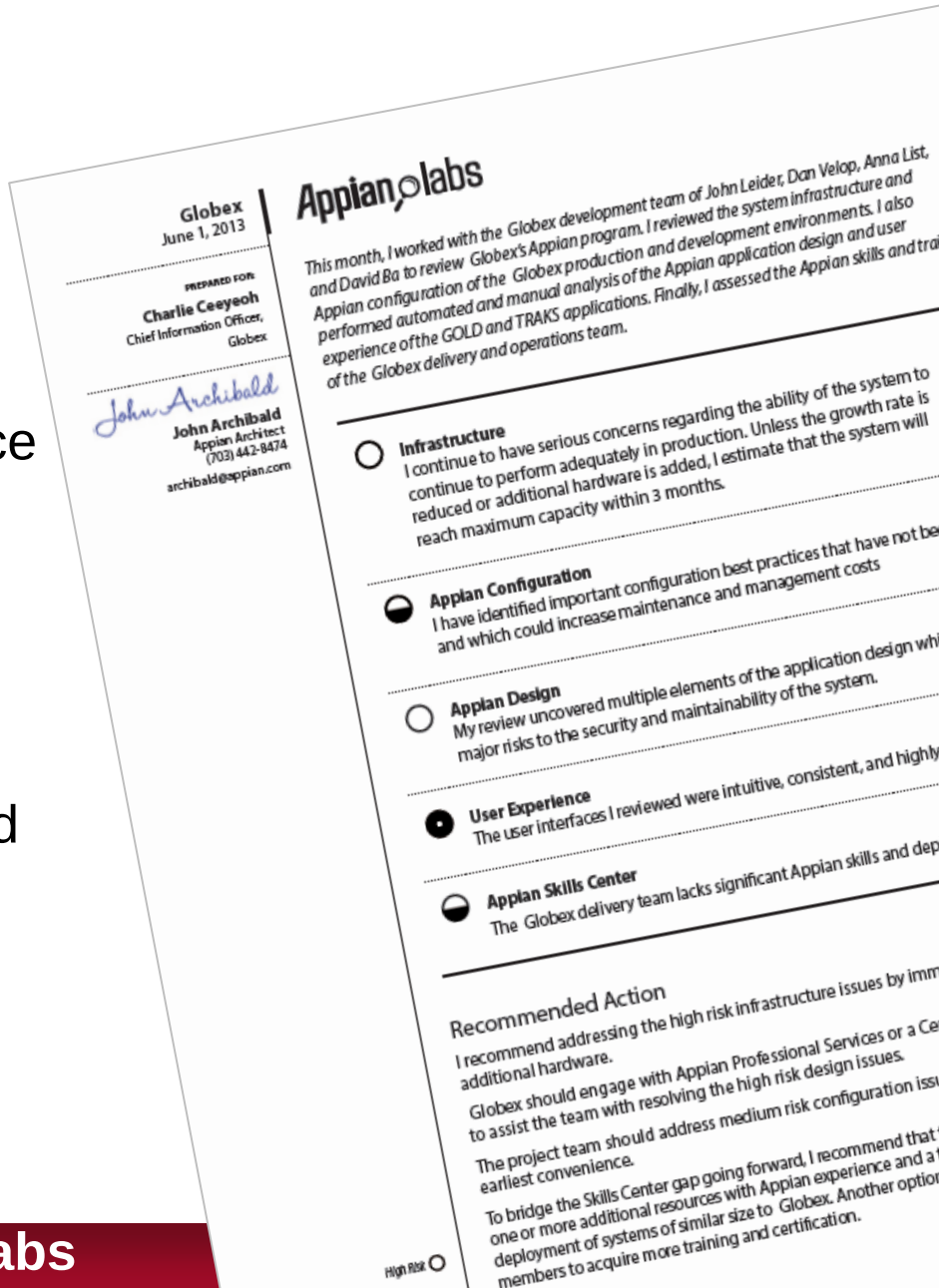
Appian labs

Monthly Health Checks

- Comprehensive Appian application audits for performance testing and assessments
- Track five categories monthly to see trends and identify potential issues
- Custom reports with recommended actions completed by elite Appian Architects

