making a mover RS-compatible

I just did this over the past couple days for CoupledMovesProtocol and CoupledMover. CoupledMovesProtocol runs the entire coupled moves protocol for 1000 trials; CoupledMover makes a single coupled move. I'm planning to develop CMP further to be compatible with explicit crystallographic waters, but I want to use CM as a part of a longer RS protocol in which waters are simply docked à la Lemmon & Meiler.

incomplete documentation!

some additional notes:

- required methods include parse_my_tag, provide_xml_schema, fresh_instance, clone, create_mover, mover_name, keyname. also need to write provide_xml_schema for the MoverCreator.
- must write a separate .hh file for the MoverCreator.
- there are parse_task_operations and parse_score_function methods pre-written that you have to include in your parse_my_tag method to be able to read a resfile, etc
- Movers can still read commandline options
- you have add lines to src/protocols/init/init.MoverCreators.ihh and src/protocols/init/init.MoverRegistrators.ihh for RS to recognize the mover
- *special note for resfiles*: although the ReadResfile TaskOperation seems way easier to use, for some reason it is no longer fashionable, and the use of the OperateOnResidueSubset TO (often with the Residue_Selectors element) is now encouraged. I guess if you use parse_task_operations, your mover should be compatible with both. OperateOnResidueSubset has nonintuitive logic because of its residue-level task operations; here is how I got it to work:

```
        <TASKOPERATIONS>
                <ReadResfile name="rrf" />
                <OperateOnResidueSubset name="subset">
                        <Not>
                                <Index resnums="68-76,123-128,146-151,158-162,192-195"/>
                        </Not>
```

- *special note on PackRotamersMover*: default setting is to design everything! must use with RestrictToRepacking TO.
- include the following line in the provide_xml_schema method to parse task operations for attributes! protocols::rosetta_scripts::attributes_for_parse_task_operations(attlist);

methods I had to add to CoupledMovesProtocol.cc

```
/// additional methods for RosettaScripts
protocols::moves::MoverOP CoupledMovesProtocolCreator::create_mover() const {
    return protocols::moves::MoverOP( new CoupledMovesProtocol ); }

std::string CoupledMovesProtocolCreator::keyname() const {
    return CoupledMovesProtocol::mover_name(); }

protocols::moves::MoverOP CoupledMovesProtocol::clone()
```

```
const
{

return protocols::moves::MoverOP( new CoupledMovesProtocol( *this
) );
}

protocols::moves::MoverOP CoupledMovesProtocol::fresh_instance()
const
{

return protocols::moves::MoverOP( new
CoupledMovesProtocol );
}

std::string CoupledMovesProtocol::get_name()
const
{

return
mover_name();
}

std::string CoupledMovesProtocol::mover_name() {

return "CoupledMovesProtocol"
;
}

void CoupledMovesProtocolCreator::provide_xml_schema( utility::tag::XMLSchemaDefinition & xsd ) const

{
CoupledMovesProtocol::provide_xml_schema( xsd );
}

void

CoupledMovesProtocol::parse_my_tag(
    utility::tag::TagCOP tag,
    basic::datacache::DataMap & data_map,
    protocols::filters::Filters_map
    const &,
    protocols::moves::Movers_map const &,
    core::pose::Pose const &
    )
{

using namespace
core::pack::task;

using namespace
core::pack::task::operation;

main_task_factory_ = protocols::rosetta_scripts::parse_task_operations( tag, data_map );
score_fxn_ = protocols::rosetta_scripts::parse_score_function( tag, data_map );

}

void
```

```
CoupledMovesProtocol::provide_xml_schema( utility::tag::XMLSchemaDefinition & xsd )
{

using namespace
utility::tag;
AttributeList attlist;
protocols::rosetta_scripts::attributes_for_parse_task_operations( attlist );

protocols::moves::xsd_type_definition_w_attributes(
xsd, mover_name(),

"Small backbone and side chain movements"
,
attlist );

}
```

additional #include lines in CoupledMovesProtocol.cc

```
#include <protocols/coupled_moves/CoupledMovesProtocolCreator.hh>

//adding #include lines
#include <protocols/rosetta_scripts/util.hh>
#include <protocols/moves/mover_schemas.hh>
#include <utility/tag/XMLSchemaGeneration.hh>
#include <utility/tag/Tag.hh>
```

CoupledMovesProtocolCreator.hh

```
#ifndef INCLUDED_protocols_coupled_moves_CoupledMovesProtocolCreator_hh
#define INCLUDED_protocols_coupled_moves_CoupledMovesProtocolCreator_hh

#include <protocols/moves/MoverCreator.hh>

namespace
 protocols {

namespace
 coupled_moves {



class CoupledMovesProtocolCreator : public
 protocols::moves::MoverCreator {

public:
            protocols::moves::MoverOP create_mover() const override;
            std::string keyname() const override;
            void provide_xml_schema( utility::tag::XMLSchemaDefinition & xsd ) const override;
        };

    }
}

#endif
```

things to check when you get compiler errors indicating missing symbols or files:

- did you declare the method in the header file?
- do you have all the necessary #include lines?
- for the MoverCreator: did you write a provide_xml_schema method for the Creator in your Mover.cc file?