IH-69 Shipping Rate Calculator

Requirements and Work estimate

Problem:

1. The algorithm that calculates shipping needs to be modified to account for dimensional weight where possible.
2. Currently, we use the library dotNetShipping to query the carrier web services, but this library does not appear to account for dimensional weight or Billable weight.
3. Currently the dimensions are hardcoded to be 12x12x12. We need to take advantage of any dimensional data in the PRODUCT table to calculate a true Dimensional weight.

User Interfaces Affected:

1. The Shipping Update button on the QuickEstimate modal window of any Product page on InkHead.com
2. The CalculateShipping popup window on any Product page, visible only when logged in to InkHead.com as an employee.
3. The Calculate Shipping modal window activated from the Orders screen of OTIS.

Proposed Solution:

1. Add a method to our internal ShippingAPI to query UPS web services for shipping with BillableWeight (i.e. the greater of Dimensional and Standard weight).
2. Implement the Shipping Calculator as a REST method in both OTIS and INKHEAD sites.
3. Query the database for all information needed to satisfy the UPS web service request.
4. The REST Api will send and receive plain JSON, so the UIs will need to be modified to receive the JSON objects and translate into html.

Alternative -- patch the stored procedures and razor files

1. modify the following SPs to return the Length, Width and Height columns from the PRODUCTS table
    a. sp_Orders_getUPSShippingValuesByOrderID
    b. usp_Customizer_getProductInformationByProductID
    c. sp_Checkout_getUPSShippingValuesByCartID
    d. sp_Cart_getUPSShippingValuesByProductData
    e. sp_Website_getShippingValuesByProductData
2. Modify the following Razor files to use the new information
    a. INKHEAD/NET/shipping/ShippingCalculator.cshtml
    b. INKHEAD/NET/shipping/GetShippingCart.cshtml
    c. INKHEAD/NET/shipping/GetShippingProduct.cshtml
    d. OTIS/Shipping/ShippingCalculator.cshtml
    e. OTIS/Shipping/Display/DisplayShippingManual.cshtml
    f. OTIS/Shipping/ShippingCalculatorManual.cshtml

3. Agree on a set of business rules to determine when to ignore dimension values returned from the database and use a sensible default instead.

Known Gotchas:

1. This will probably depend on IH-70 “Use Carrier tracking data to update Product table in database” for availability of good product shipping dimensions in the PRODUCT table. I haven’t been able to pin down requirements on this yet, but I can start on IH-69 until we have requirements for IH-70.

Work Estimate:

1. Add shipping rate query to UPSClient of ShippingAPI -- **4 points**
2. Create Controllers in both INKHEAD and OTIS apps -- **2 points**
3. Implement business logic layer in ShippingManager -- **2 points**
4. Implement Data layer to get all info for service query -- **4 points**
5. Make modifications to user interfaces -- **4 points** (might be done in parallel)
6. Write unit tests for business logic layer -- **2 points**

Total: **18 points**