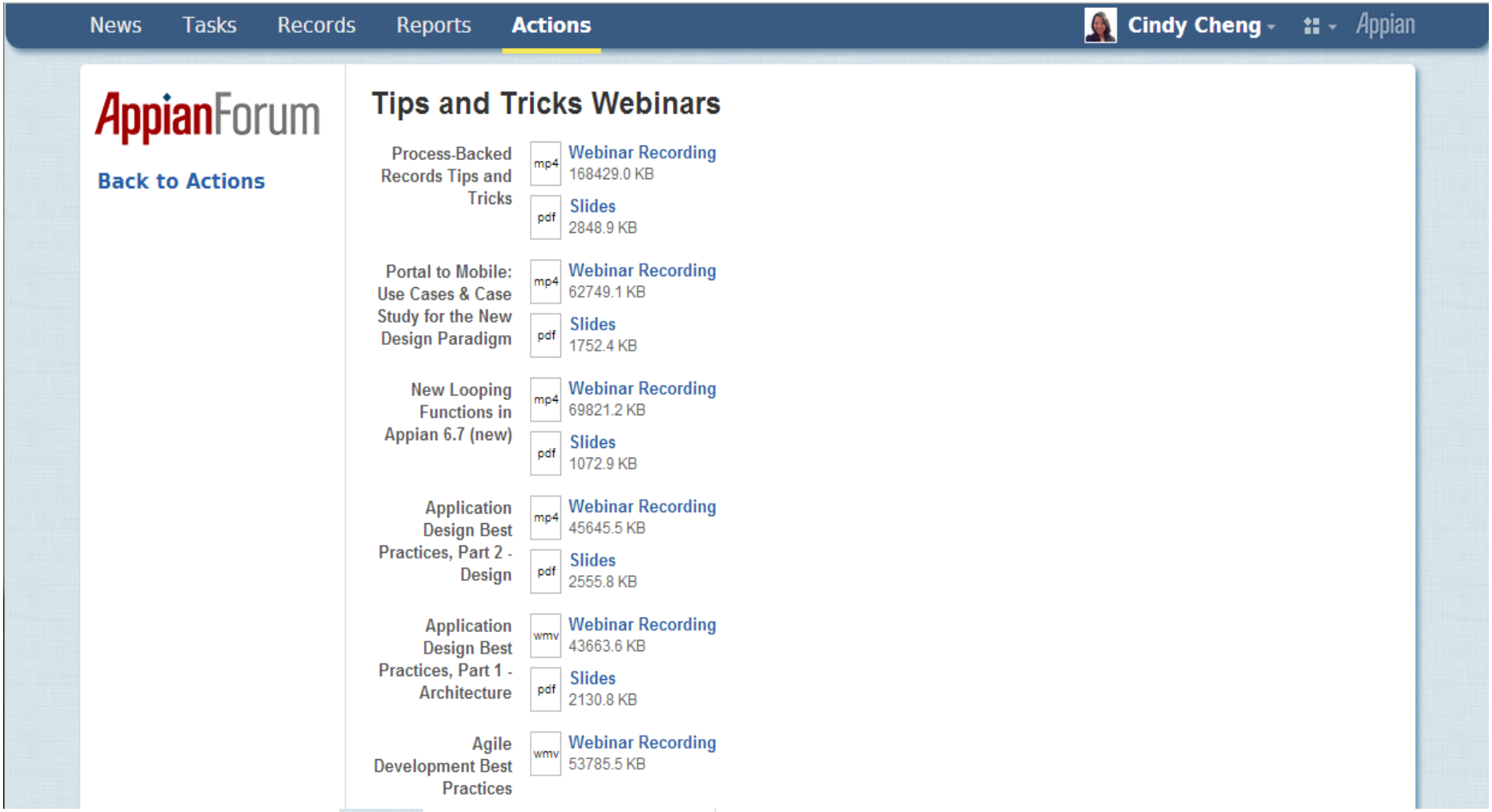
Additional Annual Reviews

- Optional Performance Review
- Optional Architecture Review



Learn more at www.appian.com/labs

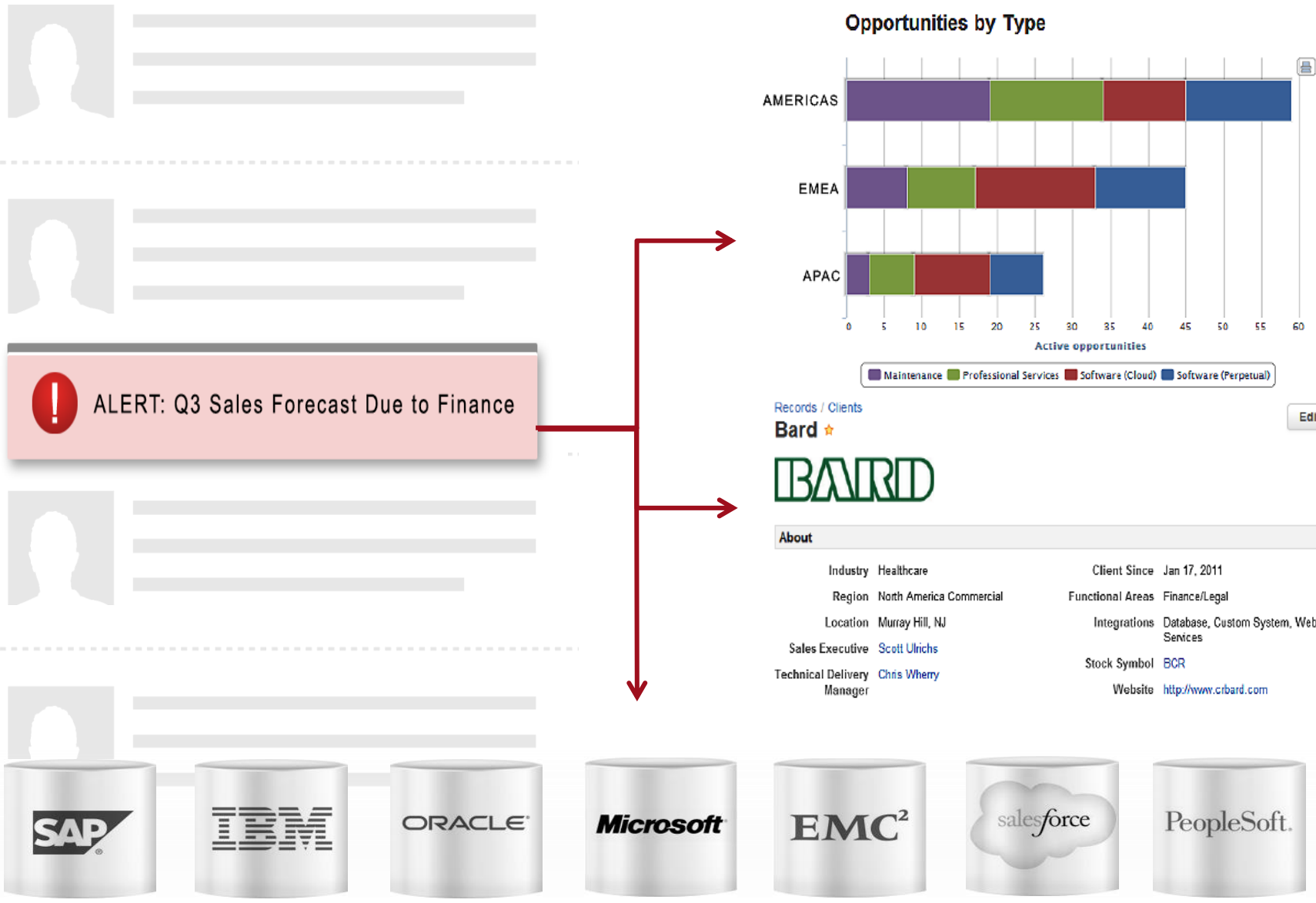
Access Tips & Tricks Recording & PDFs



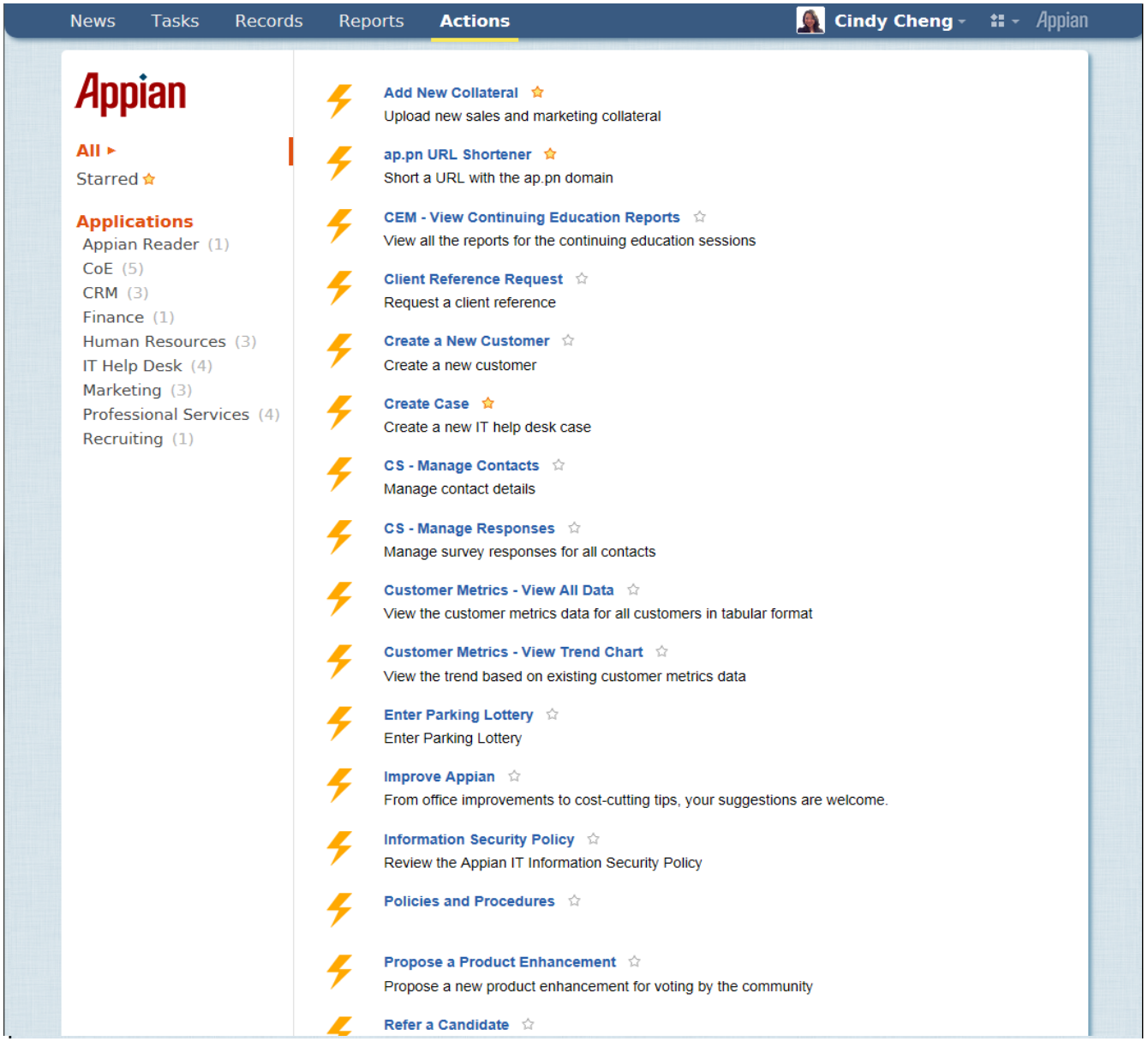
Big Data: Challenges & Opportunities

Where’s my Data?

