Demonstrate ability to use quote, quote item, address, and shopping cart rules in checkout

Interesting *quote* fields or methods:

- `ext_shipping_info` - TEXT 64k, existed in M1, not used. Good place to keep shipping API response?
- `trigger_recollect`
  - quoteResource.markQuotesRecollectOnCatalogRules - Mark quotes that depend on catalog price rules to be recollected on demand. Called on event `catalogrule_before_apply`. Good way to update carts after “hacking” prices?
  - quoteResource.markQuotesRecollect - by product IDs. Mark recollect contain product(s) quotes
    - global plugin after product save, *if price or tier price changed*, trigger recollect
    - adminhtml event after product save, *if status disabled*.
- `is_changed` - not used
  - before quote save - see temporary data-attribute quote `changed_flag` - not used
- `orig_order_id` - not used
- `items_count` - INT “Display number of items in cart”
- `items_qty` - FLOAT “Display item quantities”, e.g. 1.5 kg, 3 same chairs etc.
- `converted_at` - not used
- getItemsByProduct - matching product id and custom options. good to use right after add to cart
  - quote.addProduct
  - after withlist aded to cart, finds and changes store id
  - withlist add all to cart, on error find and remove from cart

Quote extends Model\AbstractExtensibleModel - has support of `extension_attributes`. Register extensions in `extension_attributes.xml`:

```
<extension_attributes for="Magento\Quote\Api\Data\CartInterface">
    <attribute code="shipping_assignments"
type="Magento\Quote\Api\Data\ShippingAssignmentInterface[]" />
</extension_attributes>
```

Use quote repository plugins afterLoad, beforeSave or whenever needed to populate your values. Quote does not utilize `custom_attributes` as it is not EAV by nature.

Quote address custom attributes:

- community \Magento\Quote\Model\Quote\Address\CustomAttributeList - empty, use plugin to add
- EE \Magento\CustomerCustomAttributes\Model\Quote\Address\CustomAttributeList::getAttributes
  - customer address attributes + customer attributes

Quote Item shares 2 models, quote item and quote address item.

Quote item interesting methods:

- checkData - called after adding to cart and updating options
  - quote item.setQty - triggers stock validation
  - product type instance.checkProductBuyState
- `custom_price`

- getCalculationPrice - `custom_price` or converted price
- isChildrenCalculated - product.getPriceType = CALCULATE_CHILD
- isShipSeparately - product.getShipmentType = SHIPMENT_SEPARATELY
- `additional_data` - TEXT
- getOptions
- getQtyOptions:
    - for composite products in cart when sub-products are added, returns array something like `[$subProductId = qty (from option` product_qty_{$subProductId} `), otherSubId = qty, ...]`
        - used in \Magento\CatalogInventory\Model\Quote\Item\QuantityValidator::validate
- compare - \Magento\Quote\Model\Quote\Item\Compare::compare - merge items and add qty instead of new item
    - \Magento\Quote\Model\Quote::updateItem
- representProduct - compares quote item with some new product, checks product id and custom options
    - quote.getItemsByProduct
- compareOptions
    - representProduct

Describe how to modify these models and effectively use them in customizations.

- All support extension attributes
- Quote address supports custom_attributes

Inventory validation

Qty can be updated from:

- cart model.updateItems by:
    - checkout/cart/updatePost with update_cart_action='update_qty'
- quote.updateItem by:
    - admin/sales/order/create
    - advanced checkout
    - checkout/cart/updateItemOptions
    - admin/cart_product_composite_cart/update
    - quote repository.save via Savehandler and CartItemPersister

on qty update:

- quote item.setQty
- event `sales_quote_item_qty_set_after`
- \Magento\CatalogInventory\Observer\QuantityValidatorObserver::execute
- \Magento\CatalogInventory\Model\Quote\Item\QuantityValidator::validate
    - all validation errors can set quote.addErrorInfo, quote item.addErrorInfo
    - check out of stock, parent out of stock
    - when there's sub products with qty options:
        - check qty increments
        - for each sub product - initializer option.initialize \Magento\CatalogInventory\Model\Quote\Item\QuantityValidator\Initializer\Option::initialize
            - multiply parent qty `*` sub qty, check resulting min sale qty, max sale qty, qty

increments, in stock, backorders \Magento\CatalogInventory\Model\StockState::checkQuoteItemQty, \Magento\CatalogInventory\Model\StockStateProvider::checkQuoteItemQty

- when no sub products qty:
  - initializer stock item.initialize \Magento\CatalogInventory\Model\Quote\Item\QuantityValidator\Initializer\StockItem::initialize \Magento\CatalogInventory\Model\StockState::checkQuoteItemQty \Magento\CatalogInventory\Model\StockStateProvider::checkQuoteItemQty

Add to cart

cart model - deprecated.

- checkout/cart/add [product, qty, related_product]
  - cart model.addProduct
    - filter request here, can register own via argument for \Magento\Checkout\Model\Cart\RequestInfoFilter
    - by default filtered out `form_key` and `custom_price`
  - quote.addProduct
  - product type instance.prepareForCartAdvanced
  - get same quote item or create new
  - \Magento\Quote\Model\Quote\Item\Processor::prepare
    - quote item.addQty
    - support request.customPrice -> quote item.customPrice
  - event `checkout_cart_product_add_after`
  - checkout session.setLastAddedProductId
  - cart.save
    - quote collect totals
    - event `checkout_cart_save_after`
    - reinitialize state - remove addresses and payments
  - event `checkout_cart_add_product_complete`
- checkout/cart/updatePost
  - cart.suggestItemsQty => \Magento\CatalogInventory\Model\StockStateProvider::suggestQty - qty increments, min/max qty
  - cart.updateItems
    - events `checkout_cart_update_items_before`, `checkout_cart_update_items_after`
    - quote item.setQty – triggers stock validation
  - cart.save

Describe how to customize the process of adding a product to the cart.

- plugin over product type `prepareForCartAdvanced`
- event `catalog_product_type_prepare_full_options` - custom options in `_prepareOptions`
- plugin over \Magento\Quote\Model\Quote\Item\Processor::prepare - qty, custom price
- event `checkout_cart_product_add_after`

Which different scenarios should you take into account?

- add to cart from catalog
- add to cart from wishlist
- move all wishlist to cart
- merge quote when existing customer has quote, then shops as a guest and logs in
- admin create order
- admin reorder
- configure added product - change custom options