MODS to Zotero mapping

MODS to Zotero mapping

* This is the doImport function for the MODS translator in [scrapers.sql|https://www.zotero.org/trac/browser/extension/trunk/scrapers.sql]
** Use [Scaffold|http://dev.zotero.org/scaffold_tutorial] to test changes

|| Zotero field || Existing import logic || Proposed import logic || Comments ||
| itemType |

```
var partialItemTypes = \["bookSection", "journalArticle", "magazineArticle", "newspaperArticle"\];

var isPartialItem = Zotero.Utilities.inArray(newItem.itemType, partialItemTypes);
    // TODO: thesisType, type

var marcGenres = {
    "book":"book",
    "periodical":"journalArticle",
    "newspaper":"newspaperArticle",
    "theses":"thesis",
    "letter":"letter",
    "motion picture":"film",
    "art original":"artwork",
    "web site":"webpage"
};

for each(var genre in mods.m::genre) {
    if(genre.@authority == "local" && Zotero.Utilities.itemTypeExists(genre)) {
        newItem.itemType = genre.text().toString();
    } else if(\!newItem.itemType && (genre.@authority == "marcgt" \|\| genre.@authority ==
"marc")) {
        // otherwise, look at the marc genre
        newItem.itemType = marcGenres\[genre.text().toString()\];
    }
}

if(\!newItem.itemType) {
    // try to get genre data from host
    for each(var relatedItem in mods.m::relatedItem) {
        if(relatedItem.@type == "host") {
            for each(var genre in relatedItem.m::genre) {
                if(genre.@authority == "marcgt"
\|\| genre.@authority == "marc") {
                    newItem.itemType
= marcGenres\[genre.text().toString()\];
                    break;
                }
            }
        }
    }

    if(\!newItem.itemType) {
        for each(var form in mods.m::physicalDescription.m::form) {
            if(form.@authority == "marcform" \|\| form.@authority
== "marc") {
                if(form.text().toString() ==
"electronic") {
                    newItem.itemType
= "webpage";
                    break;
                }
            }
        }

        if(\!newItem.itemType) newItem.itemType = "book";

    }
}
```

title	

```
for each(var titleInfo in mods.m::titleInfo) {
    if(titleInfo.@type \!= "abbreviated") {
        newItem.title = titleInfo.m::title;
    }
}
```

| | |
| creator, creatorType |

```
for each(var name in mods.m::name) {
    // TODO: institutional authors
    var creator = new Array();
    for each(var namePart in name.m::namePart) {
        if(namePart.@type == "given") {
            creator.firstName = namePart.text().toString();
        } else if(namePart.@type == "family") {
            creator.lastName = namePart.text().toString();
        } else {
            var backupName = namePart.text().toString();
        }
    }
    if(backupName && \!creator.firstName && \!creator.lastName) {
        creator = Zotero.Utilities.cleanAuthor(backupName, "author", true);
    }

for(var role in name.m::role.m::roleTerm) {
    if(role.@type == "code" && role.@authority == "marcrelator") {
        if(role == "edt") {
            creator.creatorType = "editor";
        } else if(role == "ctb") {
            creator.creatorType = "contributor";
        } else if(role == "trl") {
            creator.creatorType = "translator";
        }
    }
}

if(\!creator.creatorType) creator.creatorType = "author";

    newItem.creators.push(creator);
}
```

| | |
| abstractNote |

```
newItem.abstractNote = mods.m::abstract.text().toString();
```

| | |
| journalAbbreviation, publicationTitle, series, seriesTitle, seriesText, seriesNumber |

```
for each(var relatedItem in mods.m::relatedItem) {
    if(relatedItem.@type == "host") {
        for each(var titleInfo in relatedItem.m::titleInfo) {
            if(titleInfo.@type == "abbreviated") {
                newItem.journalAbbreviation =
titleInfo.m::title.text().toString();
                if(\!newItem.publicationTitle)
newItem.publicationTitle = newItem.journalAbbreviation;
            } else {
                newItem.publicationTitle =
titleInfo.m::title.text().toString();
            }
        }
        part = relatedItem.m::part;
        originInfo = relatedItem.m::originInfo;
        processIdentifiers(newItem, relatedItem.m::identifier);
    } else if(relatedItem.@type == "series") {
        newItem.series = relatedItem.m::titleInfo.m::title.text().toString();
        newItem.seriesTitle = relatedItem.m::titleInfo.m::partTitle.text().
toString();
        newItem.seriesText = relatedItem.m::titleInfo.m::subTitle.text().
toString();
        newItem.seriesNumber = relatedItem.m::titleInfo.m::partNumber.text().
toString();
    }
}
```