Access Key Data, Make Better Decisions



Appian Delivers Data-Centric BPM



Real-time activity stream of business events

Track and complete tasks from users and systems

Access records from multiple data sources

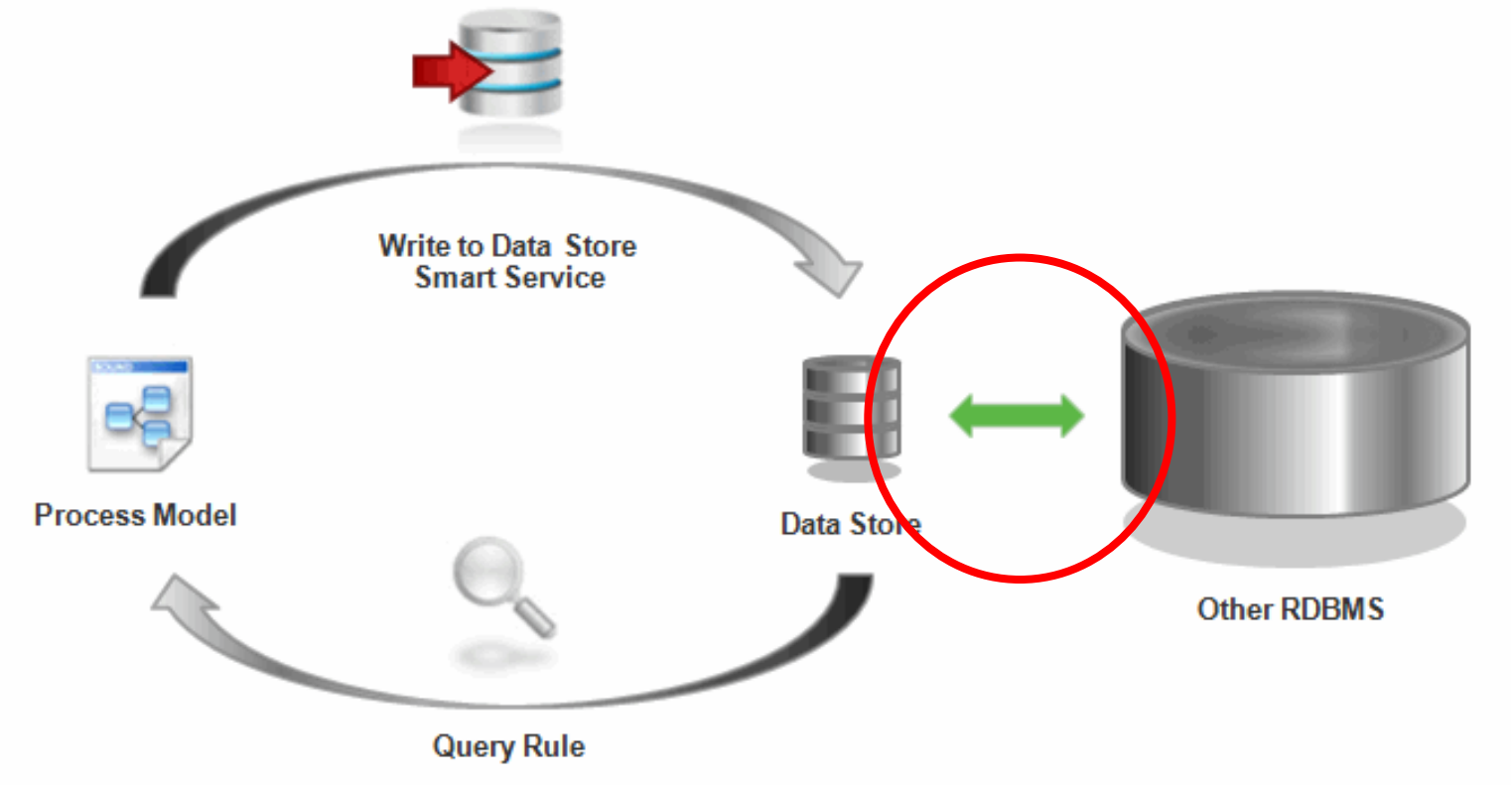
Report on enterprise data

Initiate processes

Data Design for Success

Gabriel

Basic Data Roundtrip

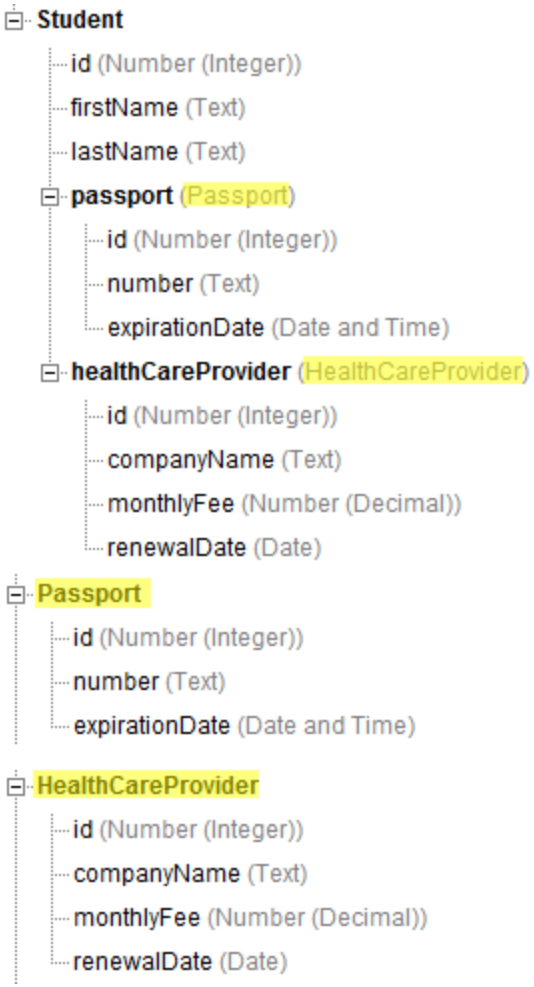


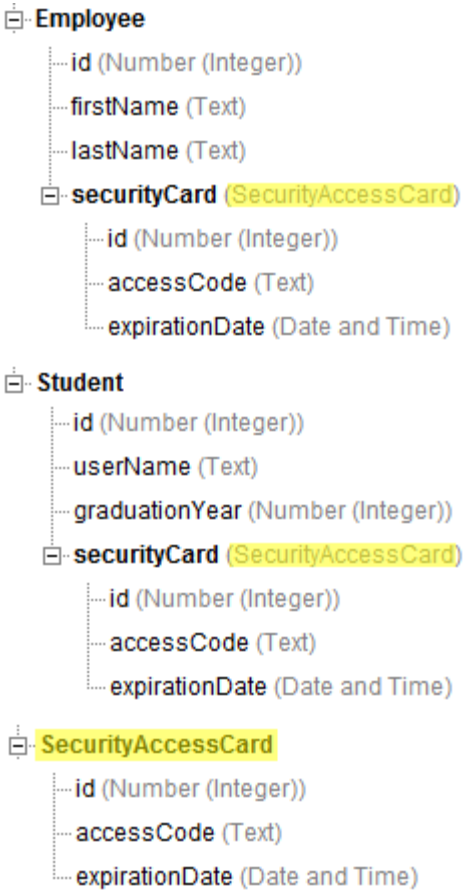
https://forum.appian.com/suite/wiki/latest/File:Using_query_rules.gif

