```
[logoURL]
    logo=1
```

In the delegate class:

```
function logoURL__htmlValue(&$record){
    return '<img src="'.htmlspecialchars($record->val('logoURL')).'" />';
}
```

Now if we load up our application we'll see this logo displayed in the upper left corner of the view tab, and in the column of the summary list.

7.2 Configuring Table Columns

By default Xataface faithfully includes all of the fields in each table as part of the application. I.e. The list tab shows a grid where the columns correspond to columns int the table. Often we may want to append columns to our table that aren't originally part of the table. This is because, in a relational database, many fields that are relevant to records of our table may actually be stored in other tables.

For example, consider our `Posts` table. It contains an ownerID field to refer to the creator of the post, however the list tab doesn't actually include any information about the owner. What if we want the Posts table to also show the owner's email address. It goes against relational database design best practices to actually include an ownerEmail field in the Posts table since this would contain duplicate data with the Profile table. So it makes sense to simply join part of the Profiles table with the Posts table so that we can present this information along with the rest of the Post content.

7.2.1 Background: SQL Joins

If we were using direct SQL to obtain our data set, we would use a JOIN to append columns onto our data set. E.g. the default SQL query to obtain all of the records in the Posts table is:

```
SELECT * FROM Posts
```

To add the ownerEmail column to this dataset we would change the query slightly:

```
SELECT p.*, pr.email as ownerEmail from Posts p
    LEFT JOIN Profiles pr on pr.profileID=p.ownerID
```

We use a LEFT join (as opposed to an INNER join) because we want the ownerEmail column to be added to the result set without actually changing the result set rows themselves (i.e. Posts that don't have a corresponding owner will still be included in the result set).

7.2.2 Overriding Default Select Query

For a given table `A`, Xataface uses a default select query of

```
select * from A
```

to load records from the database. You can, however, override this query with your own query by defining the `__sql__` directive in the global section of the fields.ini file (i.e. before any of the field definitions). Continuing the example of the `Posts` table, if we wanted to add the `ownerEmail` field to the Posts table we would add the following to the beginning of the fields.ini file for the `Posts` table:

```
__sql__ = "SELECT p.*, pr.email as ownerEmail from Posts p
        LEFT JOIN Profiles pr on pr.profileID=p.ownerID"
```

Now if we load up our application and enter the list tab of the Posts table we'll see an ownerEmail column.

Query Constraints

Before you go hog wild and start overriding the select queries for all of your tables, please consider the following constraints:

Given a table `T`. Let A be the set of records returned by the default select query of `T` (i.e. `select * from T`). Let A^* be the set of records returned by the custom select query (i.e. the query specified in the `__sql__` directive of the fields.ini file). Let A_i and A_i^* be the i^{th} rows of A and A^* respectively. The following two invariants must hold:

$$\forall i \in \{1, .., |A|\}, A_i \subseteq A_i^* \tag{7.1}$$

$$|A| = |A^*| \tag{7.2}$$

What this means in English is that your custom query cannot alter the result set in any way, except to add columns. It can not remove or reorder rows, and it cannot remove columns.

Strategies for Meeting Constraints

1. Use LEFT joins only.
2. Join only tables that have at most 1 row corresponding to each row in the default query.
3. If you need to join a table with (or potentially could have) more than 1 corresponding row, use a subquery with set functions to reduce the number of possible corresponding rows.

7.3 MySQL Views

Suppose we want to create a report to show the distribution of Profiles by country. An easy way to do this would be to create a view as follows:

```
CREATE VIEW Profiles_Countries AS
SELECT country, count(*) as num
FROM Profiles
GROUP BY country
```

If we try to navigate to this view in Xataface:

```
index.php?-table=Profiles_Countries
```

We'll receive an error message stating that the table has no primary key assigned. Xataface requires a primary key to be defined in all tables that it works with. This allows it to know which records are unique. Views, unfortunately, don't have any primary keys assigned by MySQL, however, we can use the fields.ini to assign the primary key ourselves.

To add a primary key to the `Profiles_Countries` view, we would add the following to the `Profiles_Countries` fields.ini file:

```
[country]
    Key=PRI
```