| | |
| volume, issue, section |

```
if(\!part) {
    part = mods.m::part;
    originInfo = mods.m::originInfo;
}

if(part) {
    for each(var detail in part.m::detail) {
        // volume
        if(detail.@type == "volume") {
            newItem.volume = detail.m::number.text().toString();
            if(\!newItem.volume) {
                newItem.volume = detail.m::text.
text().toString();
            }
        }

        // number
        if(detail.@type == "issue") {
            newItem.issue = detail.m::number.text().toString();
            if(\!newItem.issue) {
                newItem.issue = detail.m::text.
text().toString();
            }
        }

        // section
        if(detail.@type == "section") {
            newItem.section = detail.m::number.text().toString();
            if(\!newItem.section) {
                newItem.section = detail.m::text.
text().toString();
            }
        }
    }
```

pages	

```
for each(var extent in part.m::extent) {
    if(extent.@unit == "pages" \|\| extent.@unit == "page") {
        var pagesStart = extent.m::start.text().toString();
        var pagesEnd = extent.m::end.text().toString();
        if(pagesStart \|\| pagesEnd) {
            if(pagesStart == pagesEnd) {
                newItem.pages = pagesStart;
            } else if(pagesStart && pagesEnd) {
            newItem.pages = pagesStart+"-"+pagesEnd;
            } else {
                newItem.pages = pagesStart+pagesEnd;
            }
        }
    }
}
```

edition	

```
newItem.edition = originInfo.m::edition.text().toString();
```

| | |
| place |

```
for each(var placeTerm in originInfo.m::place.m::placeTerm) {
    if(placeTerm.@type == "text") {
        newItem.place = placeTerm.text().toString();
    }
}
```

| | |
| publisher (and publicationTitle for webpage/website) |

```
if(originInfo.m::publisher.length()) {
    if(newItem.itemType == "webpage" \|\| newItem.itemType == "website") {
        newItem.publicationTitle = originInfo.m::publisher\[0\].text().toString();
    } else {
        newItem.publisher = originInfo.m::publisher\[0\].text().toString();
    }
}
```

| | |
| date |

```
if(originInfo.m::copyrightDate.length()) {
    newItem.date = originInfo.m::copyrightDate\[0\].text().toString();
} else if(originInfo.m::dateIssued.length()) {
    newItem.date = originInfo.m::dateIssued\[0\].text().toString();
} else if(originInfo.m::dateCreated.length()) {
    newItem.date = originInfo.m::dateCreated\[0\].text().toString();
}
```

| | |
| ISBN, ISSN, DOI |

```
function processIdentifiers(newItem, identifier) {
    for each(var myIdentifier in identifier) {
        if(myIdentifier.@type == "isbn") {
            newItem.ISBN = myIdentifier.text().toString()
        } else if(myIdentifier.@type == "issn") {
            newItem.ISSN = myIdentifier.text().toString()
        } else if(myIdentifier.@type == "doi") {
            newItem.DOI = myIdentifier.text().toString()
        }
    }
}

processIdentifiers(newItem, mods.m::identifier);
```

| | |
| url |

```
newItem.url = mods.m::location.m::url.text().toString();
```

| | |
| accessdate |

```
newItem.accessDate = originInfo.m::dateCaptured.text().toString();
```

| | |
| callNumber |

```
newItem.callNumber = mods.m::classification.text().toString();
```

| | |
| archiveLocation |

```
newItem.archiveLocation = mods.m::location.m::physicalLocation.text().toString();
```

| | |
| rights |

```
newItem.rights = mods.m::accessCondition.text().toString();
```

| | |
| lastModified |

```
newItem.lastModified = originInfo.m::dateModified.text().toString();
```

| | |
| notes |

```
for each(var note in mods.m::note) {
    newItem.notes.push({note:note.text().toString()});
}
```

| | |
| tags |

```
for each(var subject in mods.m::subject.m::topic) {
    newItem.tags.push(subject.text().toString());
}
```

source	

```
newItem.source = mods.m::recordInfo.m::recordContentSource.text().toString();
```

accessionNumber	

```
newItem.accessionNumber = mods.m::recordInfo.m::recordIdentifier.text().toString();
```