For More Info See: AW2012 Advanced Data Features by Owen Parrish

Data Relationship Examples

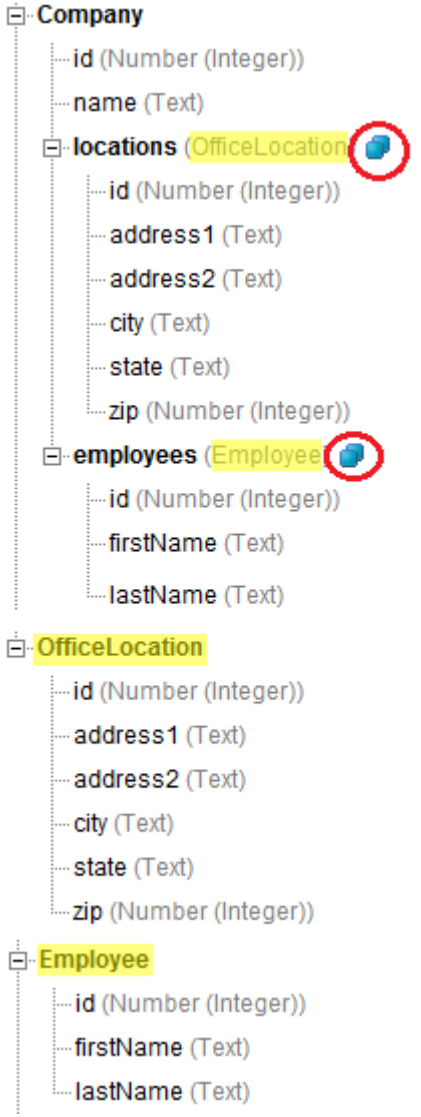
One-to-One Data Relationships (aka Master/Detail Relationship)

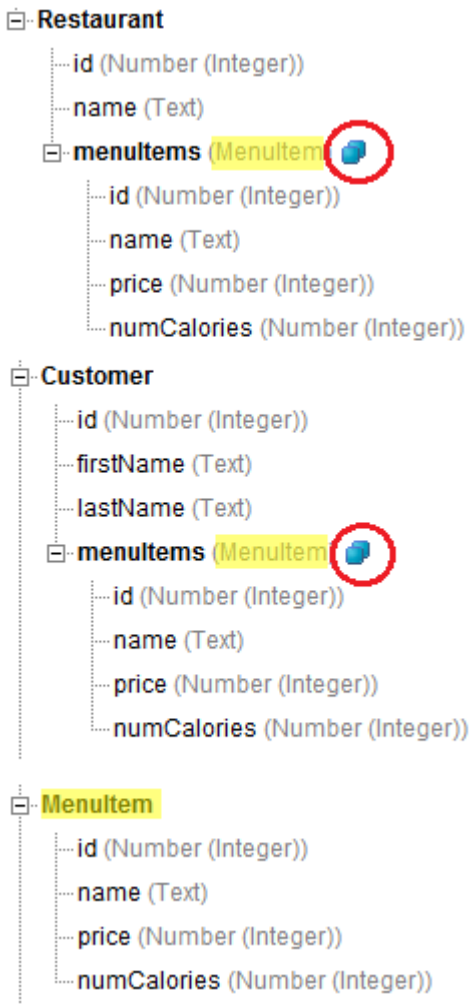




More Info: https://forum.appian.com/suite/wiki/latest/Defining_a_Complex_Data_Type

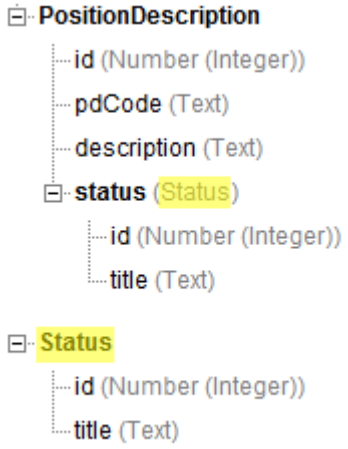
One-to-Many Data Relationships

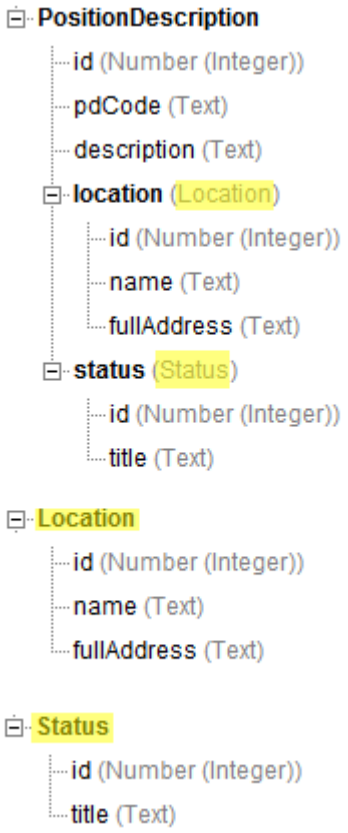


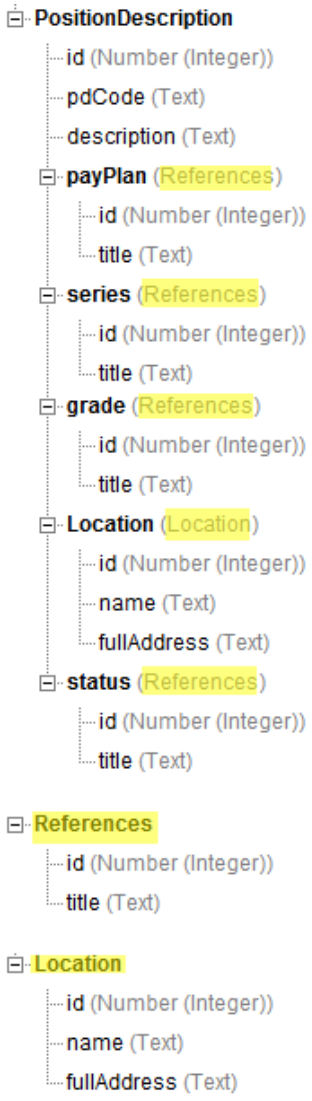


More Info: https://forum.appian.com/suite/wiki/latest/Defining_a_Complex_Data_Type

Many-to-One Data Relationships (aka Lookups)

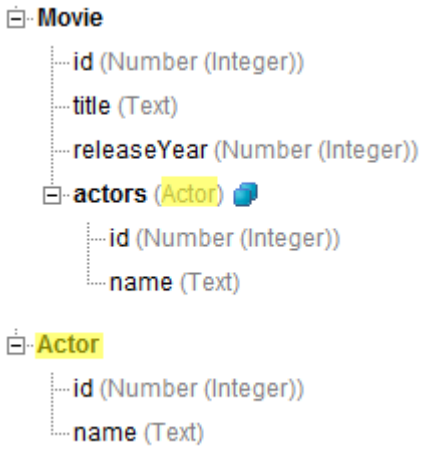




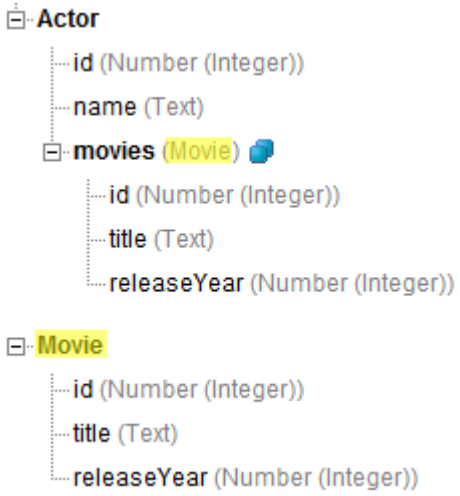


More Info: https://forum.appian.com/suite/wiki/latest/Defining_a_Complex_Data_Type

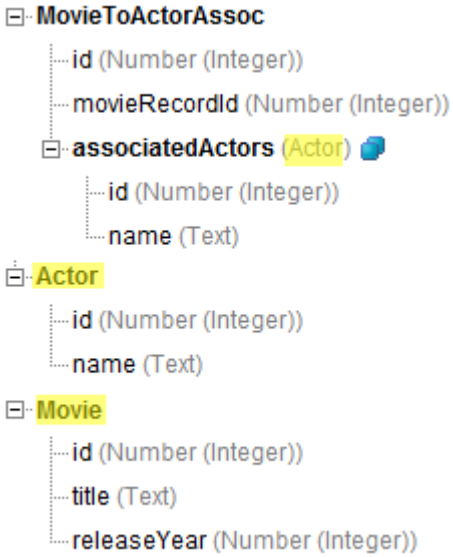
Many-to-Many Data Relationships



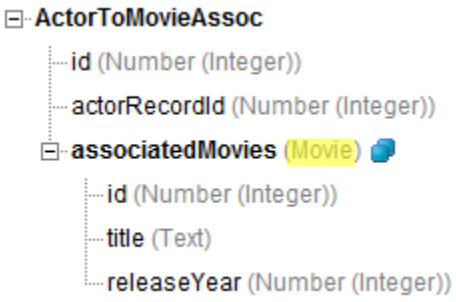
OR



@ManyToMany



AND



Two separate unidirectional relationships
@OneToMany

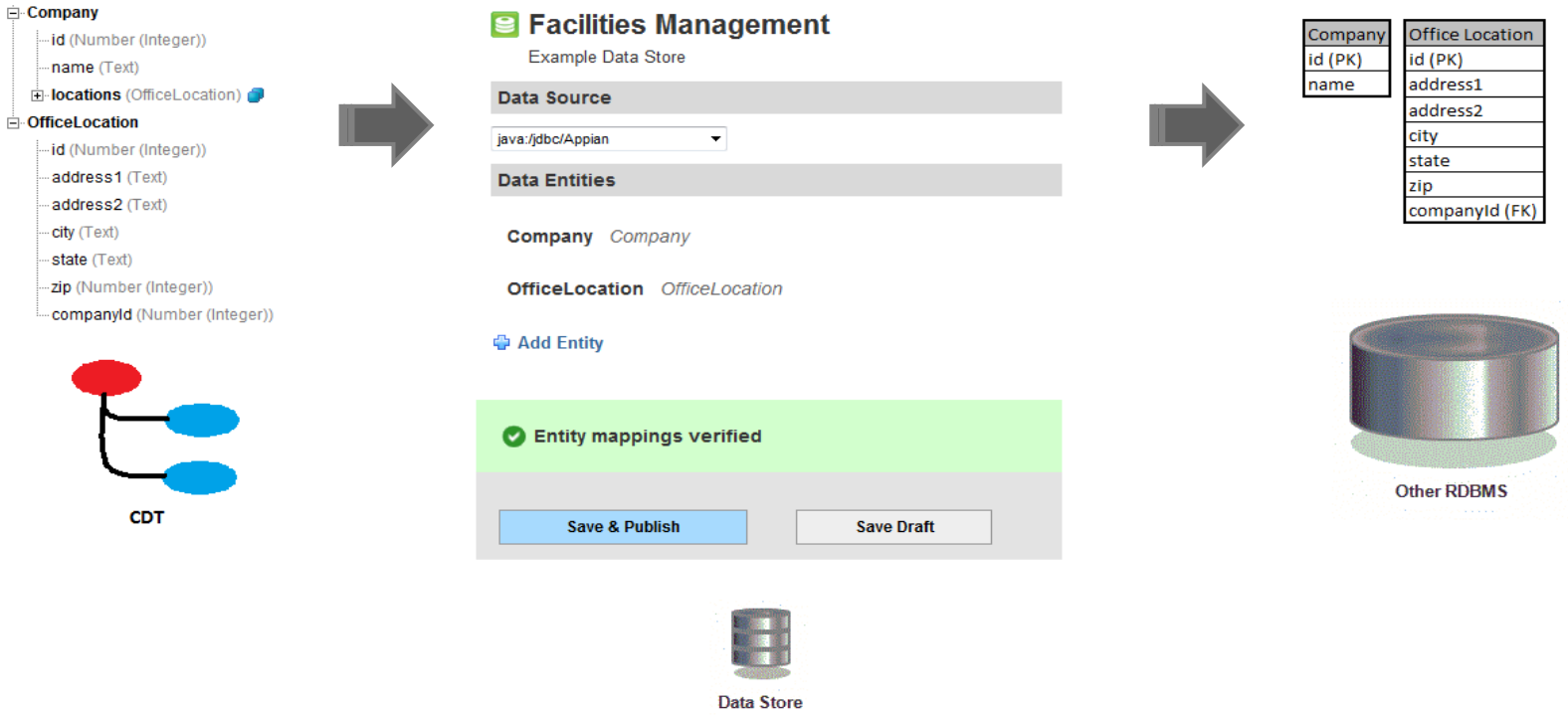
More Info: https://forum.appian.com/suite/wiki/latest/Defining_a_Complex_Data_Type

Common JPA Annotations

Michael

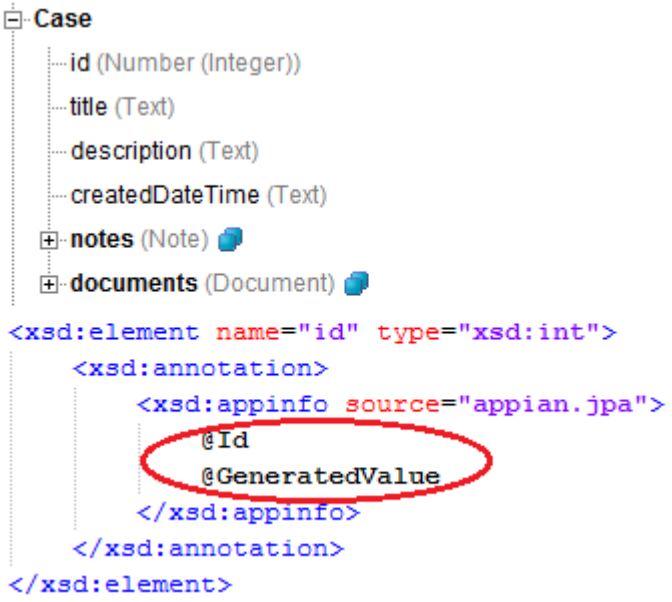
Common JPA Annotations

- Use Java Persistence API (JPA) annotations to customize CDT's and map to or create database tables
  - Map to existing tables
  - Map to/Enforce column definitions
  - Designate primary and foreign keys
  - Advanced control when configuring data associations (1:1, 1:m, m:m, m:1)
    - ie. @JoinColumn, cascadeTypes, uniqueness



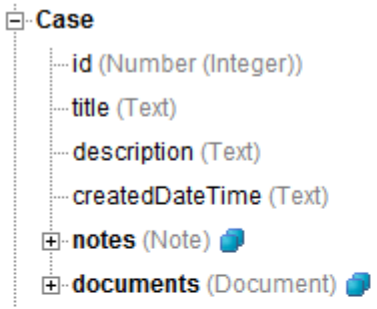
Common JPA Annotations

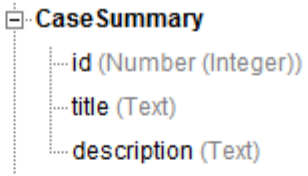
- @Id
  - Without @GeneratedValue
    - Use Case: Another system is generating/providing PK values
    - Designer must account for this
- @Id @GeneratedValue
  - Designates CDT field as PK and as auto-generated
  - When used with Write to Data Store smart service
    - No Id provided: A new record will be inserted
    - Id provided: Existing record will be updated
- @Id Required for Delete from DS Entities smart service



Common JPA Annotations

- @Table(name="CASE")
  - Indicates the table to be mapped for the CDT entity
  - Use to map to existing tables and views
  - Specify table naming convention for new tables
    - If you do not specify a name in the @Table annotation, the name attribute for <xsd:complexType> is used and a naming strategy is applied to it, which will lower-case it and truncate it as needed.
  - Multiple CDTs can be created referencing the same table



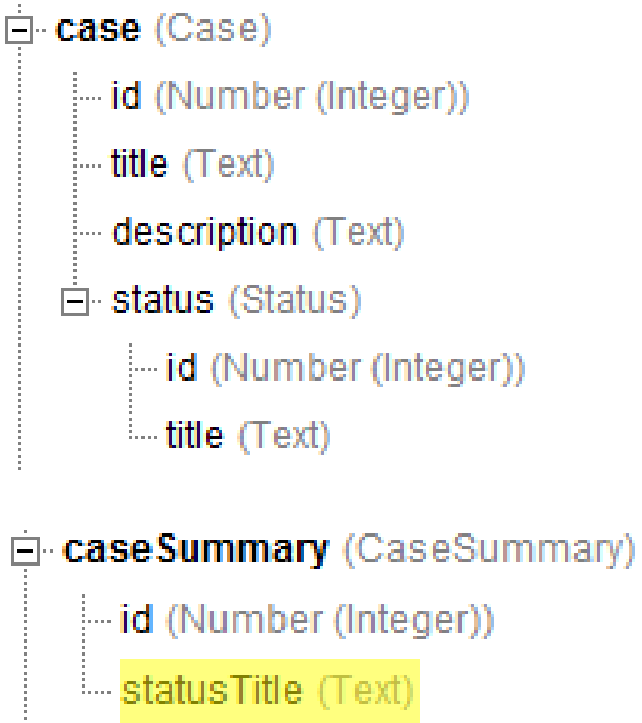



```
<xsd:complexType name="Case">
    <xsd:annotation>
      <xsd:appinfo source="appian.jpa">
       @Table(name="CASE")
      </xsd:appinfo>
    </xsd:annotation>
```

```
<xsd:complexType name="CaseSummary">
    <xsd:annotation>
      <xsd:appinfo source="appian.jpa">
       @Table(name="CASE")
      </xsd:appinfo>
    </xsd:annotation>
```

Common JPA Annotations

- @SecondaryTable
  - Designates the other table from which to retrieve data for the CDT, and the primary key join column
  - Removes the overhead of a nested structure when only a couple fields are needed from the second table
  - Use to optimize performance when child CDT is large but only one or two fields are needed from it




```
<xsd:complexType name="CaseSummary">
    <xsd:annotation>
        <xsd:appinfo source="appian.jpa">
            @Table(name="case")
            @SecondaryTable(name="status")
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="id" type="xsd:int">
            <xsd:annotation>
                <xsd:appinfo source="appian.jpa">
                    @Id
                    @GeneratedValue
                </xsd:appinfo>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="statusTitle" type="xsd:string">
            <xsd:annotation>
                <xsd:appinfo source="appian.jpa">
                    @Column(name="title", nullable="false",
                    insertable="false", updatable="false")
                </xsd:appinfo>
            </xsd:annotation>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
```

Common JPA Annotations

- @Column (name="CASE_NUM")
  - Designates column name
- @Column(columnDefinition="CHAR(15)")
  - SQL fragment used when generating DDL for column
- @Column(table="CASE")
  - If absent, column assumed to be part of primary table
- @Column (nullable=false)
  - Sets column as required
  - Default is true
- @Column(unique=true)
  - Sets a unique constraint on the column

```
<xsd:element name="caseNumber" type="xsd:string" minOccurs="1">
  <xsd:annotation>
    <xsd:appinfo source="appian.jpa">
      @Column(name="CASE_NUM", columnDefinition = "CHAR(15)", nullable=false, unique=true)
    </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
```

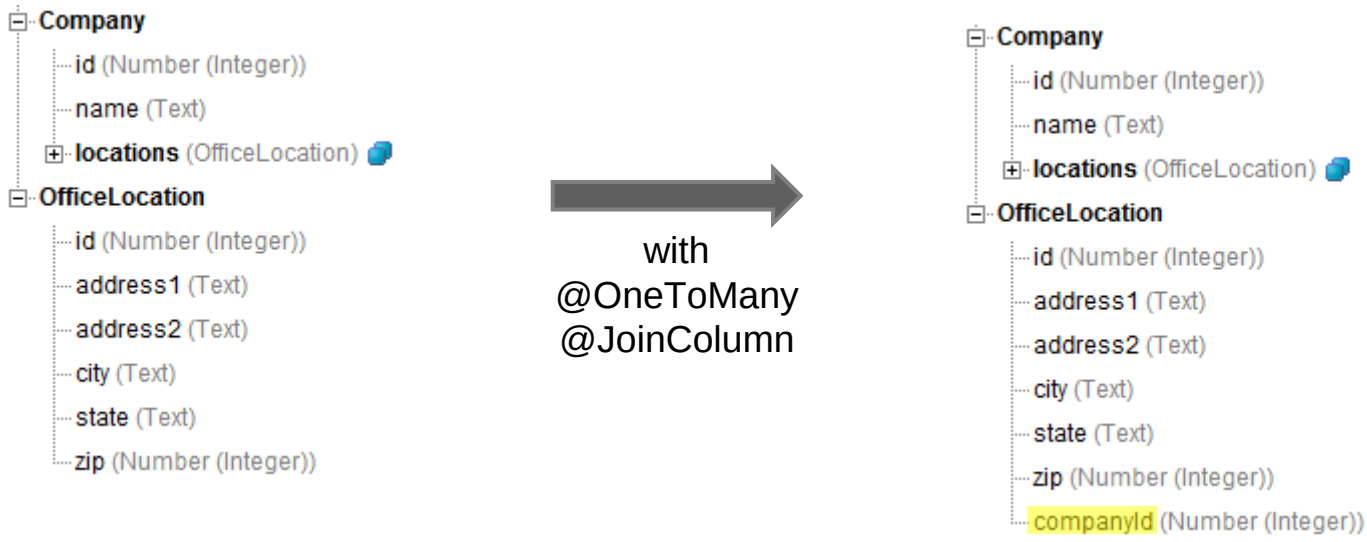
Common JPA Annotations

- @OneToOne (optional=false)
  - Sets field in parent CDT as 1:1
  - Optional attribute set to false
    - Each Employee must have a SecurityAccessCard
- @JoinColumn(name="CARD_ID", nullable=false, unique=true)
  - Identifies column name on Employee table to reference PK of SecurityAccessCard
  - Enforce uniqueness with unique attribute
    - Without, one SecurityAccessCard can belong to more than one Employee
  - Use nullable false to require each Employee to have a SecurityAccessCard

```
<xsd:element name="securityCard" type="SecurityAccessCard">
    <xsd:annotation>
        <xsd:appinfo source="appian.jpa">
            @OneToOne(cascade=ALL, optional=false)
            @JoinColumn(name="CARD_ID", nullable=false, unique=true)
        </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
```

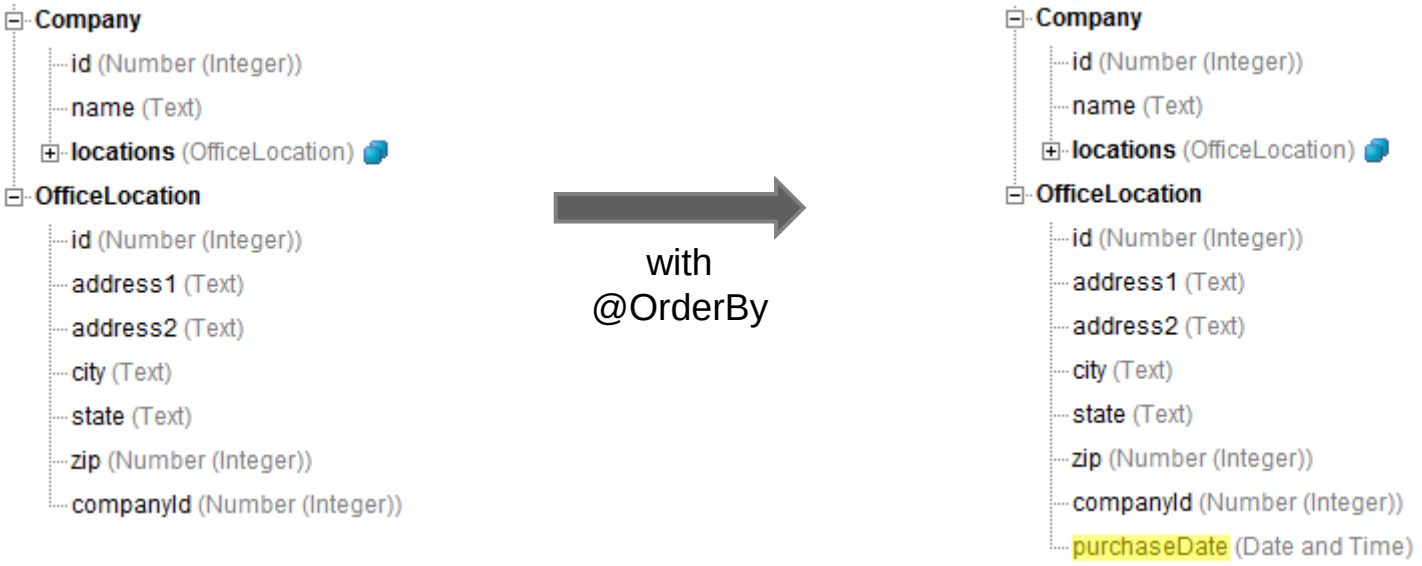
Common JPA Annotations

- @OneToMany(indexed=false)
  - Sets field in parent CDT as 1:m
  - Index column created by default
  - Setting indexed=false prevents index column from being created
- @JoinColumn(name="companyId")
  - Use to designate field in child CDT to use as the join column
  - Field in child CDT maps to PK in parent CDT
  - Allows querying child entity for specific parent



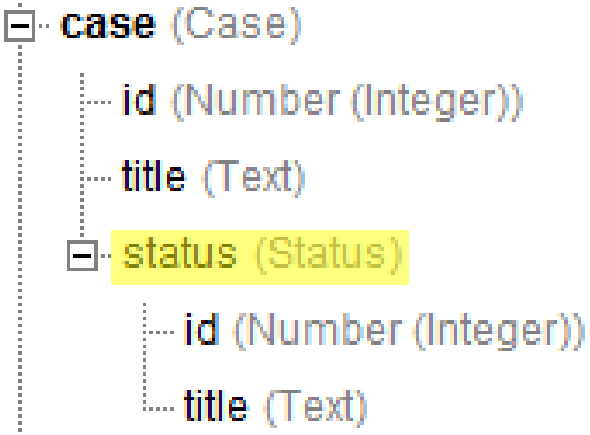
Common JPA Annotations

- @OrderBy
  - Sets order of child element results when parent CDT is queried
  - Default order is ASC
  - Change by setting @OrderBy("PURCHASE_DATE DESC")



Common JPA Annotations

- @ManyToOne (cascade=CascadeType.REFRESH)
  - Sets field in parent CDT as m:1 (aka lookup)
  - Use when the parent must not update the referenced data



- When inserting or updating `case`, only the value of `case.status.id` needs a value
- Write to Data Store smart service outputs the entire `case.status` record.

- Other CascadeTypes
  - MERGE - will update child record with values from parent record
    - eg. If `case.title` is left blank, the value will be overwritten in status table
  - PERSIST (rarely used by itself)
  - ALL (default) - Includes MERGE, PERSIST, and REFRESH
  - NONE- No cascading

Common JPA Annotations

- @ManyToMany
  - Sets field in parent CDT as m:m
    - Eg. Movies and Actors - a movie can have many actors and an actor can appear in many movies.
  - Depending on requirements the CDT can be modeled such that the focus is on movies and actors are members of movies, or it could be modeled that a list of movies belong to each actor.

```
<xsd:element minOccurs="0" maxOccurs="unbounded"
             name="actors" type="Actor">
    <xsd:annotation>
        <xsd:appinfo source="appian.jpa">
            @ManyToMany
        </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
```

or

```
<xsd:element minOccurs="0" maxOccurs="unbounded"
             name="movies" type="Movie">
    <xsd:annotation>
        <xsd:appinfo source="appian.jpa">
            @ManyToMany
        </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
```

  - In both cases a join table is created with an automatically generated name based on the tables that are being joined
- @JoinTable(name = "Actors_Movies", joinColumns="actorId", inverseJoinColumns="movieId")
  - Explicitly names join table and join columns
  - Necessary for matching to existing database

Design Considerations

Gabriel

Flat vs. Nested CDTs

- Where possible use Flat CDTs
  - Better for performance, change management, and easier to design
  - Consider utilizing @SecondaryTable when limited data is needed from the secondary table.
    - Use Case: Displaying just 1 or 2 fields from a child CDT containing 50 fields

- When Nesting...
  - Beware of cascading effects when nesting
  - See CascadeTypes
  - Consider performance implications
    - Limit the data being retrieved
    - Consider using a smaller subset-CDT when you only need some of the data from the full child CDT
    - ie. If main child element contains 20 data fields, but only 6 are needed, create a separate CDT to serve as the child element to be nested

CDTs for Views

- Use @Table annotation to map to Views
- There should always be a primary key
  - Primary Key Field should always be unique
  - If the view is spanning a master/detail relationship, the view can use the primary key of the detail as that will be unique.
    - Not always possible when using outer joins
- Watch out for casting issues when indicating columnDefinition in @Column
  - COUNT
  - AVG
  - GROUP_CONCAT
- Be wary of DDL Update script when Verifying Data Store for Views
  - DDL Updates do not necessarily point you to the problem
- There is a hard limit of 4096 columns per table in MySQL, but the effective maximum may be less for a given table. The exact limit depends on several interacting factors.
  - Limits on Table Column Count and Row Size

Managing CDTs

- CDT Updates
  - Export > Edit > Reimport XSD
    - Must delete existing type prior to re-importing
    - If previous version is not deleted, re-importing updated data type structure will not update type
    - Changes to data type name or namespace result in new data type being created
  - Impact Analysis
    - Automatically update objects using an old version
    - Only lists objects contained within applications
    - Note: Active process instances are not updated

- CDT Application Management
  - <APP> CDTs and Data Stores
    - Increased control of CDT changes
    - Additional Change Management Requirements

CDT Checklist

- Expose primary key
- Implement appropriate JPA annotations
- Avoid overwriting shared reference data types
- Prevent race conditions
- Work to limit disk space consumed by your CDTs in process.
  - Keep database and web-service interactions encapsulated within a sub-process.
  - Delete sub-process upon completion
  - Be aware, every CDT change results in a new copy of the whole CDT in process history
- Check for null prior to using length() on a field to avoid errors

Methods for Building CDTs

- Data Type Designer
- Import XSD file
- Import from WSDL
- Java Plugin

Demonstration

Michael

1. Data Type Designer
2. XSD File

Common Issues

Gabriel

Common Issues

1. Limits on Table Column Count and Row Size
    - Use TEXT instead of CHAR or VARCHAR to store large data sets
        - Keeps CDT grouped appropriately
        - Reduces read/write calls
    - http://dev.mysql.com/doc/refman/5.5/en/column-count-limit.html

2. Mapping CDT Primary Key to an existing data table
    - Use @Id and @GenerateValue
        - Without implementing these annotations, Appian will create a primary key behind the scenes, but it will not be exposed for use within the CDT
    - http://download.oracle.com/otn-pub/jcp/ejb-3_0-fr-eval-oth-JSpec/ejb-3_0-fr-spec-persistence.pdf

3. Storing selections from a paging grid
    - Paging grid populated from Query Rule: Stores record's PK
    - Paging grid populated via todatasubset(): Stores index number of the record's position in the array passed into todatasubset()
    - https://forum.appian.com/suite/wiki/latest/Paging_Grid_Component

Common Issues

4. Mapping CDT fields to columns within an existing table
   - Use @Column annotation
     - @Column(name="CREATED_BY_NM",columnDefinition="CHAR(100)", nullable=false)
   - **Changes to custom.teneo.properties no longer needed**
   - https://forum.appian.com/suite/wiki/latest/Defining_a_Custom_Data_Type#Matching_a_CDT_to_an_Existing_Table_Schema

5. Paging Grid Performance
   - Use query rule on input
   - Reference ac! for dropdown display and value
     - ac!series.id
     - ac!series.name

Relevant Resources

Relevant Resources

- https://forum.appian.com/suite/wiki/latest/Defining_a_Complex_Data_Type
- https://forum.appian.com/suite/wiki/latest/Create_Complex_Data_Types
- https://forum.appian.com/suite/wiki/latest/Managing_Custom_Data_Types
- https://forum.appian.com/suite/wiki/latest/Editing_Custom_Data_Types
- https://forum.appian.com/suite/wiki/latest/Schema_Design_Best_Practices
- https://forum.appian.com/suite/wiki/latest/Data_Stores
- https://forum.appian.com/suite/wiki/latest/Configuring_Relational_Databases
- https://forum.appian.com/suite/wiki/latest/Creating_Query_Rules
- https://forum.appian.com/suite/wiki/latest/Query_Rule_Best_Practices
- https://forum.appian.com/suite/wiki/latest/Post-Install_Configurations#Configuring_Query_Rule_Limits
- https://forum.appian.com/suite/wiki/latest/Paging_Grid_Component
- https://forum.appian.com/suite/wiki/latest/Data_Type_Impact_Analysis
- https://forum.appian.com/suite/wiki/latest/Appian_Scripting_Functions#toxml.28.29
- https://forum.appian.com/suite/wiki/latest/Conversion_Functions#torecord.28.29
- https://forum.appian.com/suite/wiki/latest/Appian_Scripting_Functions#todatasubset.28.29
- https://forum.appian.com/suite/doc/45592
- http://download.oracle.com/otn-pub/jcp/ejb-3_0-fr-eval-oth-JSpec/ejb-3_0-fr-spec-persistence.pdf

Questions & Answers